Cite As:
VERMA, J., KUMAR, S., KAIWARTYA, O., CAO, Y., LLORET, J., KATTI, C. P., KHAREL, R., 2018.
Enabling green computing in cloud environments: Network virtualization approach toward 5G support.
Transactions on Emerging Telecommunications Technologies, pp. e3434.

Enabling Green Computing in Cloud Environments: Network Virtualization Approach Towards 5G Support

Jitendra Kumar Verma[1], Sushil Kumar[1], Omprakash Kaiwartya[2], Yue Cao[2,*], Jaime Lloret[3], C. P. Katti[1], and Rupak Kharel[4]

[1]School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi-110067, India.
{jitendra.verma.in@ieee.org, skdohare@mail.jnu.ac.in, cpkatti@yahoo.com}

[2]Department of Computer and Information Sciences, Northumbria University, Newcastle, U.K.
{omprakash.kaiwartya@northumbria.ac.uk, yue.cao@northumbria.ac.uk}

[3]Department of Communications, Polytechnic University of Valencia, Camino de Vera 46022, Valencia, Spain
{jlloret@dcom.upv.es}

[4]School of Engineering, Manchester Metropolitan University, Manchester,M1 5GD, UK
{r.kharel@mmu.ac.uk}

Abstract

Virtualization technology has revolutionized the mobile network and widely used in 5G innovation. It is a way of computing that allows dynamic leasing of server capabilities in the form of services like SaaS, PaaS, and IaaS. The proliferation of these services among the users led to the establishment of large-scale cloud data centers that consume an enormous amount of electrical energy and results into high metered bill cost and carbon footprint. In this paper, we propose three heuristic models namely Median Migration Time (MeMT), Smallest Void Detection (SVD) and Maximum Fill (MF) that can reduce energy consumption with minimal variation in SLAs negotiated. Specifically, we derive the cost of running cloud data center, cost optimization problem and resource utilization optimization problem. Power consumption model is developed for cloud computing environment focusing on liner relationship between power consumption and resource utilization. A virtual machine migration technique is considered focusing on synchronization oriented shorter stop-and-copy phase. The complete operational steps as algorithms are developed for energy aware heuristic models including MeMT, SVD and MF. To evaluate proposed heuristic models, we conduct experimentations using PlanetLab server data often ten days and synthetic workload data collected randomly from the similar number of VMs employed in PlanetLab Servers. Through evaluation process, we deduce that proposed approaches can significantly reduce the energy consumption, total VM migration, and host shutdown while maintaining the high system performance.

Index Terms

Mobility; Cloud computing; Energy efficiency; Virtualization; VM consolidation.

1 Introduction

Cloud computing is a cost-effective model to deliver computing services such as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) on pay-as-you-go basis. It offers the benefit in terms of efficient IT management and maintenance with changing business needs without even letting know about the upgrading process to the clients. The proliferation of cloud computing led to the emergence of large-scale data centers around the world that host thousands of computing node facilities. These data centers consume enormous amount of electrical energy due to the huge hardware infrastructure associated. According to a report published by Environmental Protection Energy (EPA), the current power consumption by data centers in the US is more than 3% of the total electricity usage by the country that becomes approximately 1.5%–3% of global energy consumption which is growing about 12% every year [1]. Nowadays, data centers worldwide are originating more than 43 million tons of CO_2 per year and adversely affecting the environment towards global warming. Besides of the economic impact, the heat and carbon content generated by the cooling system of the data centers are expected to overtake the aviation industry by 2020 [2].

Fifth Generation (5G) cellular wireless networks is a promising solution towards high data rate demanding applications, improved quality-of-experience, end-to-end latency and lower energy consumption [3]. Liang et al. proposed an

*Corresponding Author: yue.cao@northumbria.ac.uk

Cite As:
VERMA, J., KUMAR, S., KAIWARTYA, O., CAO, Y., LLORET, J., KATTI, C. P., KHAREL, R., 2018.
Enabling green computing in cloud environments: Network virtualization approach toward 5G support.
Transactions on Emerging Telecommunications Technologies, pp. e3434.

information-centric wireless network virtualization architecture to integrate wireless network virtualization and information-centric networking for software-defined 5G mobile wireless networks [4]. The key components of this architecture are radio spectrum resource, wireless network infrastructure, virtual resources, and information-centric wireless virtualization controller [5]. Virtual resources includes content-level slicing, network-level slicing, and flow-level slicing and powers virtual resource allocation. The in-network caching strategy is formulated as an optimization problem with the fact that the gain is not only virtualization but also in-network caching. Wang et al. suggested a packet switching scheme to improve performance of wireless network virtualization by locating frames into user space and placing control and data frame in kernel space [6] . In [7], a hierarchical control based cell clustering model is suggested to integrate heterogenous wireless network and to coordinate among network resources towards optimizing resource utilization.

Virtulization is the central technology behind 5G support and success of cloud computing paradigm for resource sharing to meet rising level of resource demand in daily resource request patterns and facilitating ultra-short latency [8, 9]. Cloud computing helps to reduce carbon footprint of data centers and saves operating cost by improving resource utilization of data centers using virtualization technologies. The improved resource utilization cut downs higher energy consumption required for running cloud data centers. Virtualization technology allows creation of multiple isolated partition of resource capabilities available with computing nodes and make those partitions able to run multiple guest Operating Systems (OS) at the same computing node. These isolated partitions running guest OS share computing resources and are called Virtual Machines (VM). Virtualization technologies enable live migrations of VMs among computing nodes without even disrupting the application services hosted on VMs. The property of live migration offered by cloud infrastructure is the core heart of VM consolidations that helps to improve the resource utilization of a physical host, by migrating VMs from least loaded hosts to normal loaded hosts. In this way, smaller resource capabilities available with physical hosts are able to handle higher incoming workloads and thus reduce computing and cooling energy requirements [10, 11, 12, 13, 14].

In order to cope up with energy inefficiency problem of cloud infrastructure, we leverage the benefit of VM consolidation model for consolidating VMs on relatively small number of physical hosts and switch rest of the hosts in sleep/standby mode to minimize energy consumption [15]. Standard Performance Evaluation Corporation (SPEC) provides benchmark on power consumption data that are utilized in the measurement of total power consumption by a large scale cloud data center as summation of power consumption by the individual physical host [16]. We apply VM consolidation mechanisms by reducing active physical machines with power consumption data of servers on different load levels for measurement of power consumption. In [17], expected energy consumption model is suggested for cloud computing systems to calculate task sojurn time and mean energy consumption for the hosts. It optimize energy consumption using deadline aware task scheduling algorithm by reducing variance of service time based on similar tasks. However, it is impractical to identify the task sojurn time and expected energy consumption until the jobs are finished. Moreover, deadline aware task scheduling causes extra penalty on the system and increases the cost of scheduling. VM consolidation model has been suggested using reinforcement learning that utilize Q-learning method to learn effective control policy for a given task without prior learning of environment [18]. However, Q-learning method turns out to be slower and may trap in the local minima. Ref. [19] provides regression based utilization prediction model that approximates not only the future CPU utilization but also memory utilization of VMs and hosts along with model validation using PlanetLab and google cluster data [20, 21]. However, the model is limited to liner relationship and it is so sensitive to outliers. A utility based model has been suggested that apply frequency settings prior to VM scheduling for consolidation to force scheduling under current actual utilization rather than unreal utilizations [22]. Similarly, our approach triggers server offloading on current server utilization rates if it exceeds the dynamic utilization threshold value. Ant colony meta-heuristic has been suggested to optimize VM placement on physical hosts that uses artificial ants to consolidate VMs on reduced number of active hosts based on three resource dimensions namely, CPU, memory and network Input/Output [23]. However, we focus our study on computation part that depends on CPU utilization rate and major source of energy consumption.

In this paper, we address the problem of energy efficiency in Cloud computing environment by adopting the methodology of VM consolidation models. We propose three novel energy efficient heuristic models to optimize energy consumption on cloud data centers. The main contributions in this work are as follows:

(i) We formulate the cloud data center running cost as a cost optimization problem by considering the costs of running single physical host as a basic unit for cost calculation. It considers cost of power consumption and cost incurred due to penalties associated with hosts suffering from SLA violation due to the host overutilization.
(ii) We formulate resource utilization optimization problem to identify the availability of rooms for accommodating one or more VMs on same physical hosts to understand the consolidation process and optimize VM allocation in cloud environments.
(iii) We propose three novel heuristic models namely, Median Migration Time (MeMT), Smallest Void Detection (SVD) and Maximum Fill Technique (MFT) to optimize the VM allocation and consolidation process.
(iv) We derive a new metric called total host shutdown for performance evaluation of the proposed heuristic models.
(v) We perform simulations for the proposed algorithms by extending CloudSim simulation toolkit to evaluate the performance of the proposed methods in composition with the benchmark algorithms for VM consolidation.

Rest of the paper is organized as follows. Section 2 provides literature review and discuss related work, Section 3 presents the detail of the proposed energy aware virtualization models, Section 4 discusses the performance evaluation of the proposed models compared with the baseline benchmark methods. Section 5 concludes the paper.

Cite As:
VERMA, J., KUMAR, S., KAIWARTYA, O., CAO, Y., LLORET, J., KATTI, C. P., KHAREL, R., 2018.
Enabling green computing in cloud environments: Network virtualization approach toward 5G support.
Transactions on Emerging Telecommunications Technologies, pp. e3434.

2 Related Work

The proposed work is built upon the following considerations. An important feature of online power management is the ability of data center to deal with heterogeneous server configuration. Nathuji and Schwan proposed resource management strategies for VM consolidation that achieve efficiency through local and global policies using live migration of VMs from overloaded servers [24]. At the local level, the system applies power management policies of guest OSs. On the other hand, global manager keeps an eye the current resource allocation by the local manager. Global manager implements its policy on local resources to decide whether the VM placement is required. They did not considered any specific policy for automatic resource management at the global level.

Kusic et al. proposed the idea of Limited Lookahead Control (LLC) where they consolidate VMs as sequential optimization using LLC. The LLC explores for state-space trajectory within a horizon of length h [25]. They used Kalman filter to predict the number of impending arrivals to deal with time varying nature of workload and to perform necessary reallocations [26]. Their model requires simulation-based learning for the application-specific adjustments, Such adjustments cannot be implemented by IaaS Cloud providers such as Amazon EC2. Moreover, the execution time of the optimization controller of their model approaches to 30 minutes even for 15 nodes. Such situation is not suitable for large-scale cloud systems. On the contrary, the proposed heuristic model in this paper aims to achieve high performance for large infrastructure and does not require simulation based learning prior to deployment of application in real cloud system.

Verma et al. proposed the idea of power and migration cost-aware placement of applications in a virtualized environment based on bin-packing problem that suggests for packing not more than $\frac{11}{9}OPT + 1$ balls in a bin where OPT is given by the optimal solution [27] [28]. To represent heterogeneous environment they considered heterogeneously sized bins where hosts correspond to the bins and VMs to the balls. Meanwhile, the proposed algorithm does not ensure the Service Level Agreement (SLA) negotiated due to workload variability in the case of PlanetLab server traces and random workload traces both. Workload traces of PlanetLab servers were collected as a part of CoMon project which is a monitoring infrastructure for PlanetLab servers located at more than 500 places around the world [20].

Gmach et al. proposed host load simulator that forecast workload's time-varying resource requirement based on historical data to suggest appropriate simulated server [29]. The simulated host determines the workload demand gap for the resources by using fair share scheduling strategy static threshold for VM migrations in their work. Similarly, Beloglazov et al. proposed static threshold based approach with minimum migration technique for dynamic consolidation of workload [30]. However, static threshold based approach is not suitable for an Infrastructure-as-a-Service (IaaS) type deployment model of Cloud computing environment because static values based thresholds are not able to cope up with dynamic and unpredictable workload [31]. On the contrary, we have chosen mechanism of choosing upper and lower thresholds to be dynamic in consonance of impending dynamic workload to the hosts.

Host offloading at runtime with resource near the device at the logical edge is a is a challenging task due to the conflict between resource demand and user experience and dynamically changing context that makes for good or bad offloading strategies [32]. Beloglazov et al. proposed modified best fit decreasing heuristic for VM placement and proposed minimum migration policy of VM selection on overloaded hosts [33]. They introduced double threshold (upper and lower) method to keep utilization of hosts between the thresholds. In their another work, they proposed several adaptive heuristics for energy efficient allocation of resources to overcome the problem of static threshold values [15]. Similarly, we adopted double threshold method to keep the utilization level of a host at optimum level where lower threshold is depicted indirectly by choosing minimum utilized host as the underloaded host.

Kistowski et al. proposed several Interpolation methods like Nearest Neighbour Interpolation, Linear Interpolation, Shepard Interpolation and Polynomial Interpolation in to identify the power consumption and compared them to determine the accuracy of the proposed methods to select the best interpolation method using independent reference dataset containing a large set of data points [34]. On the contrary, we apply the standard power consumption details on given CPU utilization of hosts to estimate energy consumption for specified duration under the experimental setup used in this paper.

Live migration is a core function to replace running VMs seamlessly across distinct physical hosts [35, 36, 37]. It refers to the moving of a VMs that are in execution between different physical hosts without even disrupting its services. At the end of live migration process, the CPU, memory, storage, and network connectivity of the VM are transferred from the original guest machine to the destination [38]. In virtualized data centers, live migration provides a significant benefits that are extremely powerful tool for system management in cloud computing such as virtual server mobility without disruption of services, VM load balancing, fault tolerance, power management and other applications [24, 39, 40, 41, 42].

