Exploiting User Provided Information In Dynamic Consolidation of Virtual Machines to Minimize Energy Consumption of Cloud Data Centers

Md Anit Khan[1], Andrew P Paplinski[1], Abdul Malik Khan[1], Manzur Murshed[2] and Rajkumar Buyya[3]

[1] Faculty of Information Technology
Monash University
Clayton, Victoria, Australia
{md.khan, andrew.paplinski, malik.khan}@monash.edu

[2]School of Information Technology
Federation University Australia
Churchill, Victoria, Australia
manzur.murshed@federation.edu.au

[3]CLOUDS Lab, School of Computing and Information Systems
The University of Melbourne, Australia
rbuyya@unimelb.edu.au

Abstract— **Dynamic consolidation of Virtual Machines (VMs) can effectively enhance the resource utilization and energy-efficiency of the Cloud Data Centers (CDC). Existing research on Cloud resource reservation and scheduling signify that Cloud Service Users (CSUs) can play a crucial role in improving the resource utilization by providing valuable information to Cloud service providers. However, utilization of CSUs' provided information in minimization of energy consumption of CDC is a novel research direction. The challenges herein are twofold. First, finding the right benign information to be received from a CSU which can complement the energy-efficiency of CDC. Second, smart application of such information to significantly reduce the energy consumption of CDC. To address those research challenges, we have proposed a novel heuristic Dynamic VM Consolidation algorithm, *RTDVMC,* which minimizes the energy consumption of CDC through exploiting CSU provided information. Our research exemplifies the fact that if VMs are dynamically consolidated based on the time when a VM can be removed from CDC – a useful information to be received from respective CSU, then more physical machines can be turned into sleep state, yielding lower energy consumption. We have simulated the performance of *RTDVMC* with real Cloud workload traces originated from more than 800 PlanetLab VMs. The empirical figures affirm the superiority of *RTDVMC* over existing prominent Static and Adaptive Threshold based DVMC algorithms.**

Keywords—Cloud Energy Efficiency, Energy-Efficient Cloud, Energy-Efficient Cloud Data Center, VM Consolidation, Workload Consolidation, VM placement, Dynamic VM Consolidation, Dynamic VM Placement, Green Cloud, Cloud User Provided Information Aware, Release Time based VM Consolidation.

I. INTRODUCTION

Excessive energy consumption is one of the major drawbacks of Cloud Data Centers (CDCs). The most recent report on the United States data center energy usage [1] has revealed that amount of energy usage by all data centers in the US was 70 billion kWh in 2014, which is around 1.8% of the country's total energy usage. The momentum of rising energy consumption by data centers would not come to a halt in near future and is expected to rise by 4% from 2014-2020. According to that report, the approximate energy consumption by the US data centers in 2020 would be around 73 billion kWh. Additional studies, such as [2, 3] has highlighted that Google consumed as much energy as the city of San Francisco in 2015. Consequently, many developed countries have agreed to acknowledge and address the challenge of increasing energy consumption by Cloud data centers [4]. For instance, Joint Research Centre (JRC) of European Commission has created Code of Conduct for Energy Efficiency in Data Centres with an aim to inform and encourage data center owners and operators to effectively level off the energy consumption [5]. Most recently, in 2018, JRC has proposed a detailed guideline in relation to the best practices to maintain the energy efficiency of Data Centers [6].

One of the state-of-the-art energy-efficient Cloud resource management techniques, prevalent in the literature is VM Consolidation (VMC). When a VM is created for the first time, it is chosen to be placed in one of the many PMs located in a CDC. The algorithm which provides the solution of initial VM placement is known as VM Placement algorithm. To save energy, initial PMs are chosen in such a way, that VMs are placed in minimum possible number of PMs, so that minimum number of PMs would be required to keep it turned on and hence, CDC energy consumption can be minimized. Such algorithms to provide energy-efficient initial VM placement is referred to as Static VMC (SVMC) algorithm.

Although, SVMC algorithm complements Cloud energy-efficiency, it lacks in efficiency with the progression of time due to two reasons: Firstly, the initial VM-PM mapping changes over time and secondly, it does not consider the varying dynamics of resource demand of VMs over time. For instance, at times of hardware failure, VM(s) are migrated out to different PMs. Furthermore, resource demand of VM(s) hosted in a PM fluctuates with the varying workload over time, while the underlying amount of physical resources of a PM shared by its hosted VMs, remains fixed. At times, the total resource demand by VMs hosted in a single PM exceeds that PM's resource capacity and then those VMs encounter poor QoS due to resource contention, also known as SLA violation. Consequently, in order to prevent SLA violation, VM(s) are migrated out from an *Over-utilized PM* (*O-UPM*) (i.e., the PM with very high resource utilization) into a *Non-Over-utilized PM*, i.e., the PM that is not experiencing very high resource utilization. Thus, the initial VM-PM mapping changes over time. We refer to *Non-Over-utilized PM* as *Under-utilized PM* (*U-UPM*). Resource demand of VMs hosted in a PM may drop over time, widening the opportunity of hosting more VMs in that *U-UPM*. Now, if VMs of a *U-UPM* can be migrated out into new suitable *U-UPMs*, then that *U-UPM* which would now be having no VM, can either be put into a sleep state or shut down. Thus, energy can be saved through recurrent VM migrations according to the workload fluctuations of PMs. The algorithm, which considers the change in workload of PMs and dynamically migrates VMs according to the change in workload of PMs is a Dynamic VMC (DVMC) algorithm.

CSU provided information play a crucial role in many aspects of Cloud resource management system, such as resource scheduling, resource reservation and reservation based pricing scheme. However, from our extensive literature review on VMC algorithms [7], we have discovered that incorporation of CSU provided information in DVMC algorithm and its impact on energy-efficiency of CDC has not been investigated yet. Since, VMC is a NP-Hard problem, no VMC algorithm can guarantee or provide optimal solution in polynomial time. With this fact being underlined, we have presented a novel heuristic DVMC algorithm, referred to as *Release Time based DVMC* (*RTDVMC*) that utilizes CSU provided information to make more efficient VM consolidation decision in terms of reducing CDC energy consumption to promote green cloud. The rest of the paper is organized as follows:

In Section II, we have discussed the related literature on DVMC algorithms to explain the scope and position of our research. Before presenting our proposed algorithm, in Section III, we have explained various notations used in the algorithm, followed by mathematical modelling of different components of VMC in Section IV. Next, in Section V, we have articulated our proposed heuristic DVMC algorithm, *RTDVMC*. In Section VI, the experimental setup and performance evaluation of the proposed algorithm. Finally, in Section VII, we have summarized our research with future research directions.

II. Related Work

A DVMC algorithm has three core components: Source PM Selection, Migrating VM(s) Selection and Destination PM Selection. Source PM Selection selects PMs from which VMs would be migrated out. VM Selection component selects which VM(s) would be migrated out and Destination PM Selection component selects new PMs to host the migrating VMs.

