Directed Research Project
Title: QoS aware Virtual Machine Consolidation in Cloud Datacenter

Mohammad Alaul Haque Monil
Department of Computer and Information Science
University of Oregon
Email: mmonil@cs.uoregon.edu

Allen D. Malony
Department of Computer and Information Science
University of Oregon
Email: malony@cs.uoregon.edu

Abstract—**With rapid growth of cloud industry in recent years, energy consumption of warehouse-scale datacenter has become a major concern. Energy aware Virtual Machine consolidation has proven to be one of the most effective solutions for tackling this problem. Among the sub problems of VM consolidation, VM placement is the trickiest and can be treated as bin packing problem which is NP hard, hence, it is logical to apply heuristic approach. The main challenge of VM consolidation is to achieve a balance between energy consumption and quality of service(QoS). In this research, we evaluate this problem and design a combined strategy using best fit decreasing bin packing method and pass-based optimization in VM placement for an efficient VM consolidation. Moreover, for maintaining energy-QoS balance, we propose a rank-based VM selection strategy and a mechanism to impose constraint on QoS to achieve desired level of performance. In addition, we propose an under-load detection method using an optimization phase. We have used CloudSim toolkit to simulate our experiments. To evaluate the performance of the proposed algorithms, we used real-world work load traces from thousand VMs and also used randomly generated work loads. Results demonstrate that our proposed methods outperform other existing methods.**

I. Introduction

Cloud computing can be classified as a new era of computing which has revolutionized the IT industry with its pay-as-you-go services [4]. Having the viable business prospect, all the tech-giants have already started providing cloud services. IT companies are now moving from traditional CAPEX model (buy the dedicated hardware and depreciate it over a period of time) to the OPEX model (use a shared cloud infrastructure and pay as one uses it). To enable and ensure the global growth of computing need, cloud service providing companies are now using warehouse sized data centers to meet user demand which incurs considerable amount of energy consumption. At the beginning of the cloud computing era, cloud service providers focused mainly on catering the computing demand that lead to expansion of cloud infrastructures; hence increased energy consumption. Therefore, energy consumption by data centers worldwide has risen by 56% from 2005 to 2010 [3]. So, to sustain the increasing need of computing, energy aware technique should be applied in cloud computing infrastructure otherwise the energy need will be huge and will be threatening to the environment. To handle this problem, datacenter resource needs to be utilized in an efficient manner. An efficient approach will not only reduce the energy consumption but also will keep the performance up to the mark. Both in hardware and software there are several techniques those are being used for energy consumption reduction of a cloud system and VM consolidation is considered as one of the most promising soft approach [1].

Virtualization is the key strength of cloud computing and one major advantage of this technology is live migration of Virtual Machine. For this reason, VM consolidation in cloud data center draws researcher's attention and is an active field of research in recent times. The main idea of VM consolidation is to keep minimum number of hosts active by consolidating VMs into active hosts and the remaining hosts which are not required are kept in sleep or inactive mode. As we know that inactive host or sleeping host causes minimal energy consumption; therefore, using this technique, energy consumption can be reduced considerably [5] . However, for VM consolidation we need to migrate VMs from one host to another which may lead to QoS degradation i.e. SLA (Service Level Agreement) violation. So, algorithms must be designed in such a way that not only it reduces power consumption but also serves desired QoS (such as SLA). There are two types of VM consolidation; 1. Static VM consolidation, and 2. Dynamic VM consolidation [2]. In the first approach the consolidation is done in a single step using the peak load demand of each workload and based on that, virtual machine capacities are configured. This approach is called static consolidation since the virtual machines stay in the same physical servers during their lifetime. The utilization of the peak load demand ensures that the virtual machine is not overloaded, however it can also lead to idleness since the workloads can present variable demand patterns. On the other hand, in dynamic VM consolidation(DVMC), reevaluation of workload demand is performed periodically and executing the required configuration changes in VMs. The second one usually results in better consolidation, since it dynamically changes virtual machine capacities according to the current workload demands. In real world, computation need is very dynamic and therefore, DVMC is the best fit. However,DVMC is more complex than the static one as it needs to deploy efficient algorithms to ensure the reevaluation and re-sizing of VM effectively with minimum time lags, otherwise, QoS is compromised.