3 Energy Aware VM Consolidation

3.1 Problem Specification

In data centers jobs are majorly classified into two categories including Compute intensive and Data intensive [43]. VM consolidation models for energy efficiency majorly rely upon CPU utilization of servers. This fact makes it difficult to build a precise analytical power consumption model for modern multi core CPUs due to the unavailability of empirical data on all the variables involved. Meanwhile, We identified following issues with existing cloud computing models: (i) IaaS provider needs to find efficient packing solution for VM provisioning across a large pool of servers, (ii) For packing these VMs on non-overloaded hosts, several live migrations are required, (iii) Live migration of VMs leaves several physical

Cite As:
VERMA, J., KUMAR, S., KAIWARTYA, O., CAO, Y., LLORET, J., KATTI, C. P., KHAREL, R., 2018.
Enabling green computing in cloud environments: Network virtualization approach toward 5G support.
Transactions on Emerging Telecommunications Technologies, pp. e3434.

hosts to be underloaded that needs to be either switched into standby mode or turned off, (iv) Waking up servers in active state causes extra penalties that should be reduced by minimizing host shutdowns.

VM consolidation is the process of consolidating large number of VMs on fewer physical hosts to minimize total energy consumption. It signifies that servers are more energy efficient at higher utilization level and less efficient at lower utilization due to the ideal power consumption around 50% power consumption at full utilization level [44]. Two major problems with VM consolidation includes minimizing total cost of running a physical host; and minimizing total cost of the penalty caused by SLA violations.

Every CPU clock consumed during active, idle and sleep period of a host causes power consumption and contribute towards total cost of running physical hosts [45]. Meanwhile, the SLA violations occur when the total demand for the CPU resources exceed the available CPU capacity of a particular physical host. Let us assume that C_p is the cost of power required for running an active physical host and C_v is the cost of SLA violation by a physical host due to excessive resource demand. We define cost of running a cloud data center as cost optimization problem in (1) and resource utilization optimization problem for a host in (2).

$$\left.\begin{aligned} \underset{C}{\text{minimize}} \quad & C = \sum_{i=0}^{m}\left(\sum_{t=t_0}^{T}\left(C_p\sum_{i=0}^{m}a_{ti} + C_v\sum_{j=0}^{m}v_{tj}\right)\right) \\ \text{subject to} \quad & C_p \geq 0, \\ & C_v \geq 0, \\ & a_{ti}, v_{tj} \in 0, 1, \\ & C_p, C_v \in \mathbb{R}^+. \end{aligned}\right\} \tag{1}$$

and,

$$\left.\begin{aligned} \underset{Z}{\text{minimize}} \quad & Z = T_{u,j} - N_{u,j} \\ \text{subject to} \quad & T_{u,j} - N_{u,j} > 0, \\ & T_{u,j}, N_{u,j} \geq 0, \\ & T_{u,j}, N_{u,j} \leq 1, \end{aligned}\right\} \tag{2}$$

where $N_{u,j} = \sum_{i=1}^{n} \Upsilon(VM_{i,j})$, $a_{ti} \in \{0, 1\}$, a_{ti} is either "0" or "1" for inactive and active hosts, respectively. similarly, $v_{tj} \in \{0, 1\}$, v_{tj} is "1" if the host is suffering from SLA violations and "0" if there are no SLA violations for the corresponding host. T is the total period of observation, t_0 is the starting point of time when observation starts. Z is the resource wastage due to gap between utilization threshold $T_{u,j}$ of host j and $N_{u,j}$ is utilization of host j when host runs below the $T_{u,j}$, $\Upsilon(\cdot)$ provides ratio of MIPS allocated to the VM and MIPS capacity of the host, and $VM_{i,j}$ is i^{th} VM executing over host j. Z should be minimized to optimize VM allocation for improving resource utilization of the host and in nutshell global utilization of Cloud data center.

As can be seen in Fig. 1, VM consolidations to fewer hosts takes four steps that S_1, S_2, S_3 and S_4. Global monitoring module keeps track of CPU utilizations of m hosts in step S_1 that declare a host to be overloaded, underloaded or normal loaded. VM selection criteria is applied in S_2 to identify suitable VMs from overloaded and underloaded hosts for offloading them using live migration of VMs in S_3. In S_4, VM allocation takes place to optimize VM placement in physical hosts.

TABLE 1: Power consumption benchmark data (Watt)

Physical Host	0 %	10 %	20 %	30 %	40 %	50 %	60 %	70 %	80 %	90 %	100 %
HP ProLiant G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP ProLiant G5	93.7	97	101	105	110	116	121	125	129	133	135

3.2 Power Model

In cloud computing environment, a typical server rarely experience CPU utilization more than 50% that are provisioned with an additional capacity to deal with load spikes of high variations. Recent studies have affirmed that power consumption are described by linear relationship between power consumption and CPU utilizations where CPU is utilized in timesharing manner by various tasks executing in VMs. For a single server system, the linear relationship can be established as shown in (3).

$$P_{total} = P_{sleep} + P_{nosleep} \tag{3}$$

where P_{sleep} accounts for power consumption while server is in standby state ($P_{standby}$) and server switches from sleep mode to no-sleep mode ($P_{transition}$). Meanwhile, $P_{nosleep}$ accounts for the sum of static power consumption (P_s) when

Cite As:
VERMA, J., KUMAR, S., KAIWARTYA, O., CAO, Y., LLORET, J., KATTI, C. P., KHAREL, R., 2018.
Enabling green computing in cloud environments: Network virtualization approach toward 5G support.
Transactions on Emerging Telecommunications Technologies, pp. e3434.

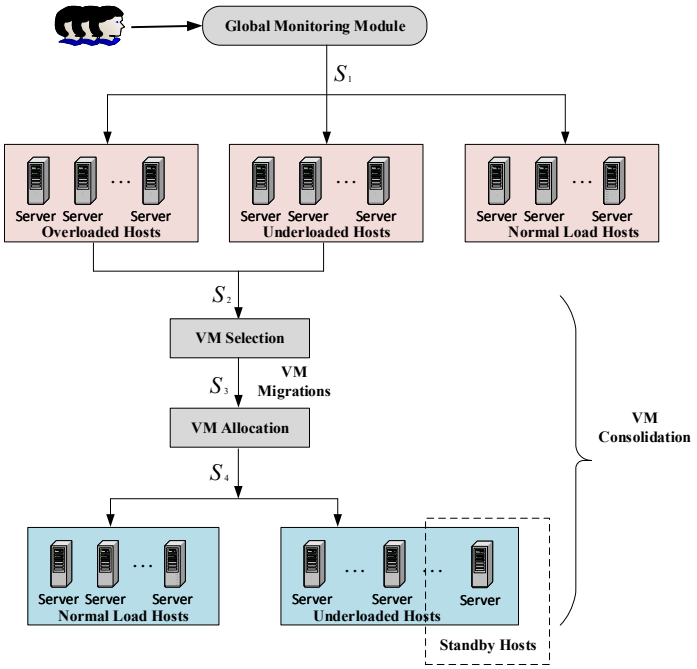


Fig. 1: In architecture shwn above, VM consolidation has two steps of VM selection and VM allocation. Before making decision of VM consolidation, all hosts are monitored for being overloaded, underloaded and normal load. VMs are selected and migrated from overloaded host to the normal loaded or underloaded hosts.

host is ideal and when host is active and performing computation ($c \cdot V^2 \cdot f$) where c is constant, V is voltage applied and f is CPU clock frequency.

In this paper, total power consumption by a data center constitutes server running cost, cooling device cost and network power consumption cost [46]. Power consumption is triggered by CPU utilization ratings (host's states) given by standard power consumption benchmarks [16] for the servers as shown in Table 1. Assuming, P_{host} to be power consumption by physical hosts, $P_{cooling}$ to be power consumption in cooling data center, and $P_{network}$ to be power consumption network equipments, then power consumption by the data center can be modeled as provided in (4).

$$P_{datacenter} = P_{host} + P_{cooling} + P_{netwok} \tag{4}$$

where P_{host} is $\sum_{i=1}^{m} P_{h_i}$, $P_{cooling}$ is $\sum_{j=1}^{p} P_j$, and $P_{network}$ is $P_{ToR} + P_{Agr} + P_{CR}$. In this context P_{h_i} is power consumption corresponding to i^{th} physical host, P_j is power consumption in cooling corresponding to the j^{th} rack of active hosts, P_{ToR} is power consumption by switches on top of racks, P_{Agr} is power consumption by aggregation layer switches and P_{CR} is power consumption by core router of the network.

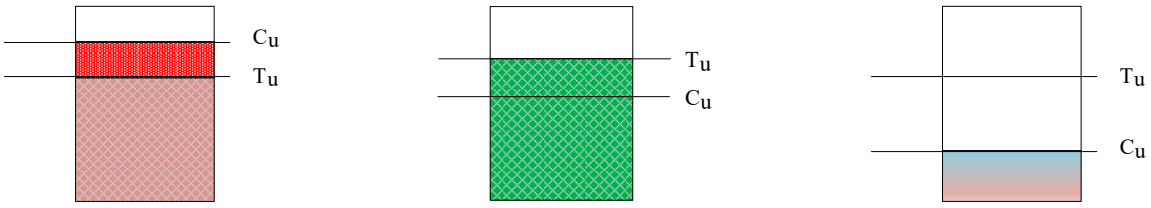
3.3 Live Migration of VMs

In the live migration process, physical memory image is pushed across network to the new destination host in several rounds as pages while the source VM continues running on old host. The pages that are transferred successfully to the new host machine are flagged in original VM at source the host (pre-copying). During the transfer, the pages might get dirty and has to be resent to ensure memory consistency. After several rounds of synchronization, a very short stop-and-copy phase is performed to transmit the remaining pages for resuming execution of VM on new destination. Meanwhile, the cost of live migrations varies significantly for different workloads due to variety of VM configurations and workload characteristics. The relatively small stop-and-copy phase results in to a VM downtime that consists of following two parts and shown in (5): (i) Time spent on transferring remaining pages during the stop-and-copy phase (T_n), and (ii) Time spent on resuming execution of VM at the destination host (T_{resume}).

$$T_{downtime} = T_n + T_{resume} \tag{5}$$

3.4 Proposed Energy Aware Algorithms

In this section, we propose host offloading techniques to bring the host utilization below the host utilization threshold and improve the host utilization to omptimize the energy consumption. Let us assume, $\mathcal{H}_{\mathcal{O}}$ and $\mathcal{H}_{\mathcal{U}}$ are the representation of hosts that are overloaded and underloaded, respectively. The overloaded and non-overloaded hosts are presented in



(a) Overloaded Host with $C_u > T_u$

(b) Non-Overloaded Host with $C_u < T_u$

(c) Under loaded Host with $C_u << T_u$

Fig. 2: Delineating states of overloaded, non-overloaded servers and underloaded host

Fig. 2(a) and Fig. 2(b), respectively where as Fig. 2(c) shows an underloaded host with CPU utilization much less than utilization threshold of the host. The proposed host offloading methods come in action when the host $h \in \mathcal{H}_{\mathcal{O}}$. The Host is considered to be underloaded that is $h \in \mathcal{H}_{\mathcal{U}}$ if the host is experiencing minimum utilization among all the hosts and therefore all the VMs shall migrate from this host. The proposed host offloading techniques, when $h \in \mathcal{H}_{\mathcal{O}}$, are as follows in the subsequent sections.

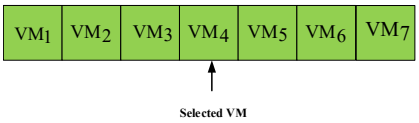
3.4.1 Median Migration Time

The best VM selection policy in [15] was shown to be Minimum Migration Time (MMT) that selects the VM v with the minimum migration time relatively to other VMs. The migration time is the ratio of amount of memory utilized by VM i to the bandwidth spare with physical host j over which i^{th} VM is executing as shown in (6).

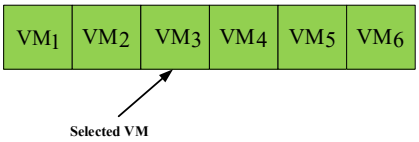
$$t_i^{transfer} = \frac{C_{i,j}^{mem}}{h_j^{BW}} \tag{6}$$

where $C_{i,j}^{mem}$ is memory utilized by i^{th} VM on j^{th} host, h_j^{BW} is spare bandwidth to the j^{th} host, and $t_i^{transfer}$ is VM migration/transfer time.

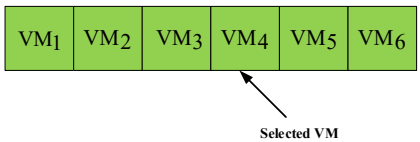
A physical host or cloud sever includes multiple processors and a large quantity of memory that demands a significant amount of electrical power for their operations involved. Much of electrical power consumed by these processors and memory dissipates as heat content resulting into creation of hot-spots. In data centers, hot-spots are a serious issue due to excessive temperature in one or more areas causing damage of equipments. Meanwhile, the MMT fails to address the issue of hot-spots causing few PMs to be highly overloaded. In addition to this, the MMT is unable to address the high VM migrations from the overloaded host necessitating several more live migrations. For instance, an overloaded host is filled up by all the VM instance types (Table 3) that is $t_1.medium$, $t_1.xlarge$, $t_1.small$, and $t_1.micro$ then MMT force the migration of one or more instances of $t_1.micro$ due to its primary criteria of selecting a VM with lowest transfer time to make the host non-overloaded. This condition does not ensure restriction on total live VM migrations. Moreover, the process of live migration of VMs increases the task completion time that are in execution due to the small downtime associated with VM transfer from source host to the destination host that disrupts task execution on VM. High VM migrations exacerbates energy consumption because of the load transfer factor in the part of network, increased task completion time, and VM migrations overhead in MMT.