Discussed previously in Section I, *O-UPMs* and *U-UPMs* are Source PMs from where VM(s) are selected to be migrated out. Based on the method of distinguishing a PM as *O-UPM* or *U-UPM,* DVMC algorithms can be classified into two groups: threshold-based DVMC and threshold-free DVMC. In threshold-based approach, a PM's resource utilization is compared to a threshold value. If the utilization is found as higher than the threshold value, then the respective PM is considered as *O-UPM* [8-10]. Conversely, in threshold-free DVMC algorithms, instead of comparing the utilization against a threshold value, the source PMs are selected either randomly or some functions are applied to favor PMs having either higher or lower resource utilization [11, 12]. Although, threshold-free approach leaves the option open to avoid local minima or local maxima, threshold-based approach limits the search space and is more time-efficient and hence, more popular.

Based on the nature of applied thresholds, threshold-based DVMC algorithms can be further classified into two groups: Static Threshold-based DVMC (STDVMC) [10] and Adaptive Threshold-based DVMC (ATDVMC) [9] algorithm. In ATDVMC algorithm, the threshold value changes according to the varied workload over time; whereas in STDVMC, the threshold value remains unchanged. Both STDVMC and ATDMVC come with their own pros and cons. Compare to STDVMC, ATDVMC limits SLA violation more by migrating out VMs before excessive resource utilization takes place. However, increased VM migration increases the probability of requirement of more PMs to host the migrating VMs, yielding higher energy consumption. We have compared our proposed DVMC algorithm with both STDVMC and ATDVMC.

Random Choice (RC), Minimization of VM Migration (MVM), High Potential Growth (HPG), Minimization of Migration Time (MMT) and Maximum Correlation (MC) are the most widely used VM selection algorithms incorporated in existing VMC algorithms [7, 8]. The RC algorithm has found to be the least energy-efficient [8]. The issue with MVM and HPG is that, VMs are sorted with respect to CPU demand avoiding the consideration of other types of resources, such as RAM and Network bandwidth. However, selection of migrating VM(s) based on only one specific type of resource, negatively affects the resource utilization maximization in terms of other types of resources [13]. Similar issue exists with the MC algorithm. Unlike MVM, HPG and MC algorithms, the MMT algorithm selects the VM having least migration time; resulting lower SLA violation. However, it does not consider energy consumption minimization aspect. Since, both the MMT algorithm and our proposed *RTDVMC* algorithm, consider time aspect of VMs to select migrating VMs, therefore, for performance comparison, we have selected those DVMC algorithms which uses the MMT algorithm as VM selection process.

Majority of the existing DVMC algorithms only consider CPU demand and CPU availability, while ignoring the requirement of RAM and Network Bandwidth is a big drawback which inhibits the applicability of these DVMC algorithms in real scenario. In contrast, our proposed *RTDVMC* algorithm gives equal importance to all types resources such as CPU, RAM and Network Bandwidth, while considers demand and availability of all types of resources.

Our research motivation is to investigate whether CSU provided information is useful to enhance Cloud energy-efficiency or not. We humbly acknowledge that [14] has presented a *PM Release Time* (explained in the following Section III.A) based initial VM placement or SVMC algorithm, which we have realized as an example of CSU provided information to be potentially useful for Cloud energy-efficiency. However, explained prior in Section I, SVMC algorithm loses its efficiency over time because of not considering the change in workload and resource availability. Oppositely, DVMC algorithm dynamically migrates VMs into fewer PMs in reflection to the changed workload and resource availability. Hence, DVMC is one of the key techniques that uphold the Cloud energy-efficiency, resource usage optimization, and profit maximization. As such, in this paper, we have proposed a novel energy-efficient STDVMC algorithm, namely *RTDVMC* which incorporates CSU provided information and gives equal importance to all types of resources.

III. CSU PROVIDED INFORMATION AND NOTATIONS USED

In our proposed DVMC algorithm, the CSU provided information is one of key distinguishing features used in VMC decision process. To ensure easy understanding of our proposed algorithm in the following section, we have first explained the assumptions and key terms, as well as the CSU provided information that have been used in the proposed algorithm.

A. VM Relase Time, PM Release Time and Assumptions

The information that would be required to receive from a CSU is the time when the CSU would release the resources that are in his acquisition. Since, DVMC algorithm attempts to dynamically select VMs to migrate into fewer number of active or turned on PMs, hence, for our DVMC algorithm, we consider VM as the resource of a CSU and the received information from a CSU is the time when the respective VM(s) can either be destroyed or shut down for a long period of time. We refer to such time as *VM Release Time* (*VMRT*). We assume that every VM would have *VMRT* which would be provided prior by respective CSU.

Apart from *VMRT*, another crucial term noteworthy explaining is *PM Release Time* (*PMRT*). *PMRT* refers to the time when a PM can be either shut down, or put into a sleep state that would consume no energy, or lower amount of energy compared to its active state. A PM can be shut down or put into sleep state, if it has either no VM hosted on it, or none of its hosted VMs is in active state. Since *VMRT* refers to the maximum time until which the VM would be in an active state, hence *PMRT* denotes the maximum *VMRT* value among all the *VMRT* values of VMs that are hosted in that PM. In the following section, we have articulated the notations used in the proposed algorithm.