(a) If $(|VM| \mod 2 \equiv 0)$, and $VM_{median_{\mathcal{E}},l,j}$ is not in migration



(b) If $(|VM| \mod 2 \equiv 0)$, and $VM_{median_{\mathcal{E}},l,j}$ is not in migration



(c) If $(|VM| \mod 2 \equiv 0)$ and $VM_{median_{\mathcal{E}},l,j}$ is in migration

Fig. 3: Median Migration Time Based VM Migration

We introduce Median Migration Time (MeMT) policy to address the issue of MMT. This technique is centered around the memory possessed by a VM. MeMT not only help to cope up with hot-spot issue but also reduce the VM migrations. the hotspot in fewer migrations often leads to a decrease in SLA violation. Therefore, this study proposes VM selection policies

Cite As:
VERMA, J., KUMAR, S., KAIWARTYA, O., CAO, Y., LLORET, J., KATTI, C. P., KHAREL, R., 2018.
Enabling green computing in cloud environments: Network virtualization approach toward 5G support.
Transactions on Emerging Telecommunications Technologies, pp. e3434.

Algorithm 1: Median Migration Time Heuristic

Input: P_m–Power Model, $hostList$
Output: MigrationMap
begin
  Initialize Simulation Parameters
  for $host$ ***in*** $hostList$ **do**
    $T_u \longleftarrow thresholdValue(type)$
    if $hostUtilization > T_u$ **then**
      $vmList \leftarrow getMigratableVms(host)$
      $vmList \leftarrow sortByVmTransferTime(vmList)$
        /* Descending order */
      if $(vmList \mod 2 \neq 0) \wedge VM_{\mathcal{M}_{\mathcal{O},j}}$ **then**
        $vmToMigrate \leftarrow \lfloor \frac{vmList}{2} \rfloor + 1$
        /* Choose VM available at median location when vmList size is odd in length */
      else if $(vmList \mod 2) = 0) \wedge VM_{\mathcal{M}_{\mathcal{E},j}}$ **then**
        $vmToMigrate \leftarrow \lfloor \frac{vmList}{2} \rfloor$
        /* Choose VM available at upper median location when vmList size is even in length and this VM is not in migration */
      else
        $vmToMigrate \leftarrow \lfloor \frac{vmList}{2} \rfloor + 1$
          /* Choose VM available at lower median location rest of the cases */
    $vmsTomgrate.add(vmToMigrate)$
  $migrationMap.add(getNewVmPlacement(vmsToMigrate))$
  $vmsToMigrate.clear()$
    /* Nullify the variable $vmsToMigrate$ after the loop */
  forall the $host$ ***in*** $hostList$ **do**
    if $isHostMinimumUtilized$ **then**
      $vmsToMigrate \leftarrow host.getVmList()$ /* All of the VMs migrates */
    $vmToMigrate.add(vmToMigrate)$
  $migrationMap.add(getNewVmPlacement(vmsToMigrate))$
  return $migrationMap$

that resolve hot-spots spending fewer migrations, resulting in lower energy consumption and SLA violation percentage. For this, VMs running on overloaded PM are ordered by ratio of memory utilized by the VM and spare bandwidth with host. Furthermore, we take the list of VMs on overloaded host and order them by memory-bandwidth ratio and pick the VM at median location of the ordered list. We define three cases of identifying median in different situation of VM list size on overloaded host are as shown in (7). The VM selection function for MeMT is depicted in Fig. 3.

$$meadian = \begin{cases} \mathcal{M}_{\mathcal{O}} = x/2 + 1 & \text{if } x \mod 2 \equiv 0, \\ \mathcal{M}_{\mathcal{E},l} = \lfloor x/2 \rfloor & \text{if } x \mod 2 \not\equiv 0, \\ \mathcal{M}_{\mathcal{E},u} = \lfloor x/2 \rfloor + 1 & \text{if } x \mod 2 \not\equiv 0. \end{cases} \tag{7}$$

where "$x \mod 2 \equiv 0$" and "$x \mod 2 \not\equiv 0$" refers to the situation when x and y are even and odd in length, respectively. $\mathcal{M}_{\mathcal{E},l} = \left\lfloor x/2 \right\rfloor$ and $\mathcal{M}_{\mathcal{E},u} = \left\lfloor x/2 \right\rfloor + 1$ are the lower median and upper median when list size even. Based on VM transfer time from one host to another host, VM ordering in descending order of VM transfer-time is shown in (8).

$$\text{vmList} = \{\forall \alpha, \beta \in V \mid t(\alpha) \geq t(\beta)\} = \left\{ \forall \alpha, \beta \in V \middle| \frac{C_u^{mem}(\alpha)}{h_j} \geq \frac{C_u^{mem}(\beta)}{h_j} \right\} \tag{8}$$

where α and β refer to the i^{th} and $(i+1)^{th}$ VMs on overloaded host P_j, $t_{i,j}(\cdot)$ provides transfer time of i^{th} VM running over j^{th} host, and $VM_u^{mem}(\cdot)$ provides memory utilization by the VM. After application of (7) on ordered list of VMs

Cite As:
VERMA, J., KUMAR, S., KAIWARTYA, O., CAO, Y., LLORET, J., KATTI, C. P., KHAREL, R., 2018.
Enabling green computing in cloud environments: Network virtualization approach toward 5G support.
Transactions on Emerging Telecommunications Technologies, pp. e3434.

obtained, VMs are selected for migration from overloaded hosts in the following manner as shown in (9).

$$v = \begin{cases} VM_{\mathcal{M}_{\mathcal{O}},j} & \text{if } (|VM| \mod 2 \not\equiv 0) \text{ and} \\ & \mathcal{B}(VM_{\mathcal{M}_{\mathcal{O}},j}) = 0, \\ VM_{\mathcal{M}_{\mathcal{E},l},j} & \text{if } (|VM| \mod 2 \equiv 0) \text{ and} \\ & \mathcal{B}(VM_{\mathcal{M}_{\mathcal{E}},l,j}) = 0, \\ VM_{\mathcal{M}_{\mathcal{E},u},j} & otherwise. \end{cases} \tag{9}$$

$$\mathcal{B} : x \rightarrow y, \quad \texttt{where } x \in VM \texttt{ and } y \in \{0, 1\}$$

where $VM_{median_{\mathcal{O}},j}$ refers to the VM available at median location (i.e. $\frac{|VM|}{2} + 1$) when VM list size is odd in length, $VM_{median_{\mathcal{E}},l,j}$ refers to the VM available at lower-median location (i.e. $\frac{|VM|}{2}$) when VM list size is even in length, and $VM_{median_{\mathcal{E}},u,j}$ refers to the VM available at upper-median location (i.e. $\frac{|VM|}{2}+1$) when VM list size is even in length. $\mathcal{B}(\cdot)$ is a boolean function that returns 0 when its argument corresponding to the VM on overloaded host is not in migration. Stepwise illustration of MeMT (Algorithm 1) is summarized below:

1) Step-1: Power model P_m and $hostList$ is supplied.
2) Step-2: Simulation parameters are initialized.
3) Step-3–16: for loop detects overloaded hosts and VMs are arranged in descending order of VM migration time. VM available at median location of the list according to the criteria provided in (7) is set for migration to offload the host if this VM is not already under migration. Furthermore, the host is evaluated for overloading and step continues until the host will become non-overloaded. It continues until all the hosts are processed.
4) Step-16–17: All the migrated VMs are optimally packed/placed on active physical machines according to the constraint $\frac{11}{9}OPT$. After allocation of VMs flush the memory variable containing list of those VMs which are selected for migration.
5) Step-18–21: Hosts are detected for undertilized host. Minimum utilized host among all hosts is declared underutilized host and all VMs from this host are migrated to pack over other active hosts.

3.4.2 Smallest Void Detection Technique

This technique is centered around the overloading detection mechanism that allow to consolidate VMs on hosts to improve CPU utilization of the hosts. Smallest Void Detection (SVD) technique aims at minimizing the gap between CPU utilization threshold and current utilization of CPU when former is greater than later one. We refer this gap as "*Void*" and it is calculated by finding $Void = |T_u - C'_u|$. C'_u is calculated as follows in (10).

$$C'_u = \frac{\sum_{i=1}^{n} ||C_{i,j}^{cpu}||}{||h_j^{cpu}||} \text{ and } T_u > U_c \tag{10}$$

where T_u is CPU utilization threshold for a host, C'_u current CPU utilization for a host, $||C_{i,j}^{cpu}||$ is absolute value of Mips (i.e.) CPU allocated to the host, p_j^{cpu} is Mips owned by j^{th} host.

$C_u - T_u > 0$ indicates that host is overloaded while $C_u - T_u \leq 0$ marks a host as non-overloaded. To make a host non-overloaded we sort the vmList at very first place in descending order of CPU utilizations by the VMs as shown in (11). We propose following two criteria for VM migration if a host is found to be overloaded: (i) Migrate a VM from sorted vmList existing at 0^{th} location until following occurs; (ii) If migrating next VM from sorted sequence makes host to be non-overloaded then identify a VM that causes smallest possible void between T_u and C_u as described in (11).

$$\text{vmList} = \left\{ \forall \alpha, \beta \in V_j \middle| C_u^{cpu}(\alpha) \geq C_u^{cpu}(\beta) \right\} \tag{11}$$

where α and β belongs to set of VMs V_j on overloaded host p_i, $VM_u^{cpu}(\cdot)$ provides absolute value of CPU utilization of corresponding VM. While selecting a VM for migration to bring down CPU utilization below utilization threshold, may create larger void between C_u and T_u and may generate unnecessary VM migrations, therefore, SVD selects such VM which creates smaller void between Cu and T_u as shown in Fig. 4(a) and Fig. 4(b).

$$v = \begin{cases} \phi & \text{if } h_j \notin \mathcal{H}_{\mathcal{O}} \\ VM_{i=0,j}^{cpu} & \text{if } h_j \in \mathcal{H}_{\mathcal{O}} \text{ and } C_u - ||C_{i=0,j}^{cpu}|| \geq T_u, \\ VM_{i=\tau,j} & \text{if } h_j \in \mathcal{H}_{\mathcal{O}} \text{ and } C_u - ||C_{\tau,j}^{cpu}|| < T_u. \end{cases} \tag{12}$$

where,

$$\begin{aligned} \mathcal{T} &= \min_{z \in VM} f(z) \\ &= min \left\{ \forall i \in VM \middle| voidList[i], \quad \text{where } 0 \leq i \leq |VM| \right\} \end{aligned}$$

Cite As:
VERMA, J., KUMAR, S., KAIWARTYA, O., CAO, Y., LLORET, J., KATTI, C. P., KHAREL, R., 2018.
Enabling green computing in cloud environments: Network virtualization approach toward 5G support.
Transactions on Emerging Telecommunications Technologies, pp. e3434.

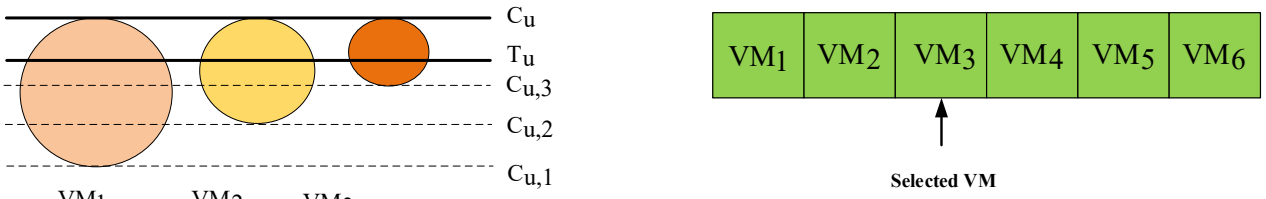
Algorithm 2: Smallest Void Detection Heuristic

Input: P_m–Power Model, $hostList$
Output: $migrationMap$

begin
  Initialize Simulation Parameters
  for $host$ ***in*** $hostList$ **do**
    $T_u \longleftarrow thresholdValue(type)$
    if $hostUtilization > T_u$ **then**
      $vmList \leftarrow getMigratableVms(host)$
      $vmList \leftarrow sortByCpuAllocation(vmList)$
        /* Descending order */
      if $hostUtilization - ||C^{cpu}_{i=0,j}|| \geq T_u$ **then**
        $vmToMigrate \leftarrow VM_{i=0,j}$
          /* Migrates largest VM by CPU allocation when its migration does not cause host utilization to fall below T_u */
      else if $hostUtilization - ||C^{cpu}_{i=0,j}|| < T_u$ **then**
        $vmToMigrate \leftarrow vmList.getIndex(Procedure\ smallestVoid(host))$
          /* Migrates a VM that creates smallest void between T_u and host utilization, when migration of VM at 0^{th} index of vmList cause utilization fall below T_u */
    else if $hostUtilization < T_u$ **then**
      $vmToMigrate \leftarrow \phi$
        /* No VM migraton due to empty set ϕ */
    $vmsToMigrate.add(vmToMigrate)$
  $migrationMap.add(getNewVmPlacement(vmsToMigrate))$
  $vmsToMigrate.clear()$
    /* Nullify the variable $vmsToMigrate$ after the loop */
  for $host$ ***in*** $hostList$ **do**
    if $isHostMinimumUtilized$ **then**
      $vmsToMigrate \leftarrow host.getVmList()$ /* All of the VMs migrates */
    $vmsToMigrate.add(vmToMigrate)$
  $migrationMap.add(getNewVmPlacement(vmsToMigrate))$
  return $migrationMap$

Procedure smallestVoid(host)

`smallestVoid(`*host*`)` **begin**
  $vmList \leftarrow getMigratableVms(host)$
  $voidLis[1 \ldots |vmList|]$
  for VM ***in*** $vmList$ **do**
    if *VM* ***is in*** *Migration* **then**
      $voidList[1 \ldots |vmList|] \leftarrow Double.MAX$
        /* Maximum value generated by *Double* wrapper class of Java */
    else if $hostUtlization - |VM| \geq T_u$ **then**
      $voidList[1 \ldots |vmList|] \leftarrow Double.MAX$
        /* Maximum value generated by *Double* wrapper class of Java */
    else if $hostUtlization - |VM| < T_u$ **then**
      $voidList[1 \ldots |vmList|] \leftarrow T_u - ||VM^{cpu}_{i,j}||$
        /* Size of *Void* generated by migration of $VM_{i,j}$ */
  $index \leftarrow findIndexOfSmallestValue(voidList1 \ldots |vmList|)$
  return $index$



(a) Possible Voids on Host due to VM migration

(b) Largest VM Selection with smallest Void (i.e. $T_u - VM_i$)

Fig. 4: Smallest Void Detection Based VM Migration

and,

$$voidList[i] = \begin{cases} \text{Double.MAX} & \text{if } \mathcal{B}(VM_{i,j}) = 1, \\ \text{Double.MAX} & \text{if } C_u - ||C_{i,j}^{cpu}|| \geq T_u, \\ T_u - ||VM_{i,j}^{cpu}|| & \text{if } C_u - ||C_{i,j}^{cpu}|| < T_u. \end{cases}$$

where $VM_u^{cpu}(\cdot)$ indicates v is VM selected for migration, $\mathcal{H}_{\mathcal{O}}$ be the set of overloaded PMs, $|VM| \in \mathbb{N}$, $|VM|$ is cardinality of set VM and $\mathbb{N}$ is a natural number, where $||C_{i,j}^{cpu}||$ is absolute value of CPU's Mips allocated to the i^{th} VM executing over j^{th} PM, $(T_u - ||C_{i,j}^{cpu}||)$ is size of void, Double.MAX is the maximum value generated from *Double* primitive wrapper class of Java, $\mathcal{B}(VM_{i,j}) = 1$ indicates i^{th} VM on j^{th} host is in migration. Stepwise illustration of SVD (Algorithm 2) is given below:

1) Step-1: Power model P_m and $hostList$ is supplied.
2) Step-2: Simulation parameters are initialized.
3) Step-3–14: for loop detects overloaded hosts and VMs are arranged in descending order of CPU utilization rates. VMs are set for migration from the overloaded hosts according to the criteria shown in (12). If migration of the VM available at 0^{th} index of the list does not causes to fall utilization level of the host below the utilization threshold of the host then VM indexed at 0^{th} location of the list will set for migration. Otherwise, a VM whose migration will leave minimum void between utilization threshold of the host and utilization level of the host on removing that VM will set for migration. For this purpose an auxiliary list of void is created in the following manner and index of minimum cell value will be chosen to identify the index of VM to be migrated as summarized below and shown in Procedure "smallestVoid":
   a) If corresponding VM is under migration then set cell value as the maximum value provided by "Double" wrapper class.
   b) Subtracting CPU utilization rates of corresponding VM from current utilization level of host becomes value greater than equal to utilization threshold (will never occur) then set the cell value as the maximum value provided by "Double" wrapper class.
   c) Otherwise set value of difference of current CPU utilization host and CPU utilization rates of corresponding VM.

   Furthermore, the host is evaluated for overloading and step continues until the host will become non-overloaded. It continues until all the hosts are processed.
4) Step-15–16: All the migrated VMs are optimally packed/placed on active physical machines according to the constraint $\frac{11}{9}OPT$. After allocation of VMs flush the memory variable containing list of those VMs which are selected for migration.
5) Step-17–21: Hosts are detected for undertilized host. Minimum utilized host among all hosts is declared underutilized host and all VMs from this host are migrated to pack over other active hosts.

3.4.3 Maximum Fill Technique

This technique is also centered around host overloading detection criteria. We went one step ahead in this method to consolidate VMs more tightly on hosts running beyond host utilization thresholds. Maximum Fill Technique (MFT) aims upon migration of VMs from overloaded host in such a way that leads to packing of a host with high numbers of VMs to pack a host tightly while host is in active mode. Let us assume that a random host has n VMs executing over it then current host utilization is ratio of sum of MIPS allocated to the VM that is $\sum_{i=1}^{n} C_{i,j}^{cpu}$ and total MIPS capacity of the host h_j. Let V_j be a set of VMs currently allocated to the host j, MFT finds a VM v that satisfies the condition given in (13) and depicted in Fig. 5(a)–5(b).

$$\text{vmList} = \left\{ \forall \alpha, \beta \in V_j, P_j \middle| C_u^{cpu}(\alpha) \leq C_u^{cpu}(\beta) \right\} \tag{13}$$

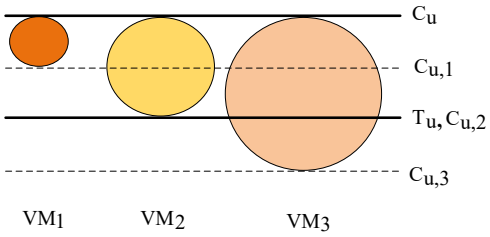
where α and β are the VMs from VM pool available on j^{th} overloaded host P_j, $VM_u^{cpu}(\cdot)$ is the CPU utilization. In the sorted sequence of VM, MFT selects a suitable VM that either causes minimum increase in CPU allocation to the VM.

Cite As:
VERMA, J., KUMAR, S., KAIWARTYA, O., CAO, Y., LLORET, J., KATTI, C. P., KHAREL, R., 2018.
Enabling green computing in cloud environments: Network virtualization approach toward 5G support.
Transactions on Emerging Telecommunications Technologies, pp. e3434.

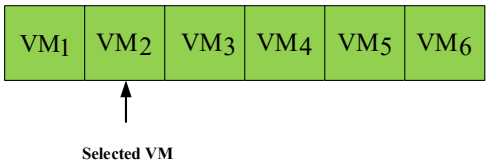
Algorithm 3: Maximum Fill Technique Heuristic

Input: P_m–Power Model, $hostList$
Output: $MigrationMap$

```
1  begin
2      Initialize Simulation Parameters
3      for host in hostList do
4          T_u ⟵ thresholdValue(type)
5          if hostUtilization > T_u then
6              vmList ← getMigratableVms(host)
7              vmList ← reverseSortByCpuAllocation(vmList)
                                                    /* Ascending order */
8              n_max ← |vmList| − 1
9              if (C_u − T_u > VM) ∧ (n = n_max) then
10                 vmToMigrate ← VM_{i=n,j}
                       /* If all the VMs on overloaded host are smaller in CPU allocation from
                   (C_u − T_u) while C_u > T_u */
11             else if (C_u − T_u ≤ VM) ∧ (0 ≤ i < n_max) then
12                 vmToMigrate ← VM_{i,j}
                       /* If one or more VMs on overloaded host are large in CPU allocation from
                   (C_u − T_u) while C_u > T_u */
13             else if (C_u − T_u ≤ VM) ∧ (i < n_max) then
14                 vmToMigrate ← φ
                                                    /* No VM migraton due to empty set φ */
15         vmsToMigrate.add(vmToMigrate)
16     migrationMap.add(getNewVmPlacement(vmsToMigrate))
17     vmsToMigrate.clear()
                                    /* Nullify the variable vmsToMigrate after the loop */
18     for host in hostList do
19         if isHostMinimumUtilized then
20             vmsToMigrate ← host.getVmList()          /* All of the VMs migrates */
21         vmsToMigrate.add(vmsToMigrate)
22     migrationMap.add(getNewVmPlacement(vmsToMigrate))
23     return migrationMap
```



(a) Possible Voids on Host due to VM migration



(b) Largest VM Selection with MFT criteria

Fig. 5: Maximum Fill Technique Based VM Migration

Assuming $Void = (T_u - ||C_{i,j}^{cpu}||)$, a host will run below its efficiency as much as large the void size. Meanwhile, migrating the largest possible VM in vmList from overloaded host may create a large size void. Moreover, SVD does not provide a suitable criteria to maximize the server utilization up to its capacity. A server is marked as overloaded if $C_u > T_u$ holds true. To cope up with void size problem, we scan the vmList obtained from (13) and identifies a VM that fulfill following condition as shown in (14) in consonance with (15).

$$C_u - T_u < VM_{index,j}^{cpu} \text{ where } 0 \leq index \leq n_{max} \tag{14}$$

$$v = \begin{cases} \emptyset & \text{if } C_u - T_u > C_{i,j}^{cpu} \text{ and } i < n_{max} \\ V_{i=n,j} & \text{if } C_u - T_u > C_{i,j}^{cpu} \text{ and } n = n_{max} \\ VM_{i,j} & \text{if } C_u - T_u \leq C_{i,j}^{cpu} \text{ and } 0 \leq i < n_{max} \end{cases} \tag{15}$$

where $n_{max} = (|VM| - 1)$ and $|VM|$ is the maximum possible size of vmList on overloaded host, "v" is the VM selected for migration on applying MFT criteria on overloaded host. $VM_{i,j}$ represents a VM i executing over j^{th} host. $C_{u,j}$ is current utilization of j^{th} host and $T_{u,j}$ utilization threshold of j^{th} host. Stepwise illustration of MFT (Algorithm 3) is given below:

1) Step-1: Power model P_m and $hostList$ is supplied.
2) Step-2: Simulation parameters are initialized.
3) Step-3–15: for loop detects overloaded hosts and VMs are arranged in descending order of CPU utilization rates. VMs are set for migration from the overloaded hosts according to the criteria shown in (15) and summarized as given below:
   a) No migration if CPU utilization rates of VM smaller than gap between current utilization (C_u) of host and utilization threshold (T_u) of index of VM is other than maximum index of vmList (n_{max}).
   b) Migrate VM at highest (n_{max}) index of vmList CPU utilization rates of VM smaller than gap between current utilization (C_u) of host and utilization threshold (T_u).
   c) Migrate a largest VM at index other than (n_{max}) whose CPU utilization rates are greater than gap between current host utilization (C_u) and host utilization threshold (T_u).

   Furthermore, the host is evaluated for overloading and step continues until the host will become non-overloaded. It continues until all the hosts are processed.
4) Step-16–17: All the migrated VMs are optimally packed/placed on active physical machines according to the constraint $\frac{11}{9}OPT$. After allocation of VMs flush the memory variable containing list of those VMs which are selected for migration.
5) Step-18–22: Hosts are detected for undertilized host. Minimum utilized host among all hosts is declared underutilized host and all VMs from this host are migrated to pack over other active hosts.

4 Performance Evaluation

In this section, we conduct performance evaluation of the proposed heuristic models using simulations along with comparative study of benchmark algorithms. The metrics like energy consumption, host shutdowns, total VM migrations are considered for the expalantion.

4.1 Simulation Setup

We targeted IaaS deployment model of cloud computing for our experimentations. An IaaS deployment model of Cloud is supposed to create a view of infinite computing resources to the client side. It is tough to conduct experiments on large scale real infrastructure repetitively for the proposed heuristics. Therefore, we find simulations as the best alternative for performance evaluation. CloudSim is chosen as a testbed for performing simulations on proposed heuristics [47]. It is a Java based advanced simulator where all the requirements for realization of cloud computing environment are already present. For experimentation purpose, one has to make certain settings according to the necessity of the experiment. The simulation parameters taken into consideration for our experimentations are given in Table 2. The underlying hardware architecture used for experimental analysis is as follows: Intel(R) Core(TM) *i*7-3770 CPU @3.40 GHz (8 MB SmartCache) with 4 cores having the capability to issue 8 hyper threads, 10 GB RAM DDR3 1330 with systems architecture of 64 bits, 1 TeraByte secondary storage and Windows 10 Pro N operating system. We have considered simulation architecture as shown in Fig. 6 with assembly of overloaded, normal loaded and underloaded hosts as container of virtual servers/machines that are made available for execution of impending jobs arriving from dynamic workload traces.

We have simulated a data center with 1200 heterogeneous physical machines among which 600 of them are HP ProLiant ML110 G4 servers, and rest of the 600 are HP ProLiant ML110 G5 servers. The characteristics of the servers and data on their power consumption are given in Table 1. We take Amazon EC2 instance type [48] VMs having characteristics as shown in Table 3. The VMs simulated are of single core type due to the reason that workload data taken from traces comes from single core physical machines. At the initial stage, the VMs are allocated according to the resource requirements as

Cite As:
VERMA, J., KUMAR, S., KAIWARTYA, O., CAO, Y., LLORET, J., KATTI, C. P., KHAREL, R., 2018.
Enabling green computing in cloud environments: Network virtualization approach toward 5G support.
Transactions on Emerging Telecommunications Technologies, pp. e3434.