B. Notations

TABLE I. NOTATIONS USED

Notations	Meaning
P	The set of m number of PMs, where $m \in \mathrm{N}$
$\vert\,\vert$	Cardinality of a Set
P_i	The ith PM of P, where $P_i \in P$ and $1 \le i \le \vert P\vert$
V	The set of VMs
V_j	The jth VM of V, where $1 \le j \le \vert V\vert$
V^i	The set of VMs hosted in PM, P_i, where $P_i \in P$ and $1 \le i \le \vert P\vert$
V_j^i	The jth VM of V^i. $V_j^i \in V^i$ and $1 \le j \le \vert V^i\vert$
T_{V^i}	The set of VMRT of VMs that are hosted in PM, P_i where $P_i \in P$ and $1 \le i \le \vert P\vert$
$T_{V_j^i}$	VMRT of VM, V_j^i, where $V_j^i \in V^i$, $1 \le i \le \vert P\vert$ and $1 \le j \le \vert V\vert$
T_{P_i}	PMRT of PM, P_i, where $P_i \in P$ $T_{P_i} = \max(T_{V^i})$ The maximum VMRT value among the set of VMRT of VMs that are hosted in the PM, P_i.
x	Placement Matrix
$x_{i,j}$	Element of Placement Matrix, x
R	The set of Different Types of Resources
R_k	The kth Resource of R. $R_k \in R$ and $1 \le k \le \vert R\vert$
i	Index to denote a PM which belongs to P
j	Index to denote a VM which belongs to V
k	Index to denote a type of Resource which is a member of R
D_j^k	Demand of Resource, R_k by VM, V_j, where $R_k \in R$, $V_j \in V$, $1 \le j \le \vert V\vert$ and $1 \le k \le \vert R\vert$
U_i^k	Utilization of Resource, R_k of PM, P_i, where $R_k \in R$, $P_i \in P$, $1 \le i \le \vert P\vert$, $1 \le k \le \vert R\vert$
C_i^k	Capacity of PM, P_i in terms of Resource, R_k, where $R_k \in R$, $P_i \in P$, $1 \le i \le \vert P\vert$, $1 \le k \le \vert R\vert$
θ_{MAX}	Maximum Threshold
OP	The set of *O-UPMs*
o	Index used to denote a type of *O-UPM* which is a member of OP
P_o	The oth *O-UPM*, where $P_o \in OP$
vmsToMigrate	The set VMs to migrate out into new PMs
V_l	The lth VM of *vmsToMigrate*, where $1 \le l \le \vert vmsToMigrate\vert$
l	Index to denote a PM in *vmsToMigrate*
V^o	The set of VMs hosted in an *O-UPM*, P_o, where $P_o \in OP$ and $1 \le o \le \vert OP\vert$
V_q^o	The qth VM of V^o, $V_q^o \in V^o$ and $1 \le q \le \vert V^o\vert$
q	Index to denote a VM which belongs to V^o
D_q^k	Demand of Resource, R_k by VM, V_q^o, where $R_k \in R$, $V_q^o \in V^o$, $1 \le q \le \vert V^o\vert$ and $1 \le k \le \vert R\vert$
U_o^k	Utilization of Resource, R_k of PM, P_o, where $R_k \in R$, $P_o \in OP$, $1 \le o \le \vert OP\vert$, $1 \le k \le \vert R\vert$
C_o^k	Capacity of PM, P_o in terms of Resource, R_k, where $R_k \in R$, $P_o \in OP$, $1 \le o \le \vert OP\vert$ and $1 \le k \le \vert R\vert$
P_x	The xth *Non-Over-utilized PM*, where $P_x \in NOP$ and $1 \le x \le \vert NOP\vert$

Notations	Meaning
x	Index to denote a PM which belongs to *NOP*
SP	The set of PMs that are in sleep mode or switched off
P_s	The s^{th} *PM in SP*, $P_s \in SP$ and $1 \le s \le \|SP\|$
s	Index to denote a PM which belongs to *SP*
P_d	Destination PM
NOP	The set of Non-*Over-utilized PMs* which are neither *O-UPMs* nor in Sleep mode or switched off
destinationPMs	The set of new destination PMs for migrating VMs of source *O-UPMs*
candidateSources	The set of PMs from which a PM would be selected as *U-UPM*
P_c	The c^{th} PM of P_c, where P_c belong to *candidateSources* and $1 \le c \le \|candidateSources\|$
c	Index to denote a PM in *candidateSources*
V^c	The set of VMs hosted in PM, P_c
V_n^c	VM belong to V^c, where $V_n^c \in V^c$, $1 \le n \le \|V^c\|$
n	Index to denote a VM, V_n^c hosted in P_c, where $P_c \in$ *candidateSources,* $1 \le n \le \|V^c\|$ and $1 \le c \le \|candidateSources\|$
candidateDestinations	The set of PMs from which a PM would be selected to host migrating VM of a *U-UPM*
destinations	The set of new destination PMs for migrating VMs of source *U-UPMs*
m	Index to denote a PM of *candidateDestinations*
P_m	A PM belong to *candidateDestinations*
E_i	Energy Consumption by PM, P_i
E_{CDC}	Energy Consumption by the CDC

IV. Modeling Resource Utilization, Constraints and Energy Consumption

Using the notations presented previously in section III.B, we have first explained the modelling of the resource utilization and the constraints used in proposed *RTDVMC* algorithm followed by the explanation of power consumption by a PM and CDC.

A. Modeling Resource Utilization

DVMC algorithm migrates VMs from one PM to another PM, so that VMs would be placed in minimum number of PMs and thus increase resource utilization and energy-efficiency. VMs hosted in a PM utilizes the resources of that PM. Let, U_i^k denotes utilization of Resource type, R_k of PM, P_i. Hence, the equation for calculating U_i^k is as follows:

$$U_i^k = \sum_{j=1}^{|V|} D_j^k \cdot x_{i,j} \tag{1}$$

where D_j^k denotes Demand of Resource type, R_k by VM, V_j, $x_{i,j}$ denotes the element of placement matrix, x and value of $x_{i,j}$ is determined as follows [13]:

$$x_{i,j} = \begin{cases} 1, & \mathit{if}\ V_j\ \mathit{is\ placed\ in}\ P_i \\ 0, & \mathit{Otherwise} \end{cases} \tag{2}$$

B. Modeling Resource Constraint

Since, Resource Capacity of a PM, P_i is fixed and it cannot provide additional resources to its hosted VMs than its capacity, Therefore, a VM, V_j can only be placed in a PM, P_i if it satisfies the following equation:

$$C_i^k - U_i^k \geq V_j^k \tag{3}$$

In other words, a PM, P_i cannot host a VM, V_j if the available resource of P_i is lesser than the resource demand of V_j. We denote such constraint presented in (3) as *Resource Constraint* (*RC*).

C. Modeling O-UPM and Maximum Utilization Threshold Constraint

As discussed previously in Section I, at times workload of VMs could rise very high resulting in steep resource utilization of the hosting PM. We denote such PM with heavy resource utilization as *O-UPM* and use a threshold, referred to as *Maximum Threshold,* θ_{MAX} to distinguish whether a PM is *Over-utilized* or not. Let us denote *OP* as a set of *O-UPMs* then,

$$OP = \{P_i \mid U_i^k \geq \theta_{MAX}\ \mathit{for\ any}\ R_k \in R\ \mathit{and}\ 1 \leq i \leq |P|\} \tag{4}$$

If VM(s) were not migrated out of an *O-UPM, then* SLA violation would unfold. In order to avoid causing SLA violation, during destination PM selection for a migrating VM, it is essential to ensure that hosting the migrating VM would not turn the destination into an *O-UPM, which* we have modelled through the following equation:

$$D_j^k + U_i^k < \theta_{MAX} \tag{5}$$

We refer such constraint presented in (5) as *Maximum Utilization Threshold Constraint* (*MUTC*).