TABLE 2: Simulation Parameters

		Values/Type					
	Parameters	**NPA**	**THR**	**MAD**	**IQR**	**LR**	
1.	Total Hosts	1200	1200	1200	1200	1200	
2.	Safety Parameter	None	None	1.5	2.5	1.2	
3.	Threshold	None	0.8	$1 - s \times MAD$	$1 - s \times IQR$	$1 - s \times LR$	
4.	PE/Cloudlet	1	1	1	1	1	
5.	Scheduling Interval	300	300	300	300	300	
6.	Simulation Limit	24 Hours	24 ours	24 Hours	24 Hours	24 Hours	
7.	PlanetLab Workload	03/03/2011 06/03/2011 09/03/2011 22/03/2011 25/03/2011 03/04/2011 09/04/2011 11/04/2011 12/04/2011 20/04/2011	03/03/2011 06/03/2011 09/03/2011 22/03/2011 25/03/2011 03/04/2011 09/04/2011 11/04/2011 12/04/2011 20/04/2011	03/03/2011 06/03/2011 09/03/2011 22/03/2011 25/03/2011 03/04/2011 09/04/2011 11/04/2011 12/04/2011 20/04/2011	03/03/2011 06/03/2011 09/03/2011 22/03/2011 25/03/2011 03/04/2011 09/04/2011 11/04/2011 12/04/2011 20/04/2011	03/03/2011 06/03/2011 09/03/2011 22/03/2011 25/03/2011 03/04/2011 09/04/2011 11/04/2011 12/04/2011 20/04/2011	Real traces of PlanetLab server workload by Dates
8.	Random Workload	1052 898 1061 1516 1078 1463 1358 1233 1054 1033	1052 898 1061 1516 1078 1463 1358 1233 1054 1033	1052 898 1061 1516 1078 1463 1358 1233 1054 1033	1052 898 1061 1516 1078 1463 1358 1233 1054 1033	1052 898 1061 1516 1078 1463 1358 1233 1054 1033	Total VMs Simulated for synthetic workload

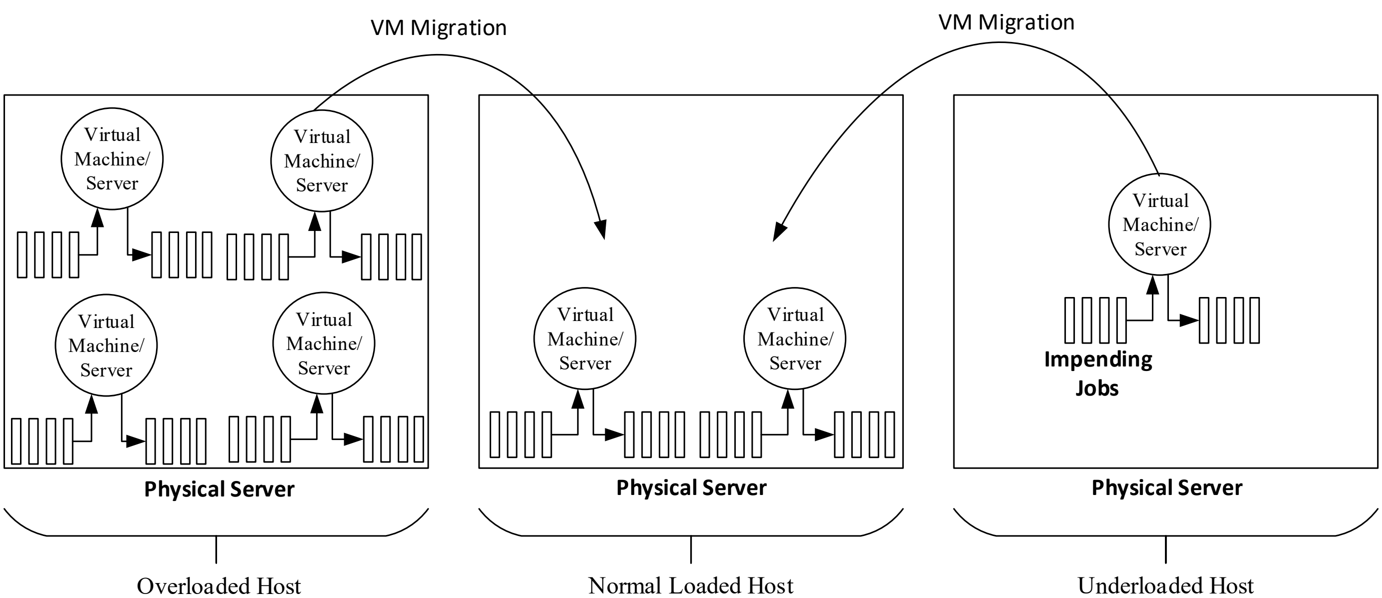


Fig. 6: Simulation Architecture

TABLE 3: Characteristics of Amazon EC2 VM instances

Resource Types	**High-CPU Medium Instance** ($t_1.medium$)	**Extra Large Instance** ($t_1.xlarge$)	**Small Instance** ($t_1.small$)	**Micro Instance** ($t_1.micro$)
MIPS	2500	2000	1000	500
RAM	0.85 GB	3.75 GB	1.7 GB	613 MB

Cite As:
VERMA, J., KUMAR, S., KAIWARTYA, O., CAO, Y., LLORET, J., KATTI, C. P., KHAREL, R., 2018.
Enabling green computing in cloud environments: Network virtualization approach toward 5G support.
Transactions on Emerging Telecommunications Technologies, pp. e3434.

per the VM types. However, VMs suffer from resources underutilization during the simulations according to workload provisioning and create opportunities for dynamic consolidation.

To evaluate the proposed methods we adopted simulation based validation that rely upon real workload traces from real systems that are available publicly. We used data provided from CoMon project for monitoring infrastructure of PlanetLab servers [20] that was included in CloudSim simulation toolkit for 10 random days (during March and April 2011). The workload traces contains data on CPU utilizations with interval of measurement of 5 minutes from more than a thousand VMs hosted on more than 1000 server located at 645 places of the world [13]. The workload traces collected are representative of IaaS type clouds similar to Amazon EC2 where independent users create and manage VMs of single-core. As the utilization value of some VMs in the data set is so low, we filter the original data and only consider the range of CPU and memory utilization between 5% to 90%. So we can evaluate the proposed VM consolidation by considering both CPU and memory intensive tasks. Apart from this, we utilize synthetic workload data (provided by CloudSim class) to test out the performance of proposed algorithms with dynamic and non-stationary random workloads. The random workload generates cloudlets as a small task that takes simulation time as a seed value. In Table 4, P-i's are PlanetLab workload traces and R-i's are synthetic random workload traces coming out of the similar number of VMs as in PlanetLab traces.

TABLE 4: PlanetLab Workload Data Traces

PlanetLab Workload Instance	Date	Number of VMs	Mean	St. Dev.	Quartile 1 (%)	Median (%)	Quartile 3 (%)	Random Workload Instance
PL-1	03/03/2011	1052	12.31	17.09	2	6	15	R-1
PL-2	06/03/2011	898	11.44	16.83	2	5	13	R-2
PL-3	09/03/2011	1061	10.70	15.57	2	4	13	R-3
PL-4	22/03/2011	1516	9.26	12.78	2	5	12	R-4
PL-5	25/03/2011	1078	10.56	14.14	2	6	14	R-5
PL-6	03/04/2011	1463	12.39	16.55	2	6	17	R-6
PL-7	09/04/2011	1358	11.12	15.09	2	6	15	R-7
PL-8	11/04/2011	1233	11.56	15.07	2	6	16	R-8
PL-9	12/04/2011	1054	11.54	15.15	2	6	16	R-9
PL-10	20/04/2011	1033	10.43	15.21	2	4	12	R-10

4.2 Baseline Methods

We discuss the utility of employing a utilization prediction model by comparing proposed heuristics with baseline heuristics proposed in [33] and [15]. Two types of methods are used for comparative evaluation of proposed models: (i) Host overloading detection methods, and (ii) Host offloading methods. Host overloading methods consider two approaches, namely static thresholds (THR) and dynamic thresholds. The first one assumes static values of host utilization thresholds while later one calculate dynamic host utilization threshold values on runtime. Dynamic methods like Median Absolute Deviation (MAD), Interquartile Range (IQR) and Local Regression (LR) for host overloading detection thresholds [15] that are shown in general equation (16).

$$T_u = \begin{cases} 1 - s \cdot X & \text{if } X \in \{IQR, MAD, LR\} \\ THR, & \text{otherwise.} \end{cases} \tag{16}$$

Host offloading methods consider three benchmark methods namely, (i) Random Selection (RCS) policy that selects a VM for migration according to uniformly distributed discrete random variable $X \overset{d}{=} U((0, |V_j|))$, whose values index a set of VMs V_j allocated to a host j. (ii) Minimum Migration Time (MMT) policy that selects a VM v for migration that takes minimum time to migrate a VM allocated the overloaded host. The migration time is calculated as the ratio of amount of RAM utilized by the VM to the spare network bandwidth available for the host j as formalized as $\frac{RAM_u(v)}{NET_j}$, and (iii) Maximum Correlation (MC) selects a VM for migration from overloaded host having highest multiple correlation coefficient that is squared coefficient of correlation between observed value y and predicted value $\hat{y}$ for the random variable Y on CPU utilization by that VM [15] [49] [50]. On considering n VMs to be hosted by the physical host, the multiple correlation coefficient is calculated as shown in (17).

$$R^2_{Y,X_1,X_2,\cdots,X_{n-1}} = \frac{\sum_{i=1}^{n}(y_i - \mu_Y)^2(\hat{y}_i - \mu_{\hat{Y}})^2}{\sum_{i=1}^{n}(y_i - \mu_Y)^2 \sum_{i=1}^{n}(\hat{y}_i - \mu_{\hat{Y}})^2} \tag{17}$$

where $X_1, X_2, \cdots X_{(n-1)}$ are the random variable on CPU utilizations by those $(n-1)$ VMs that are not eligible to migrate in accordance to the MC policy, y_i and $\hat{y}$ are observed and predicted values of dependent variable Y, μ_Y and $\mu_{\hat{Y}}$ are the sample means of Y and $\hat{Y}$, respectively.

TABLE 5: Energy Consumption Metric

		NPA	DVFS	THR						MAD						IQR						LR					
				RCS	MMT	MC	MeMT	SVD	MFT	RCS	MMT	MC	MeMT	SVD	MFT	RCS	MMT	MC	MeMT	SVD	MFT	RCS	MMT	MC	MeMT	SVD	MFT
1(a).	PL_20110303	3616.20	803.91	184.27	193.58	183.37	179.02	165.54	183.87	175.89	185.32	175.23	170.62	157.94	149.96	180.38	190.16	180.19	174.63	161.65	153.40	151.05	163.06	150.14	149.27	140.75	139.63
(b).	Random_1052	3616.20	1051.36	819.90	840.07	820.34	797.68	747.96	747.96	888.06	905.80	887.49	860.84	810.14	811.03	931.81	947.39	932.89	174.63	856.24	854.44	664.77	689.16	665.36	659.16	650.60	650.60
2(a).	PL_20110306	3616.20	623.77	139.90	146.11	138.64	135.48	124.55	118.48	133.67	141.65	133.45	130.13	118.22	112.77	136.12	144.00	136.10	133.40	121.68	115.30	115.05	124.87	114.79	113.65	107.03	105.49
(b).	Random_898	3616.20	899.01	702.55	717.72	701.24	680.06	639.29	639.29	759.22	776.26	758.97	735.44	693.54	692.13	795.66	812.01	796.07	133.40	731.43	732.29	569.01	589.46	570.00	564.04	555.47	555.47
3(a).	PL_20110309	3616.20	708.68	159.32	166.92	157.98	152.86	140.48	133.40	156.66	162.84	153.66	150.60	135.97	130.59	159.19	168.17	157.90	154.55	140.53	134.22	130.27	143.42	130.30	128.88	118.76	117.47
(b).	Random_1061	3616.20	1061.72	830.03	848.00	829.63	804.86	757.65	757.65	895.25	914.58	897.29	868.38	818.64	818.09	941.05	957.79	941.12	154.55	866.15	864.16	671.27	694.74	672.26	667.99	657.53	657.53
4(a).	PL_20110322	3616.20	1014.21	194.52	204.13	195.61	186.77	171.70	166.33	189.67	198.11	188.72	181.55	166.61	161.07	194.43	203.25	193.23	185.43	170.34	166.26	162.17	180.32	163.82	156.59	147.62	146.35
(b).	Random_1516	3616.20	1513.80	1179.47	1208.19	1177.94	1148.47	1079.83	1079.83	1269.93	1285.05	1269.52	1233.49	1168.57	1166.55	1302.57	1303.26	1303.31	185.43	1229.90	1228.24	953.67	991.09	956.45	947.42	936.29	936.22
5(a).	PL_20110325	3616.20	785.49	169.86	176.31	167.83	161.59	149.06	142.69	163.55	172.39	162.37	156.70	143.79	137.13	166.06	174.81	166.52	159.73	147.40	140.71	139.75	152.79	139.39	136.92	128.16	126.00
(b).	Random_1078	3616.20	1077.17	842.35	861.98	841.15	819.98	768.67	768.67	909.12	929.04	910.03	880.65	830.16	831.15	954.63	975.35	954.77	159.73	878.79	878.21	681.13	706.47	681.77	676.95	668.17	668.17
6(a).	PL_20110403	3616.20	1071.90	251.93	260.90	249.50	242.10	223.54	217.11	241.58	254.33	240.35	231.86	214.50	207.60	247.05	260.08	246.93	237.52	220.15	213.68	206.87	222.65	206.70	201.96	193.32	191.60
(b).	Random_1463	3616.20	1460.71	1139.44	1166.53	1138.37	1108.96	1041.62	1041.62	1231.51	1252.74	1229.72	1193.03	1126.63	1128.32	1279.14	1284.98	1279.40	237.52	1187.63	1188.74	922.58	956.46	922.82	916.69	905.10	905.10
7(a).	PL_20110409	3616.20	928.59	201.30	210.53	198.62	191.87	177.49	172.53	193.02	204.69	191.78	185.57	171.10	165.99	197.72	209.82	196.27	189.89	176.04	170.66	165.74	179.80	165.30	160.99	152.12	151.38
(b).	Random_1358	3616.20	1356.55	1059.90	1082.80	1056.35	1028.04	968.96	968.96	1141.14	1167.92	1143.86	1108.66	1045.51	1045.33	1199.48	1219.46	1200.69	189.89	1102.30	1102.48	857.44	887.23	858.38	850.83	840.15	840.15
8(a).	PL_20110411	3616.20	903.08	197.19	207.06	195.66	188.97	174.87	169.81	190.87	200.52	189.25	182.32	168.81	163.50	194.60	205.76	194.47	186.86	174.17	167.68	159.95	177.84	162.04	156.43	149.21	148.46
(b).	Random_1233	3616.20	1234.25	963.17	985.08	959.42	935.66	879.81	879.81	1038.80	1062.15	1040.57	1009.68	952.26	952.03	1091.71	1112.59	1091.15	186.86	1005.19	1003.30	780.11	806.54	780.11	773.95	764.06	764.11
9(a).	PL_20110412	3616.20	766.75	171.55	178.33	169.68	163.68	152.44	146.46	165.43	173.72	165.41	158.86	146.38	141.92	169.87	178.32	169.50	164.18	150.99	146.77	140.18	152.91	140.13	135.72	129.93	128.10
(b).	Random_1054	3616.20	1054.07	823.02	842.46	823.00	801.11	751.08	751.08	889.22	909.05	889.55	860.38	813.38	812.28	933.38	952.06	934.00	164.18	858.23	857.50	666.50	689.45	667.46	662.06	652.94	652.94
10(a).	PL_20110420	3616.20	1014.21	194.52	204.13	195.61	186.77	171.70	166.33	189.67	198.11	188.72	181.55	166.61	161.07	194.43	203.25	193.23	185.43	170.34	166.26	162.17	180.32	163.82	156.59	147.62	146.35
(b).	Random_1033	3616.20	1513.80	1179.47	1208.19	1177.94	1148.47	1079.83	1079.83	1269.93	1285.05	1269.52	1233.49	1168.57	1166.55	1302.57	1303.26	1303.31	185.43	1229.90	1228.24	953.67	991.09	956.45	947.42	936.29	936.22

TABLE 6: Total VM Migrations

		NPA	DVFS	THR						MAD						IQR						LR					
				RCS	MMT	MC	MeMT	SVD	MFT	RCS	MMT	MC	MeMT	SVD	MFT	RCS	MMT	MC	MeMT	SVD	MFT	RCS	MMT	MC	MeMT	SVD	MFT
1(a).	PL_20110303	0	0	24324	27041	23997	23096	20751	24373	23792	26699	23876	22454	20814	17869	23763	26451	23589	22825	20683	18731	23271	28336	22848	22110	17461	14493
(b).	Random_1052	0	0	83942	93111	83528	76452	63468	63468	92596	101162	92273	84946	71848	71882	98686	106782	98466	22825	77266	77235	63064	82072	63439	57937	47125	47125
2(a).	PL_20110306	0	0	18990	20754	18682	18187	16112	13901	19026	20767	18587	18047	15939	14021	18487	20765	18696	18031	16303	14063	17584	21057	17218	17196	13052	10656
(b).	Random_898	0	0	71973	79580	71668	65331	54350	54350	79183	86785	79042	72382	61869	61751	84449	91684	84394	18031	66276	66568	53904	69569	54709	49270	39884	39884
3(a).	PL_20110309	0	0	21318	23855	21322	19480	17193	15478	21130	23174	20924	19517	17632	15803	21124	23856	20914	20008	17709	16203	19698	24898	20011	19647	14661	12096
(b).	Random_1061	0	0	84844	93704	84579	77226	64099	64099	92991	102111	93243	85450	72417	72613	99735	107945	99319	20008	78107	78155	63374	81392	64675	59673	47443	47443
4(a).	PL_20110322	0	0	26453	29435	26704	24550	22269	20052	25843	28731	25305	24183	22256	20994	26446	29226	26067	24340	22119	21069	27325	32494	27682	25802	20252	18249
(b).	Random_1516	0	0	120177	133432	119667	109624	91040	91040	132330	146227	132493	120718	102034	101731	143481	157174	142906	24340	109582	109593	89035	119023	91972	83652	67931	67586
5(a).	PL_20110325	0	0	22755	25093	22860	21107	18921	17376	21823	24758	21795	20812	18587	17336	22019	24802	22428	20377	19050	17496	21904	26369	21826	21656	17370	13803
(b).	Random_1078	0	0	85977	95579	86014	78686	65225	65225	94538	103797	94910	86525	73691	73827	101195	110156	100583	20377	79043	79459	64405	83646	64558	59787	48234	48234
6(a).	PL_20110403	0	0	32808	35732	32354	30792	27258	24634	32078	35524	31576	29465	27045	24385	31744	35542	31834	29594	26833	25462	32951	38465	32894	30940	24945	21315
(b).	Random_1463	0	0	116103	128794	115749	105572	87603	87603	128455	140360	127694	116446	98865	98922	136541	148658	136013	29594	106496	106552	86825	113292	88034	81475	65542	65542
7(a).	PL_20110409	0	0	26940	29806	26314	24914	22415	20584	25682	29140	25546	24460	21989	20268	25940	29349	25386	23946	22713	21075	26906	31556	27282	24536	20187	17187
(b).	Random_1358	0	0	108297	119646	107607	98131	81518	81518	118550	130508	118611	108655	91785	91905	126657	137484	126384	23946	98768	98660	81496	104306	82089	74613	61118	61118
8(a).	PL_20110411	0	0	26374	28959	25793	24413	22174	20406	25294	28335	25197	23967	22025	20374	25741	28824	25412	24200	22207	21137	26541	31047	26587	24761	20493	17780
(b).	Random_1233	0	0	98284	108773	97748	89467	74404	74404	108224	118486	108167	98949	83736	83750	115412	125056	114842	24200	90217	90187	73898	94192	74042	68884	54874	55024
9(a).	PL_20110412	0	0	22761	25101	22760	21406	18904	17706	22003	24241	21634	20417	19214	17992	22408	24881	22093	21186	19227	18604	22021	26482	22702	20644	16911	14363
(b).	Random_1054	0	0	84110	93091	83933	76867	63862	63862	92668	101417	92517	84675	72111	72108	98924	107164	98473	21186	77809	77304	63404	80382	63742	58257	47335	47335
10(a).	PL_20110420	0	0	26453	29435	26704	24550	22269	20052	25843	28731	25305	24183	22256	20994	26446	29226	26067	24340	22119	21069	27325	32494	27682	25802	20252	18249
(b).	Random_1033	0	0	120177	133432	119667	109624	91040	91040	132330	146227	132493	120718	102034	101731	143481	157174	142906	24340	109582	109593	89035	119023	91972	83652	67931	67586

Cite As:
VERMA, J., KUMAR, S., KAIWARTYA, O., CAO, Y., LLORET, J., KATTI, C. P., KHAREL, R., 2018.
Enabling green computing in cloud environments: Network virtualization approach toward 5G support.
Transactions on Emerging Telecommunications Technologies, pp. e3434.

4.3 Performance Evaluation Metrics

The main goals of the proposed work includes: (i) to reduce the energy consumption by servers deployed in cloud data center; (ii) to minimize the excessive number of shutdowns of physical servers (cycle of host deactivation and reactivation); and (iii) to minimize the VM migrations. We evaluate the performance of the proposed models consider following performance metrics.

- ***Energy Consumption:*** Energy consumption of a physical machine has two components, namely, (i) Static component ($E_{sleep} = P_{sleep}t + E_{transition}$) when hosts is in sleep/standby mode that includes the period he transition period when the host moves from sleep state to wake-up mode, and (ii) Dynamic component ($E_{nosleep} = P_0 t + (cV^2 f)t$) when host is in active mode that includes the period when the host is either performing computation or host is in idle mode. Meanwhile, energy consumption is also written as a function of host utilization as shown in (18).

$$E(t) = \int_{t=0}^{T} P(u(t))dt \tag{18}$$

  E_{sleep} and P_{sleep} are the energy and power consumption when host is in sleep mode, $E_{nosleep}$ and $P_{nosleep}$ are energy and power consumption when host is in active mode, $E_{transition}$ is the energy consumption when host trigger from sleep state to idle/active state, P_0 is idle power consumption, c is constant, V is the voltage and f is CPU frequency, and t is time.
- ***Total Host Shutdowns:*** Excessive host shutdowns (frequent cycle of host deactivation and reactivation) causes extra penalty for invoking the servers back due to sudden upsurge in demand and leads to non-fulfillment of SLAs due to delay in fulfilling resource demanded by application services. The proposed approaches turn off the hosts that are found to be underutilized or ideal to cut down the energy consumption. However, excessive shutdowns cause extra efforts and delay to revoke them when need arises.
- ***VM Migration:*** Live migration of VMs is a costly operation that causes involvement of (i) some amount of CPU processing in the part of source server, (ii) the usage of bandwidth link between the source and destination servers, (iii) the downtime of the application services executing over migrating VM and (iv) total migration time of the VM [51]. In addition, VM migration consumes negligible energy that is considered for measuring SLA violations as 10% of CPU utilization during all VM migrations in the servers.
- ***SLA Violations:*** SLA violations (SLAV) metric was proposed in [15] to measure the SLAs delivered to VMs in IaaS clouds. SLAV is a composite metric that capture the SLA violations due to over utilization (SLAVO) and SLA violation due the VM migrations (SLAVM) together. Measurement of SLAV, SLAVO, and SLAVM are performed using equations shown in (19), (20) and (21).

$$SLAV = SLAVO \times SLAVM \tag{19}$$

$$SLAVO = \alpha \frac{T_v}{T_a} = \frac{1}{M} \sum_{i=1}^{M} \frac{T_{s_i}}{T_{a_i}} \tag{20}$$

$$SLAVM = \beta \frac{C_v}{C_a} = \frac{1}{N} \sum_{j=1}^{N} \frac{C_{d_j}}{C_{r_j}} \tag{21}$$

  where α and β are the scaling constant, M is the total number of physical hosts, N is the total number of VMs, T_{a_i} is the total time during which host i experienced CPU utilizations 100% leading to SLA violations. T_{a_i} is the total time during which host i remained active, C_{d_j} is estimate of performance degradation for VM j and $C_{r,j}$ is total CPU capacity requested by VM j during VMs lifetime.
- ***ESV:*** It is a composite metric $ESV(t)$ proposed in [33] that attempts to capture the effect of energy consumption and SLAV at once as shown in (22).

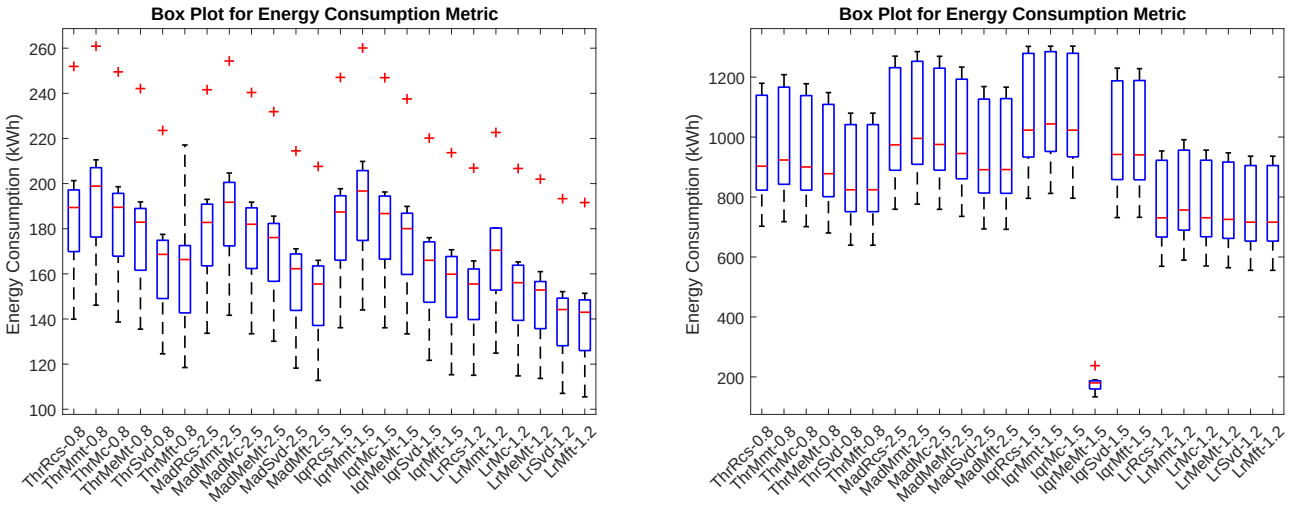
$$ESV = E(t) \times SLAV \tag{22}$$

4.4 Analysis of Results

We evaluate proposed methods on considered simulation framework considering ten workload traces of PlanetLab Servers and ten synthetic random workload traces. We get the results of Energy consumption metric and Total VM migrations metric as shown in Table 5 and Table 6. Similarly, we have collected the results of rest of the performance metrics and plot the results of performance metrics in Fig. 7–13.

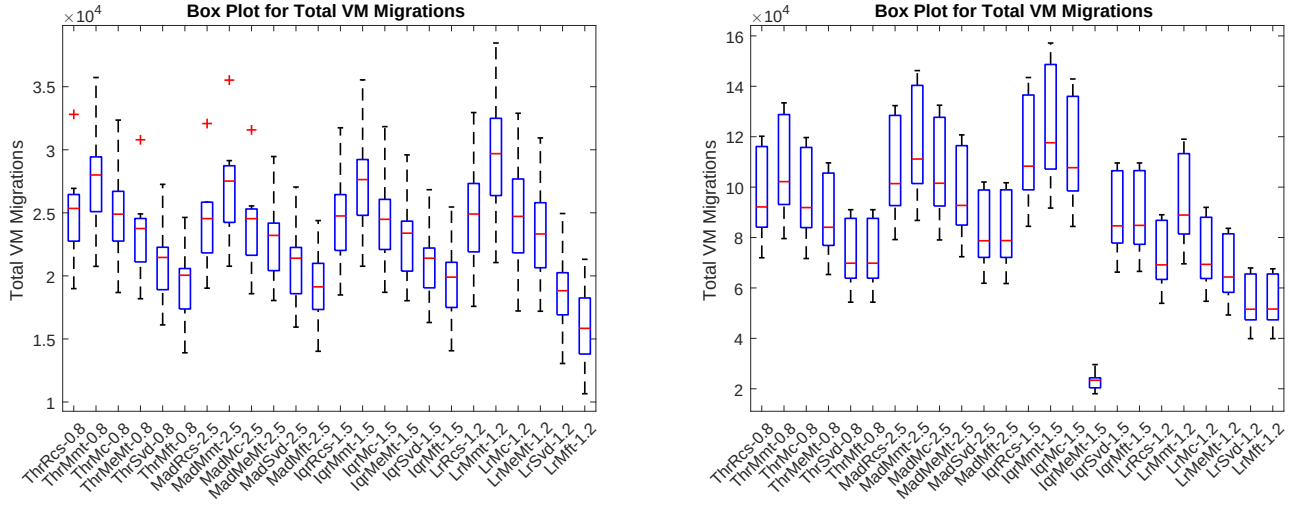
On simulating benchmark algorithms and proposed heuristic algorithms, we have found that proposed approaches outperforms over the benchmark methods on energy consumption metric for both of the workload traces as indicated in results of Table 5 and Fig. 7(a)–7(b). The efficiency of the algorithm is attributed to efficient packing of VMs on servers that utilize the server resources for long span of time and improve resource utilizations which do not generates unnecessary VM migrations.