D. Modeling Energy Consumption

Most of the existing VMC algorithms have mentioned that energy consumption of a PM is primarily dominated by its CPU utilization [10, 13]. Hence, our energy consumption model is a function of CPU utilization (6), where, E_i denotes the energy consumption by PM, P_i, where $P_i \in P$ and $1 \le i \le |P|$.

$$E_i = f(U_i^{CPU}) \tag{6}$$

In order to relate closely to the real energy consumption by PMs, for our energy consumption model, we have opted to draw energy consumption benchmark results of two different types of PMs: Hewlett-Packard Company ProLiant ML110 G4 [15] and Hewlett-Packard Company ProLiant ML110 G5 [16]. In Table II, we have articulated respective energy consumption of these two types of PMs at varying load level [15, 16]. Based on (6), we can determine the total energy consumption of the CDC through (7), where E_{CDC} denotes the total energy consumption of the CDC.

$$E_{CDC} = \sum_{i=1}^{|P|} E_i \tag{7}$$

V. Proposed Solution

VMC attempts to consolidate VMs in minimum number of PMs without violating any of the PMs' resource capacity, so that energy consumption can be minimized. Multi-dimensional Vector Packing Problem (MDVPP) refers to an NP-hard combinatorial optimization problem in which a set of items is required to be packed into minimum number of bins without violating any of the bins' capacities. Note that, if we refer to bins as PMs and items as VMs that are needed to be packed into minimum possible bins or PMs, given that none of bins or PMs' resource capacity is violated, then VMC turns into a NP-hard problem. As such, our proposed heuristic-based DVMC algorithm, *RTDVMC*, which aims to minimize the number of active PMs is listed as Algorithm 1.

A. The RTDVMC Algorithm

Algorithm 1 The *RTDVMC* algorithm

```
Input: $P$, $V$, $R$, $SP$
Output: VM Placement
The first phase: O-UPMs
1:  for each $P_i$ in $P$ do
2:      if (4) is satisfied then
3:          $OP \leftarrow \{P_i \cup OP\}$
4:      end if
5:  end for
6:  for each $P_o$ in $OP$ do
7:      $migratingVMs \leftarrow$ Invoke the VSO algorithm with $P_o$
8:      $vmsToMigrate \leftarrow \{migratingVMs \cup vmsToMigrate\}$
9:  end for
10: Sort vmsToMigrate in the order of decreasing VMRT
11: $NOP \leftarrow \{P - OP - SP\}$
12: for each $V_l$ in vmsToMigrate do
13:     $P_d \leftarrow$ Invoke the DPSVO algorithm with $V_l$ and NOP
14:     $destinationPMs \leftarrow \{P_d \cup destinationPMs\}$
15:     if $P_d$ is in SP then
16:         $SP \leftarrow \{SP - P_d\}$
17:         $NOP \leftarrow \{P_d \cup NOP\}$
18:     end if
19: end for
The second phase: U-UPMs
20: $candidateSources \leftarrow \{P - OP - SP - destinationPMs\}$
21: $candidateDestinations \leftarrow \{P - OP - SP\}$
22: Sort candidateSources in the order of increasing PMRT
23: for each $P_c$ in candidateSources do
24:     $candidateDestinations \leftarrow \{candidateDestinations - P_c\}$
25:     $destinations \leftarrow$ Invoke the DPSVU algorithm with $P_c$
            and candidateDestinations
26:     $candidateSources \leftarrow \{candidateSources - destinations\}$
27: end for
```

Algorithm 2 The VMs Selection From *O-UPM* (*VSO*) algorithm

```
Input: The O-UPM, $P_o$
Output: List of VMs to be migrated out from the given O-UPM
1:  Sort $V^o$, the set of VMs of $P_o$ in order of decreasing VMRT
2:  $q \leftarrow 1$
3:  while $q \le |V^o|$ and $\theta_{MAX} < U_o^k$ for any $R_k$ in $R$ do
4:      $migratingVMs \leftarrow \{V_q^o \cup migratingVMs\}$
5:      for each $R_k$ in $R$ do
6:          $U_o^k \leftarrow U_o^k - D_q^k$
7:          $C_o^k \leftarrow C_o^k + D_q^k$
8:      end for
9:      $q \leftarrow q + 1$
10: end while
11: return migratingVMs
```

Algorithm 3 The Destination PM Selection for VM of *O-UPM* (*DPSVO*) algorithm

```
Input: The VM $V_j$ to be migrated out from a O-UPM
Input: NOP
Output: The new destination PM for the given migrating VM
1:  Sort NOP in the order of increasing PMRT
2:  for each $P_x$ in NOP do
3:      suitable $\leftarrow$ Invoke the PST algorithm with $P_x$ and $V_j$
4:      if suitable is true then
5:          return $P_x$
6:      end if
7:  end for
8:  return the most energy-efficient $P_s$ from SP
```

Algorithm 4 The PM Suitability Test (*PST*) algorithm

```
Input: PM, VM
Output: A decision whether the given PM can host the given VM
1:  if (3) and (5) are satisfied for all $R_k$ in $R$ then
2:      return true
3:  end if
4:  return false
```

Algorithm 5 The Destination PMs Selection for VMs of *U-UPMs* (*DPSVU*) algorithm

```
Input: A $UPM$, $P_c$ from the set of candidateSources
Input: candidateDestinations
Output: List of new destination PMs to host migrating VMs
1:  Sort $V^c$, the set of VMs of $P_c$ in order of decreasing VMRT
2:  for each $V_n^c$ in $V^c$ do
3:      $P_d \leftarrow$ null
4:      Sort candidateDestinations in order of increasing PMRT
5:      for each $P_m$ in candidateDestinations do
6:          suitable $\leftarrow$ Invoke the PST algorithm with $P_m$ and $V_n^c$
7:          if suitable is true
8:              $P_d \leftarrow P_m$
9:              hostList $\leftarrow \{P_d \cup$ hostList$\}$
10:             break loop
11:         end if
12:     end for
13:     if $P_d$ is null then
14:       break loop
15:     end if
16: end for
17: return hostList
```

B. Two Phases of RTDVMC Algorithm

In this section, we have explained our proposed *RTDVMC* algorithm. The objectives of migrating VMs out of a PM and then placing those migrating VMs into new destination PMs are twofold: *Firstly*, limiting SLA violations which are taking place in *O-UPMs* and *Secondly*, limiting the total number of active PMs by migrating VMs of a *U-UPM* into new appropriate *U-UPMs*, so that the *U-UPM* having no VM can be put into sleep state. Hence, *RTDVMC* algorithm works in two phases. In **the first phase**, SLA violations of *O-UPMs* are limited through migrating out required number of VMs from those *O-UPMs* (Line 1 to 19 of Algorithm 1); while in **the second phase**, VMs from a range of *U-UPMs* are migrated out and then consolidated in lesser number of active PMs (Line 20 to 27 of Algorithm 1). In the following section, we have comprehensibly discussed each of these phases and its components:

1) The first phase O-UPms: In order to identify *O-UPMs*, we compare a PM's resource (i.e., CPU, RAM and Bandwidth) utilization (1) to a fixed threshold value, θ_{MAX}. As expressed in (4), if for any resource type, the resource utilization of a PM is found equal or greater than θ_{MAX}, then the PM is referred to as *O-UPM* (Line 1 to Line 5 of Algorithm 1). Next, VM(s) are selected from O-UPMs to migrate out into new destination PMs.

a) VMs Selection From O-UPMs: The VSO algorithm, articulated as Algorithm 2, provides the solution to select VMs which would be migrated out from an *O-UPM*. In order to select VM(s) from an *O-UPM*, a list of VMs sorted in the order of decreasing *VMRT* is first prepared (Line 1 of Algorithm 2) (i.e., the first VM of the sorted list has the largest *VMRT* value, while the second VM of the sorted list has the second largest *VMRT* value and so forth) and then, the first VM of the sorted list is selected to migrate out, followed by the second VM and so forth, until the resource utilization of the *O-UPM* drops below θ_{MAX} (Line 2 to 11 of Algorithm 2). For every *O-UPM*, such sorted list of migrating VMs is created and then, combining all those lists a single list of VMs to be migrated out of all *O-UPMs* is created which is further sorted in descending order of *VMRT* (Line 6 to 10 of Algorithm 1). We denote the list as *vmsToMigrate*. The reason to migrate out VMs with greater *VMRT* is to minimize the duration of active state of *O-UPMs*, so that at a particular point in time, more number of PMs can be found that are in sleep state which would lower the energy consumption.

b) Destination PMs Selection For VMs of O-UPMs: After creating the sorted list of migrating VMs from *O-UPMs* (i.e., *vmsToMigrate*), new destination PMs for those migrating VMs are selected through executing the *DPSVO* algorithm (referred to as Algorithm 3). The new destination PMs are chosen from the set of PMs which are neither Over-utilized PMs nor inactive PMs (i.e., the PMs that are currently in the sleep state or in the switched off state). We denote such a set of PMs as *NOP* (Line 11 of Algorithm 1). The PMs belong to *NOP* are first sorted in the order of increasing *PMRT* (Line 1 of Algorithm 3). Next, for the first VM of *vmsToMigrate*, it is checked whether the first PM of sorted *NOP* is suitable to host that PM or not. The *PST* algorithm, listed as Algorithm 4 is invoked to decide whether the PM is suitable to host the given VM. Referring back to Section IV.B and Section IV.C, if the PM satisfies both *RC* (3) and *MUTC* (5) constraints, then the PM is considered as suitable (Line 1 to 4 of Algorithm 4). If the first PM of sorted *NOP* is found as unsuitable, then the second PM of sorted *NOP* is checked, followed by the third PM and so forth,until a suitable PM is found (Line 2 to 7 of Algorithm 3). However, if none of the PMs from sorted *NOP* is found as suitable, then the most energy-efficient PM is chosen from the set of inactive PMs (i.e., the PMs which are currently in the sleep state or in the switched off state) to host the migrating VM (Line 8 of Algorithm 3). The same process is repeated to select a destination PM for the second VM of *vmsToMigrate*, followed by the third VM and so forth (Line 12 to 19 of Algorithm 1). Thus new destination PMs are selected for all migrating VMs of *O-UPMs*.

Although, selecting the VM with highest *VMRT* to migrate out from O-UPMs minimizes the active duration of O-UPMs; one might argue that in worst-case scenario, it can pose the increased active duration on destination PMs. To alleviate such worst-case scenario, we have followed the approach to sort all the migrating VMs of *O-UPMs* in descending order of *VMRT* and sort all the candidate

TABLE II. ENERGY CONSUMPTION AT DIFFERENT LOAD LEVELS IN WTS

Server	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

destination PMs in ascending order of *PMRT* and then compared the highest *VMRT* against the lowest *PMRT* first, followed by the second lowest *PMRT* and so forth which ensures that the hosting of a new VM would incur the least amount of increased active duration of destination PMs.

However, given *RC* (3) *and MUTC* (5) constraints, in best-case scenario, without any increase of active duration of the destination PM which is going into sleep state first compared to all other candidate PMs, will finish the most time-consuming task before it moves into sleep state, while leaving the shorter tasks for rest of candidate PMs. In other words, considering *RC* (3) and *MUTC* (5) constraints, the destination PM selection process would provide the best-fit solution in terms of maximizing utilization of a PM before it moves into sleep state or switched-off state, since the most time-consuming job is finished by the PM which is going to be into sleep state faster than the rest of the candidate PMs. Hence, by ensuring the increased utilization of active PMs that are going into sleep state quicker, the probability of increased active duration of rest of the PMs due to upcoming workload is minimized which limits the total number of active PMs at a particular point in time. Furthermore, if no suitable PM is found in the list of active PMs of *NOP*, then that switched off PM or the PM which is in sleep state is selected which would exert the least amount of increase in energy consumption due to hosting the migrating VM. Thus, energy-efficiency is ensured.

2) The second phase U-UPMs: After completion of VM migrations from O-UPMs, the algorithm enters into **the second phase** where VMs from *U-UPMs* are attempted to consolidate in fewer number of PMs (Line 20 to 27 of Algorithm 1). The PMs with resource utilization lower than θ_{MAX} are denoted as *U-UPMs*. As previously discussed in Section I, workload of *U-UPMs* might drop over time, allowing to pack more VMs in it. Hence, if VMs of an *U-UPM* can be migrated out into rest of the *U-UPMs*, then that *U-UPM* which would not be having any VM, can be either shut down or put into sleep state, resulting in reduced number of active PMs and lower energy consumption. We aim to maximize the utilization of an *U-UPM* without increasing its active duration which eventually would minimize the load of rest of the PMs. Consequently, with reduced workload remaining, it would be possible to consolidate more VMs in fewer number of *U-UPMs* than before. Hence, more *U-UPMs* can be put in the sleep state, resulting reduced energy consumption.

To implement this strategy in our proposed *RTDVMC* algorithm, we first prepare the set of PMs which are potential candiates as *U-UPMs*, denoted by *candidateSources* (CS) and the set of candidate destination PMs, denoted by *candidateDestinations* (CD) which can potentially host those VM(s) that would be migrated out from *U-UPMs*. Next, we assign all the PMs in both CS and CD except *O-UPMs* and the set of PMs that are in sleep state or in switched off state (denoted by *SP*), so that *O-UPMs* do not turn into *O-UPMs* again due to hosting more VMs, while not waking up the PMs which are in sleep state or switched off state would complement the energy consumption minimization. In addition, from CS, we remove the set of PMs which hosted those VMs that were migrated out of *O-UPM.* These set of PMs is denoted by *destinationPMs* (Line 20 to 21 of Algorithm 1). Since, VMs may migrate out from an *U-UPM,* threfore, if PMs of *destinationPMs* were selected as *U-UPMs*, then a number of such VMs might migrate out from PMs of *destinationPMs* which had already been migrated out once in the first phase of the algorithm. Hence, re-migration would have taken place which would incur the increased SLA violation overhead.