Cite As:
VERMA, J., KUMAR, S., KAIWARTYA, O., CAO, Y., LLORET, J., KATTI, C. P., KHAREL, R., 2018.
Enabling green computing in cloud environments: Network virtualization approach toward 5G support.
Transactions on Emerging Telecommunications Technologies, pp. e3434.



(a) PlanetLab Workload (b) Random Workload

Fig. 7: Energy consumption metric on workload traces of Planetlab servers and random synthetic workload



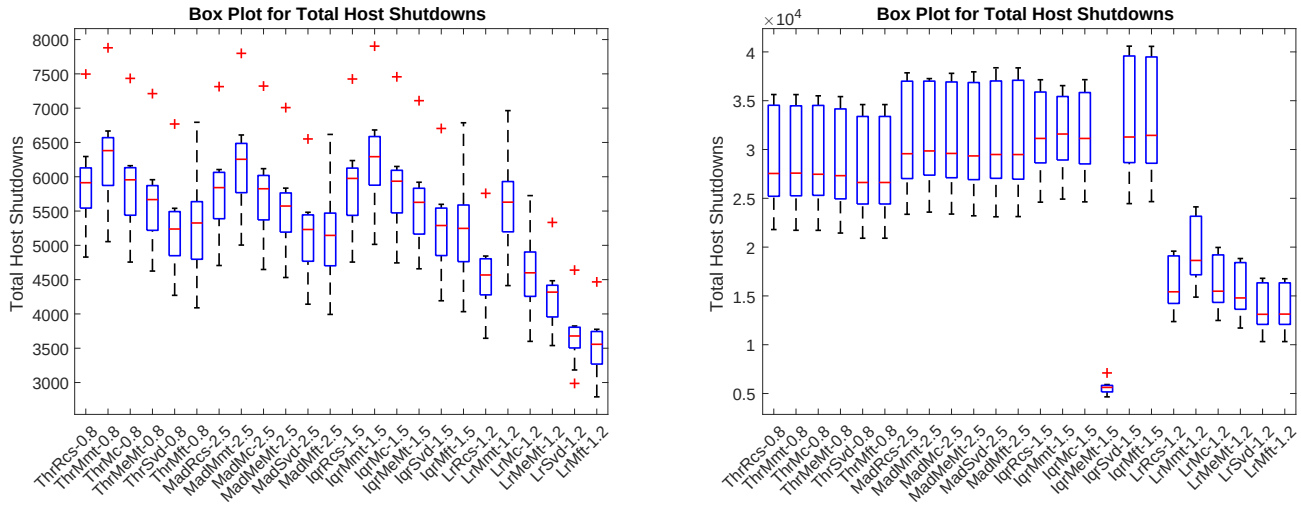
(a) PlanetLab Workload (b) Random Workload

Fig. 8: Total VM migrations metric on workload traces of Planetlab servers and random synthetic workload

Total VM migrations is an important parameter for real environment of cloud computing technology which involves a huge bandwidth cost for live migrations of VMs. Results of Table 6 and Fig. 8(a)–8(b) show that proposed algorithms report low VM migrations in comparison to benchmark algorithms on both of the PlantLab and random workloads. However, MeMT reports lowest VM migration rate with IQR threshold on employing random workload. Meanwhile, MeMT with combination of IQR performs extraordinarily over all possible combinations on employing random traces of synthetic workload due to detection of hotspots (overloaded hosts) easily. For rest of the cases, MFT model with LR overloading detection method reports lowest value of total VM migration metric. The success of the proposed models is attributed to controlled VM migration through VM selection methods with identified safety parameter for the overloading detection method.

Results of Fig. 9(a)–9(b) show that the proposed models are not only able to reduce the energy consumption and total VM migrations but also reduce the total host shutdowns. The reason of lower host shutdowns is attributed to efficient packing of VMs on fewer hosts that do not violate the utilization thresholds of the servers and may not require further VM migration until the applications hosted in VMs finish their jobs. MeMT reports lowest host shutdowns along with lowest energy consumption due to ability of spotting of hotspots associated with MeMT as shown in Fig. 9(a)–9(b). Similarly, we collected the results on SLA violations, SLAVO, SLAVM and ESV metric whose results are plotted in Fig. 10(a)–10(b), Fig 11(a)–11(b), Fig. 12(a)–12(b), and Fig. 13(a)–13(b). The proposed methods, namely, MeMT, SVD and MFT yields best results with combinations of static and dynamic thresholds in terms of energy consumption, total host shutdowns and the total VM migrations due to efficient VM packing under the constraint of $\frac{11}{9}OPT + 1$ VMs over a host [28] in comparison

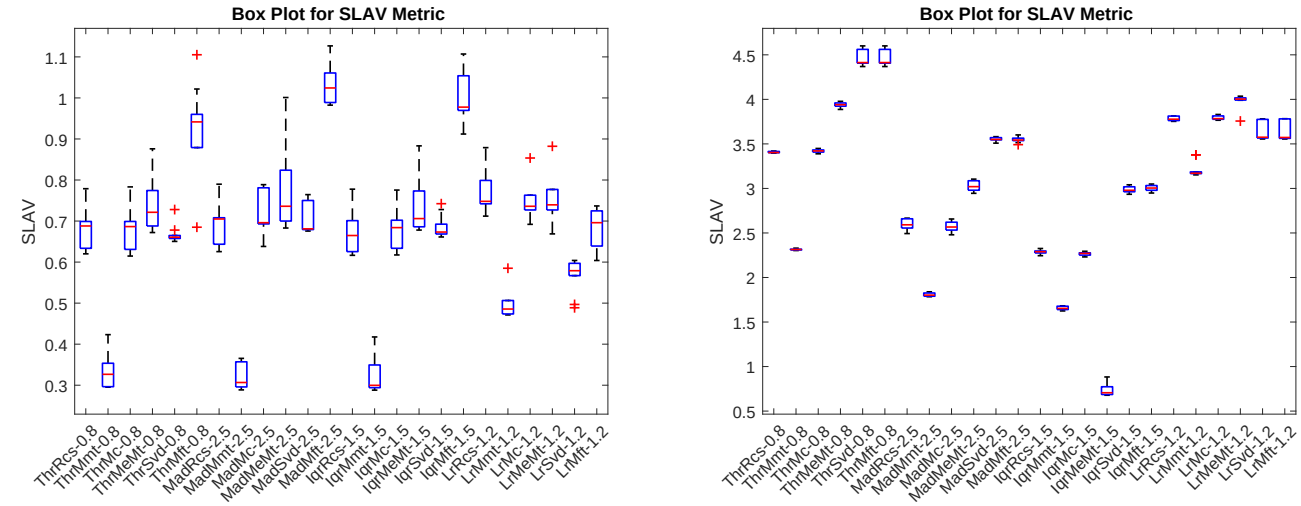
Cite As:
VERMA, J., KUMAR, S., KAIWARTYA, O., CAO, Y., LLORET, J., KATTI, C. P., KHAREL, R., 2018.
Enabling green computing in cloud environments: Network virtualization approach toward 5G support.
Transactions on Emerging Telecommunications Technologies, pp. e3434.



(a) PlanetLab Workload (b) Random Workload

Fig. 9: Total host shutdowns metric on workload traces of Planetlab servers and random synthetic workload

to existing VM selection models viz. RCS, MMT and MC due to associated server runtime with relatively high workload due to inefficient packing.



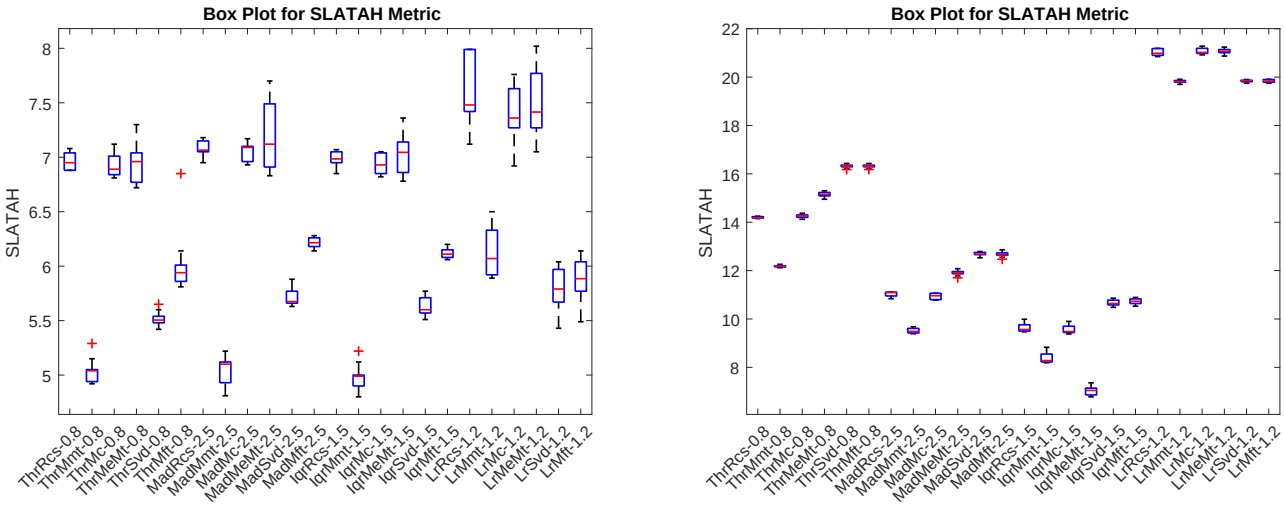
(a) PlanetLab Workload (b) Random Workload

Fig. 10: SLA violations metric on workload traces of Planetlab servers and random synthetic workload

Apart from the results discussed above, results of SLAV, SLAVO and SLAVM do not give any clue about the fitness of any algorithm due to variations existing in the results as can be seen in Fig. 10(a)–10(b), Fig. 11(a)–11(b) and Fig. 12(a)–12(b). However, the results obtained for proposed algorithms are not kind of outlier in comparison to the benchmark algorithms and therefore confirms their suitability. Meanwhile, $ESV(t)$ is not weighted metric and it is difficult to decide over efficiency of any algorithm by looking at value of $ESV(t)$ only. Considering example of Non Power Aware (NPA) and Dynamic Voltage and Frequency Scaling (DVFS) based application deployment environment, both of them do not generate any SLA violations due to no VM migrations (Fig. 8(a)–8(b)), hence results into minimum values of ESV that is zero. However, both of them are not the efficient way to deploy applications in Cloud environment, therefore, we do not set ESV as the primary criteria to decide over goodness of algorithm.

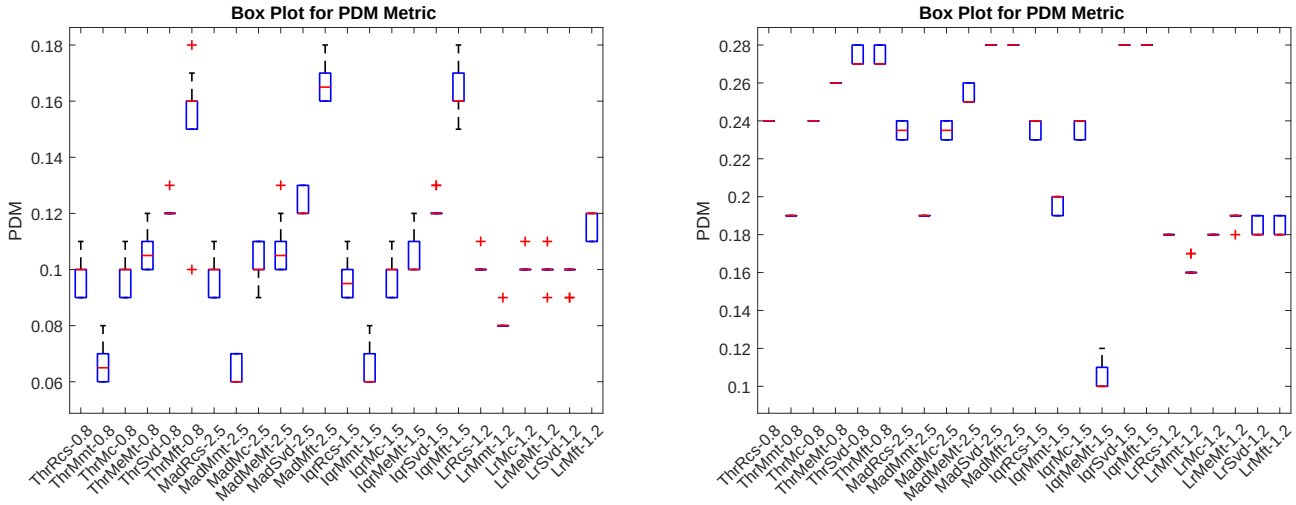
4.5 Test of Significance

Using workload described in Section 4.1, we simulated all combinations of four overloading detection methods (THR, MAD, IQR and LR) with six server offloading methods (RCS, MMT, MC, MeMT, SVD, and MFT). Moreover, we have taken safety parameter to be 0.8, 2.5, 1.5 and 1.2 for THR, MAD, IQR and LR, respectively to control aggressiveness of the methods for consolidating VMs. To perform the test of significance on proposed methods we select Kolmogorov-Smirnov