a) Source U-UPMs Selection: After preparing the set of candidate source *U-UPMs* (i.e., CS) and set of potential destination PMs to host migrating VMs of U-UPMs (i.e., CD), we reach to the state where one or more source *U-UPMs* belong to CS are selected from which VM(s) would be migrated out into suitable destination PMs of CD. In order to do so, we first sort the PMs of CS in increasing order of *PMRT* (Line 22 of Algorithm 1) and then we start to select destination PMs for the VM(s) of the first PM of sorted CS (i.e., the PM with lowest *PMRT* or the PM which is going into sleep state first), followed by the second PM and so forth (Line 23 to 25 of Algorithm 1). The migrating VMs selection and the corresponding destination PMs selection process are accomplished through the *DPSVU* algorithm, referred to as Algorithm 5 and is articulated in the following section.

b) Selection of Migrating VMs and Corresponding Destination PMs: In the destination PMs selection part for migrating VMs, we start with the first PM of sorted CS, followed by the second PM and so forth. The PM which is selected to be checked whether its VMs can be migrate out into suitable destination PMs or not, we first remove that PM from CD, since a source PM cannot be the destination PM of VMs that were migrated out from itself in the first place (Line 24 of Algorithm 1). Next, we sort the VMs of that selected PM in the order of decreasing *VMRT* (Line 1 of Algorithm 5) and start searching for a suitable PM for the first VM with highest *VMRT* (Line 2 of Algorithm 5). The rationale to select the VM with highest *VMRT* is that it would shorten the active duration of the source PM.

In order to find the respective destination PM for a VM, the PMs of CD are first sorted in the order of increasing *PMRT* (Line 4 of Algorithm 5) and checked if the first PM of sorted CD is suitable to host the VM or not. The *PST* algorithm, listed as Algorithm 4 is utilized to check if a PM is suitable for hosting the given VM. However, if the first PM of sorted CD is found as unsuitable, then the suitability of the second PM of sorted CD is checked, followed by the third PM and so forth, unless a suitable PM is found (Line 5 to 12 of Algorithm 5). If no such suitable PM can be found in sorted CD, the VM is chosen not to be migrated out and the destination PM selection process for VMs of an *U-UPM* terminates (Line 13 to 16 of Algorithm 5). To explain more, if a VM with higher *VMRT* can not be migrated out, then the rest of the VMs with lower *VMRT* are not migrated out, since a PM's active duration can only be lowered if the VM with highest *VMRT* among all of its hosted VMs can be migrated out.

One noteworthy point to underline is that after finding the set of new destination PMs, denoted by *destinations* for VMs of a source *U-UPM* through the *DPSVU* algorithm, those new destination PMs of *destinations* are removed from CS (Line 26 of Algorithm 1), which restricts unnecessary VM re-migration and consequent SLA violation as well as improves the running time of the *RTDVMC* algorithm.

In summary, the VM selection process from an *U-UPM* only selects VMs with higher *VMRT* which minimizes the active duration of the source PM, while the destination PM selection process selects suitable destination PMs with higher *PMRT* than the source PM (Line 22 to 27 of T Algorithm 1) ensuring that destination PMs' active duration is not extended because of hosting any migrating VM of a *U-UPM*. Consequently, active duration of source PM is shortened without exerting prolonged ative duration of destination PM as it is illustrated in Fig 1. Hence, at a particular point in time, the total number of PMs in sleep state would increase, yielding lower energy consumption. In addition, given *RC* (3) *and MUTC* (5) constraints, the destination PM selection process places a migrating VM in the PM which is going to be in sleep state sooner than rest of the suitable PMs, resulting in increased resource utilization of a PM while it is in active state. In other words, considering *RC* (3) and *MUTC* (5) constraints, *RTDVMC* provides the best fit solution in terms of maximizing utilization of an *U-UPM* before it moves into sleep state or switched off state; since, the longest job across time dimension is assigned to the PM which is going to be switched into idle state soonest compared to all the available suitable PMs.

VI. Performance Evaluation

To evaluate the performance of *RTDVMC*, we have modelled and simulated a cloud environment and implemented our proposed algorithm, *RTDVMC* in CloudSim [17]. Then, we have simulated *RTDVMC* algorithm in different workload scenarios. The other notable DVMC algorithms, namely, ACS-VMC [10], THR-MMT [8], MAD [9], IQR [9], LR [9] with which we have compared the performance of *RTDVMC*, also evaluated using CloudSim.

To ensure the fairness in performance comparison of *RTDVMC* with ACS-VMC, THR-MMT, MAD, IQR and LR, we have carried out our simulation using same environment in CloudSim in relation to CDC, VM, PM and energy module, as used by respective authors in their research. Hence, our simulated CDC is comprised of 800 heterogeneous PMs. Two different server configurations HP ProLiant ML110 G4 (Intel Xeon 3040, 2 cores→1860 MHz, 4 GB) [15], and HP ProLiant ML110 G5 (Intel Xeon 3075, 2 cores→2660 MHz, 4 GB) have been used. Each server is provided with 1 GB/s network bandwidth. The energy consumption characteristics of these servers with varying workload is articulated in Table II**.**

The characteristics of the different VM types match with the VMs used by ACS-VMC [10], THR-MMT [8], MAD [9], IQR [9], LR [9] and correspond to Amazon EC2 instance types [18]. However, the difference between the simulated VMs and Amazon EC2 instance types is that the simulated VMs are single-core, which is explained by the fact that the workload data used for the simulations come from single-core VMs. Since, the single-core is used, the amount of RAM is divided by the number of cores for each VM type: High-CPU Medium Instance (2500 MIPS, 0.85 GB); Extra Large Instance (2000 MIPS, 3.75 GB); Small Instance (1000 MIPS, 1.7 GB); and Micro Instance (500 MIPS, 613 MB). At the outset, VMs are provided with the resources defined by the VM types. However, during the lifetime, VMs utilize less resources according to the workload data, widening opportunities for dynamic consolidation.

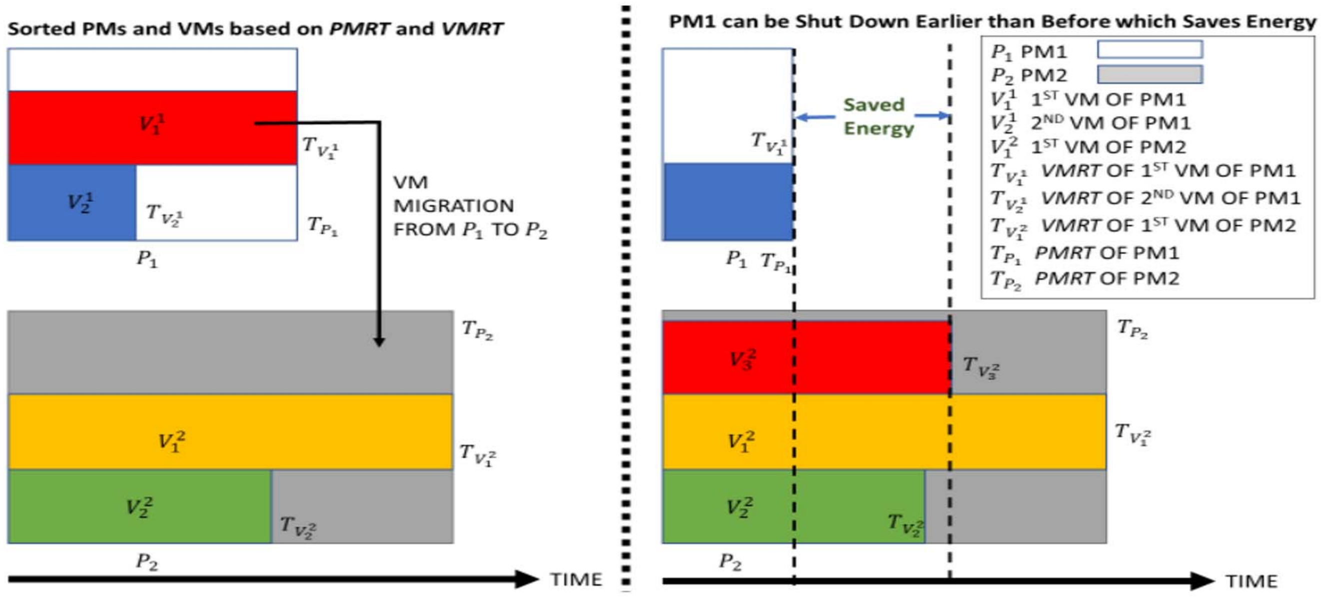

Fig. 1. VMC by RTDVMC

The objective of DVMC is to minimize the energy consumption of CDC. Hence, we have evaluated the performance of *RTDVMC* in terms of CDC energy consumption and compared with that of ACS-VMC [10], THR-MMT [8], MAD [9], IQR [9], LR [9]. As expressed in (7), CDC energy consumption is the sum of energy consumption of all the PMs, while each PM's energy consumption is derived from Table II according to its current CPU utilization. The upper utilization threshold for each of algorithms is set to 80%. We have measured the performance of *RTDVMC* and other DVMC algorithms with two types of workload which are articulated in the following sections.

A. Synthetic Workload

We created a synthetic workload with simulation where every VM runs applications with variable utilization of CPU which is generated with a uniform distribution. In Fig. 2, we have articulated the amount of CDC energy consumption caused by the *RTDVMC*, ACS-VMC, IQR, MAD, THR-MMT and LR for the random workload. The performance enhancements by *RTTDVMC* are 11.4%, 40.5%, 39.2%, 32.8% and 22.9% respectively. The reason of significant performance improvement displayed by *RTDVMC* is that *RTDVMC* takes optimal VM migration decision through considering *VMRT* and *PMRT* which minimizes the active duration of source PMs without increasing the active duration of destination PMs as illustrated in Fig. 1.

B. Real Workload

We have evaluated the performance of *RTDVMC* with real Cloud workload traffic traces. Real workload data is provided as part of the CoMon project, a monitoring infrastructure for PlanetLab [19]. Data of CPU usage of thousands of VMs has been collected every five minutes, while these VMs had been hosted in PMs spread globally across 500 locations. We have measured the performance of *RTDVMC* and rest of the DVMC algorithms in terms of CDC energy consumption with the data of three different days: 3 March, 6 March and 9 March, which are plotted in Fig. 3, Fig. 4, and Fig. 5 respectively. Respective authors of ACS-VMC, IQR, MAD, THR-MMT and LR had also used same data to evaluate the performance.

The empirical outcomes as portrayed in Fig. 3, Fig. 4 and Fig. 5, demonstrates that *RTDVMC* effectively limits CDC energy consumption. For the data of 3 March (i.e., Fig. 3), the energy consumption minimization by *RTDVMC* compare to ACS-VMC, IQR, MAD, THR-MMT and LR are 15.6%, 41%, 40.4%, 42.2% and 32.3%, respectively. For rest of the days, *RTDVMC* also shows significant performance improvement compare to the rest of the algorithms as shown in Fig. 4 and Fig. 5.

The main reason of performance improvement by *RTDVMC* compared to rest of the algorithms is that *RTDVMC*

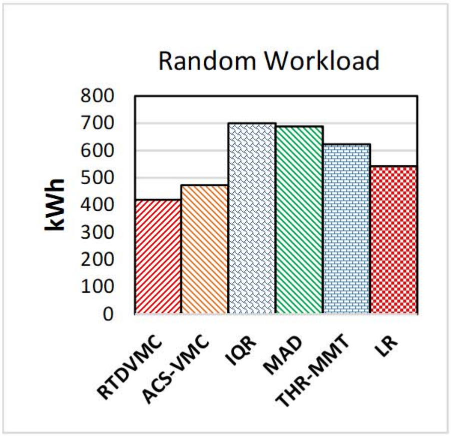

Fig. 2. CDC Energy Consumption with Random Workload

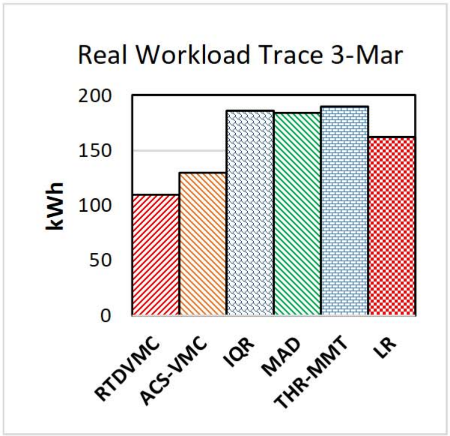

Fig. 3. CDC Energy Consumption with Real Workload of 3 Mar 2011

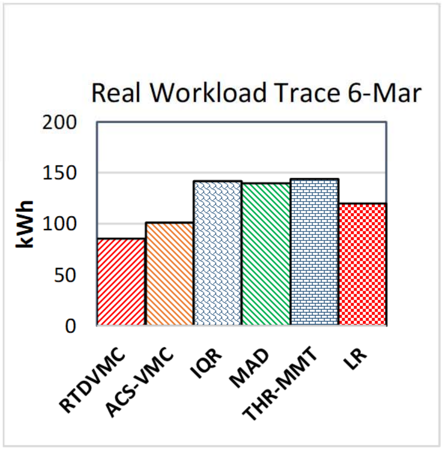

Fig. 4. CDC Energy Consumption with Real Workload of 6 Mar 2011

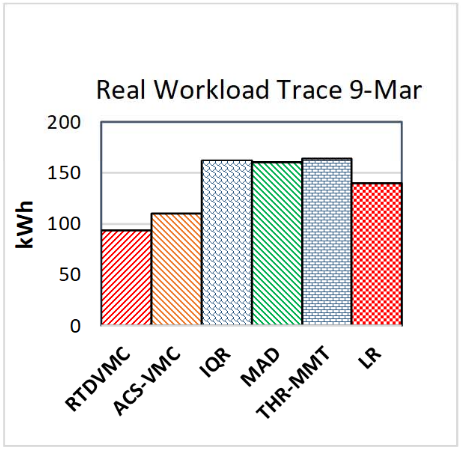

Fig. 5. CDC Energy Consumption with Real Workload of 9 Mar 2011

utilizes the information received from CSU (i.e., *VMRT*) in VM consolidation decision process, while none of the existing algorithms takes *VMRT* and *PMRT* into account in order to consolidate VMs dynamically.