(a) PlanetLab Workload (b) Random Workload

Fig. 11: SLAVO metric on workload traces of PlanetLab servers and random synthetic workload

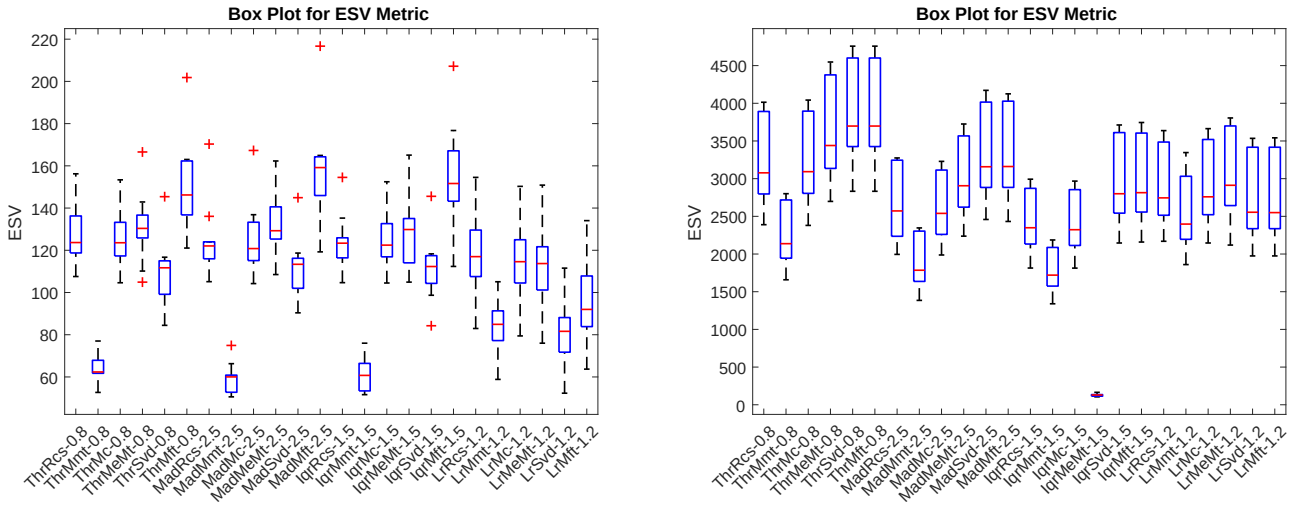


(a) PlanetLab Workload (b) Random Workload

Fig. 12: SLAVM metric on workload traces of PlanetLab servers and random synthetic workload

test and Shapiro-Wilk test [52] [53]. Both of the test yields $p - value > 0.05$ for all the proposed combination and pass the test of normality validating that results are statistically significantly different. The proposed algorithms not only reduce the energy consumption but also obtains low VM migrations and host shutdowns.

Cite As:
VERMA, J., KUMAR, S., KAIWARTYA, O., CAO, Y., LLORET, J., KATTI, C. P., KHAREL, R., 2018.
Enabling green computing in cloud environments: Network virtualization approach toward 5G support.
Transactions on Emerging Telecommunications Technologies, pp. e3434.



(a) PlanetLab Workload

(b) Random Workload

Fig. 13: ESV metric on workload traces of PlanetLab servers and random synthetic workload

5 Conclusion and Future Work

Resource under-utilization is the major root cause of higher energy consumption in cloud computing infrastructure and it is necessary to improve resource utilization to cut down the energy consumption. In this paper, we proposed Median Migration Time (MeMT), Smallest Void Detection (SVD) and Maximum Fill Technique (MFT) algorithms for efficient consolidation of VMs on relatively small number of physical hosts. We tested proposed algorithm with Static Threshold (THR), Median Absolute Deviations (MAD), Interquartile Range (IQR) and Local Regression (LR) based overloading detection methods. The proposed algorithms outperforms on energy consumption, total host shutdowns and total VM migrations metrics with all combinations of overloading detection methods. Additionally, we perform statistical test of significance on proposed algorithms. The algorithms pass the test of normality and confirms that results are statistically significantly different.

In future authors we intend to identify more real datasets to test out the behaviour of proposed methods. We also intend to propose VM cost/price models which is also an important and exciting issue for the cost optimization in the part of end-user and cloud provider [54]. Furthermore, we plan to investigate the performance of proposed methods over OpenStack Neat and CloudStack like environments as a future direction of research [55, 56].

References

[1] G. Cook and J. Van Horn, "How dirty is your data? a look at the energy choices that power cloud computing," *Greenpeace (April 2011)*, 2011.
[2] J. Varia, "Best practices in architecting cloud applications in the aws cloud," *Cloud Computing: Principles and Paradigms*, pp. 457–490, 2011.
[3] E. Hossain and M. Hasan, "5G cellular: key enabling technologies and research challenges," *IEEE Instrumentation & Measurement Magazine*, vol. 18, no. 3, pp. 11–21, 2015.
[4] C. Liang, F. R. Yu, and X. Zhang, "Information-centric network function virtualization over 5G mobile wireless networks," *IEEE network*, vol. 29, no. 3, pp. 68–74, 2015.
[5] S. Khan and J. L. Mauri, *Green Networking and communications: ICT for sustainability*. CRC press, 2013.
[6] X. Wang, C. Xu, G. Zhao, and S. Yu, "Tuna: an efficient and practical scheme for wireless access point in 5G networks virtualization," *IEEE Communications Letters*, 2017.
[7] Z. Feng, C. Qiu, Z. Feng, Z. Wei, W. Li, and P. Zhang, "An effective approach to 5G: Wireless network virtualization," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 53–59, 2015.
[8] I. Farris, T. Taleb, H. Flinck, and A. Iera, "Providing ultra-short latency to user-centric 5G applications at the mobile network edge," *Transactions on Emerging Telecommunications Technologies*, 2017.
[9] C. A. García-Pérez and P. Merino, "Experimental evaluation of fog computing techniques to reduce latency in LTE networks," *Transactions on Emerging Telecommunications Technologies*, 2017.
[10] A. S. Sadiq, T. Z. Almohammad, R. A. B. M. Khadri, A. A. Ahmed, and J. Lloret, "An energy-efficient cross-layer approach for cloud wireless green communications," in *Fog and Mobile Edge Computing (FMEC), 2017 Second International Conference on*. IEEE, 2017, pp. 230–234.
[11] A. T. Al-Hammouri, Z. Al-Ali, and B. Al-Duwairi, "ReCAP: A distributed captcha service at the edge of the network to handle server overload," *Transactions on Emerging Telecommunications Technologies*.
[12] J. Shuja, A. Gani, K. Ko, K. So, S. Mustafa, S. A. Madani, and M. K. Khan, "SIMDOM: A framework for simd instruction translation and offloading in heterogeneous mobile architectures," *Transactions on Emerging Telecommunications Technologies*, 2017.
[13] P. Arroba, J. M. Moya, J. L. Ayala, and R. Buyya, "Dynamic voltage and frequency scaling-aware dynamic consolidation of virtual machines for energy efficient cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 10, 2017.
[14] S. Andrade-Morelli, E. Ruiz-Sánchez, S. Sendra, and J. Lloret, "Router power consumption analysis: towards green communications," in *International Conference on Green Communications and Networking*. Springer, 2012, pp. 28–37.
[15] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[16] K.-D. Lange, "Identifying shades of green: The specpower benchmarks," *Computer*, vol. 42, no. 3, pp. 95–97, 2009.
[17] D. P. Doane and L. E. Seward, "Applied statistics in business and economics," *USA: Irwin*, 2005.
[18] F. Farahnakian, P. Liljeberg, and J. Plosila, "Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning," in *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on*. IEEE, 2014, pp. 500–507.
[19] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, N. T. Hieu, and H. Tenhunen, "Energy-aware vm consolidation in cloud data centers using utilization prediction model," *IEEE Transactions on Cloud Computing*, 2016.
[20] K. Park and V. S. Pai, "CoMon: a mostly-scalable monitoring system for planetlab," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 1, pp. 65–74, 2006.
[21] "Traces of Google workloads: Migrated from Google code repository," https://github.com/google/cluster-data, (Accessed on 06/16/2017).
[22] E. Arianyan, H. Taheri, and V. Khoshdel, "Novel fuzzy multi objective dvfs-aware consolidation heuristics for energy and sla efficient resource management in cloud data centers," *Journal of Network and Computer Applications*, vol. 78, pp. 43–61, 2017.
[23] F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, and H. Tenhunen, "Using ant colony system to consolidate vms for green cloud computing," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 187–198, 2015.
[24] R. Nathuji and K. Schwan, "VirtualPower: coordinated power management in virtualized enterprise systems," in *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6. ACM, 2007, pp. 265–278.
[25] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster computing*, vol. 12, no. 1, pp. 1–15, 2009.
[26] H. C. Andrew, "Forecasting, structural time series models and the kalman filter," *Cambridge University*, 1989.
[27] A. Verma, P. Ahuja, and A. Neogi, "pMapper: power and migration cost aware application placement in virtualized systems," in *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, 2008, pp. 243–264.
[28] N. Karmarkar and R. M. Karp, "An efficient approximation scheme for the one-dimensional bin-packing problem," in *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*. IEEE, 1982, pp. 312–320.
[29] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Resource pool management: Reactive versus proactive or lets be friends," *Computer Networks*, vol. 53, no. 17, pp. 2905–2922, 2009.
[30] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*. IEEE, 2010, pp. 577–578.
[31] J. K. Verma, C. P. Katti, and P. C. Saxena, "MADLVF: An energy efficient resource utilization approach for cloud computing," *International Journal of Information Technology and Computer Science*, vol. 6, no. 7, pp. 56–64, 2014.
[32] G. Orsini, D. Bade, and W. Lamersdorf, "Cloudaware: Empowering context-aware self-adaptation for mobile applications," *Transactions on Emerging Telecommunications Technologies*.
[33] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.
[34] J. V. Kistowski and S. Kounev, "Univariate Interpolation-based Modeling of Power and Performance," in *Proceedings of the 9th EAI International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2015)*, December 2015.
[35] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 273–286.
[36] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "Live migration of virtual machine based on full system trace and replay," in *Proceedings of the 18th ACM international symposium on High performance distributed computing*. ACM, 2009, pp. 101–110.
[37] M. Nelson, B.-H. Lim, G. Hutchins *et al.*, "Fast transparent migration for virtual machines." in *USENIX Annual technical conference, general track*, 2005, pp. 391–394.
[38] A. Lesovsky, *Getting Started with OVirt 3.3*. Packt Publishing Ltd, 2013.
[39] T. Wood, P. J. Shenoy, A. Venkataramani, M. S. Yousif *et al.*, "Black-box and gray-box strategies for virtual machine migration." in *NSDI*, vol. 7, 2007, pp. 17–17.
[40] A. B. Nagarajan, F. Mueller, C. Engelmann, and S. L. Scott, "Proactive fault tolerance for hpc with xen virtualization," in *Proceedings of the 21st annual international conference on Supercomputing*. ACM, 2007, pp. 23–32.
[41] A. Kangarlou, D. Xu, P. Ruth, and P. Eugster, "Taking snapshots of virtual networked environments," in *Virtualization Technology in Distributed Computing (VTDC), 2007 Second International Workshop on*. IEEE, 2007, pp. 1–8.
[42] B. Sotomayor, K. Keahey, and I. Foster, "Combining batch execution and leasing using virtual machines," in *Proceedings of the 17th international symposium on High performance distributed computing*. ACM, 2008, pp. 87–96.
[43] J. Jiang, Y. Feng, J. Zhao, and K. Li, "DataABC: A fast abc based energy-efficient live vm consolidation policy with data-intensive energy evaluation model," *Future Generation Computer Systems*, 2016.
[44] S. Power and P. B. Methodolody, "V2. 1," *Standard Performance and Evaluation Corporation (SPEC)*, 2011.
[45] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis, "Energy-efficient cloud computing," *The computer journal*, vol. 53, no. 7, pp. 1045–1051, 2010.
[46] S. Esfandiarpoor, A. Pahlavan, and M. Goudarzi, "Structure-aware online virtual machine consolidation for datacenter energy improvement in cloud computing," *Computers & Electrical Engineering*, vol. 42, pp. 74–89, 2015.
[47] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.
[48] "Amazon EC2 Instance Types: Amazon Web Services (AWS)," https://aws.amazon.com/ec2/instance-types/, (Accessed on 12/23/2017).
[49] H. Abdi, "Multiple correlation coefficient," *The University of Texas at Dallas*, 2007.
[50] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, "Server workload analysis for power minimization using consolidation," in *Proceedings of the 2009 conference on USENIX Annual technical conference*. USENIX Association, 2009, pp. 28–28.
[51] A. Murtazaev and S. Oh, "Sercon: Server consolidation algorithm using live migration of virtual machines for green computing," *IETE Technical Review*, vol. 28, no. 3, pp. 212–231, 2011.
[52] H. W. Lilliefors, "On the kolmogorov-smirnov test for normality with mean and variance unknown," *Journal of the American statistical Association*, vol. 62, no. 318, pp. 399–402, 1967.
[53] S. S. Shapiro and R. Francia, "An approximate analysis of variance test for normality," *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 215–216, 1972.
[54] J. K. Verma and C. P. Katti, "Study of cloud computing and its issues: A review." *Smart Computing Review*, vol. 4, no. 5, pp. 389–411, 2014.
[55] "OpenStack Neat: Dynamic consolidation of virtual machines in openstack clouds," http://openstack-neat.org/, (Accessed on 12/22/2017).
[56] "Apache cloudstack: Open source cloud computing," https://cloudstack.apache.org/, (Accessed on 12/22/2017).