One remarkable point to note from the above presented experimental results is that Ant Colony System (ACS) metaheuristic shows better performance among rest of the algorithms in every experiment. The potential reason is that ACS meta-heuristic leaves the option open to avoid local minima or local maxima. Beyond that, it is noteworthy that utilizing the advantage of regression, LR outperforms THR-MMT, IQR and MAD.

VII. Conclusions and future work

VMC is one of the state-of-the-art energy-efficient Cloud resource management technique, which effectively lowers CDC energy consumption through increasing Cloud resource utilization. Through our extensive literature review on VMC algorithms [7], we have learned that incorporation of CSU provided information in DVMC algorithm and its impact on energy-efficiency of CDC has not been investigated. In this research work, we have articulated a novel heuristic DVMC algorithm, *RTDVMC*, which exploits CSU provided information such as *VMRT* in its substratum.

RTDVMC takes *Release Time* of VMs and PMs as well as energy-efficiency of PMs into account in three components of DVMC algorithm: Source PM selection, VM selection and Destination PM Selection. The *PMRT* based source PM selection (i.e., *U-UPM* selection) and *VMRT* based VM selection strategy combined with *PMRT* based destination PM selection technique increases the resource utilization of PMs while endeavouring to lower the turned-on durations of source PMs through migrating out VM(s) having higher *VMRT* into destination PMs without undesirably affecting (i.e., increasing) the respective turned-on durations. Hence, at a given point in time, the *RTDVMC* can shut down more PMs compared to existing DVMC algorithms, which do not consider *VMRT* or any of the CSU feedback in VMC decision process. Consequently, improved energy-efficiency is achieved as reflected in empirical evaluations.

We assumed that *VMRT* of all VMs would be known prior. However, such mix of VMs may exist in CDC that for some VMs, *VMRT* would be known and for rest of the VMs, *VMRT* would be unknown. Furthermore, since *VMRT* information would be provided by CSU, accuracy of information can affect the performance of the algorithm. We aim to address such challenges in our future research. Another important aspect of DVMC algorithm to investigate is to bound SLA violations by migrating out VMs from *O-UPs*. In our forthcoming research, we would investigate the impact on QoS by our proposed DVMC algorithm.

References

[1] A. Shehabi, et al.: 'Uniter States Data Center Energy Usage Report.', 'Book Uniter States Data Center Energy Usage Report.' (LAWRENCE BERKLEY NATIONAL LABORATORY, 2016, edn.).

[2] 'The Energy of the Cloud', 2016. [Online]. Available: http://large.stanford.edu/courses/2016/ph240/brackbill2/, accessed 05/03/2018.

[3] Q. Hardy: 'Google Says It Will Run Entirely on Renewable Energy in 2017', 2016. [Online]. Available: https://www.nytimes.com/2016/12/06/technology/google-says-it-will-run-entirely-on-renewable-energy-in-2017.html, accessed 05/03/2018.

[4] M. Avgerinou, P. Bertoldi, and L. Castellazzi: 'Trends in Data Centre Energy Consumption under the European Code of Conduct for Data Centre Energy Efficiency', Energies, 2017, 10, (10), pp. 1470.

[5] 'Code of Conduct for Energy Efficiency in Data Centres', 2016. [Online]. Available: https://ec.europa.eu/jrc/en/energy-efficiency/code-conduct/datacentres, accessed 05/03/2018.

[6] A. Mark, et al.: '2018 Best Practice Guidelines for the EU Code of Conduct on Data Centre Energy Efficiency',: 'Book 2018 Best Practice Guidelines for the EU Code of Conduct on Data Centre Energy Efficiency' (2018, edn.). [Online]. Available: https://ec.europa.eu/jrc/en/publication/2018-best-practice-guidelines-eu-code-conduct-data-centre-energy-efficiency-version-910.

[7] M.A. Khan, A. Paplinski, A.M. Khan, M. Murshed, and R. Buyya: 'Dynamic Virtual Machine Consolidation Algorithms for Energy-Efficient Cloud Resource Management: A Review': 'Sustainable Cloud and Energy Services' (Springer, 2018), pp. 135-165.

[8] 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, 2012, 28, (5), pp. 755-768.

[9] 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, 2012, 24, (13), pp. 1397-1420.

[10] F. Farahnakian, et al.: 'Using Ant Colony System to Consolidate VMs for Green Cloud Computing', IEEE Transactions on Services Computing, 2015, 8, (2), pp. 187-198.

[11] R. Nasim, J. Taheri, and A.J. Kassler: 'Optimizing Virtual Machine Consolidation in Virtualized Datacenters Using Resource Sensitivity', 'Book Optimizing Virtual Machine Consolidation in Virtualized Datacenters Using Resource Sensitivity' (2016, edn.), pp. 168-175.

[12] A. Marcel, et al.: 'Thermal aware workload consolidation in cloud data centers', 'Book Thermal aware workload consolidation in cloud data centers' (2016, edn.), pp. 377-384.

[13] M.H. Ferdaus, M. Murshed, R.N. Calheiros, and R. Buyya: 'Virtual machine consolidation in cloud data centers using ACO metaheuristic', 'Book Virtual machine consolidation in cloud data centers using ACO metaheuristic' (Springer, 2014, edn.), pp. 306-317.

[14] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes: 'Release-time aware VM placement', 'Book Release-time aware VM placement' (2014, edn.), pp. 122-126.

[15] 'SPECpower_ssj2008', 2017. [Online]. Available: http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110124-00338.html, accessed 28/10/2017.

[16] 'SPECpower_ssj2008', 2017. [Online]. Available: http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110124-00339.html, accessed 28/10/2017.

[17] 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, 2011, 41, (1), pp. 23-50.

[18] 'Amazon EC2 Instance Types', 2017. [Online]. Available: https://aws.amazon.com/ec2/instance-types/, accessed 9/11/2017.

[19] K. Park, and V.S. Pai: 'CoMon: a mostly-scalable monitoring system for PlanetLab', ACM SIGOPS Operating Systems Review, 2006, 40, (1), pp. 65-74.