II. Related Works

Considerable number of researches has been conducted for VM consolidation using various methods. In [3-5] Beloglazov et al. proposed heuristic based approach to deduce thresholds through different statistical measures. They divided the VM Consolidation problem into sub-problems (1. Overload detection, 2. VM selection from Overloaded host to migrate to new host, 3. Under load detection and put a host to sleeping mode and 4. VM placement). The authors proposed algorithms for each sub-problem. Algorithms are designed in such a way that they act, adapt and keep their threshold changing based on different scenario in different time so that they can still provide the functionality and consolidation decision for the changed environment. This adaption process allows the system to be dynamic. They designed threshold based (e.g. IQR) and prediction based (e.g. LR) host overload detection mechanisms. They treated VM placement problem as a bin packing problem and devised a Power-aware Best fit decreasing algorithm for efficient placement. They provided an effective framework for VM consolidation. However, there is no optimization phase for their VM placement and under-load detection algorithms.

References [6-7] describe CloudSim which provides various functionality of a cloud environment and facilitates cloud simulation. CloudSim allows to work with different sub problems and workload traces can be plugged to this simulator. Reference [3-5] have also used CloudSim for simulation. The main components of CloudSim are host, Virtual Machine (VM) and cloudlet. Cloudlet can be data from real cloud or can be randomly generated. The simulator creates hosts, Virtual Machine and cloudlet based on defined parameters. When the simulation starts, Virtual Machines are placed in the hosts for processing. To extend the VM consolidation , one needs to create new class and develop new methods for sub-problems. In this research we used CloudSim as simulator to develop and test our algorithms.

In [2], Ferreto et al. proposed VM consolidation with migration control where they treated the VM placement as bin packing problem and proposed heuristics for solution. In this research, VMs with steady usage are not migrated and non-steady VMs are migrated to ensure better performance. The migrations are triggered and done by heuristic approaches. But this research does not engage other sub problems rather focuses on only the VM placement problem. Farahnakian et al. [10] used ant colony system to deduce a VM placement solution for VM consolidation and in [21], they proposed prediction-aware VM placement method and developed it in CloudSim. The prediction-aware VM placement which is a modificaton of BFD algorithm does not rearrange the VMs in active hosts. in [20], Feller et al. also used ant colony based workload placement methods

Farahnakian et al. [22] proposed a Reinforcement Learning-based Dynamic Consolidation method (RL-DC) to minimize the number of active host considering the resource requirement. The RL-DC utilizes an agent that uses the past knowledge to take intelligent decision whether to keep the host in active or sleep mode and improves itself as the workload changes. It also dynamically adapts changes. In [12] linear regression has been used to predict CPU utilization by the same author. These researches are developed in CloudSim and followed the distributed architecture which provides opportunity to compare the results. However, the power-aware BFD algorithm they used for VM placement does not have optimization phase after allocation.

Cao et al. [13] proposed a redesigned energy-aware heuristic framework for VM consolidation to achieve a better energy-performance trade-off. They designed a Service Level Agreement (SLA) violation decision algorithm which is used to decide whether a host is overloaded with SLA violation or not. This research is based on CloudSim and used the Power-aware BFD VM placement approach. Mastroianni et al. [18] presented ecoCloud, a self-organizing and adaptive approach for the consolidation of VMs on CPU and RAM. Assignment and migration decisions are driven by probabilistic processes and based on local information. Focusing on the VM placement problem, they experimented in real datacenter. However, all the sub-problems are not addressed. Madani et al. [17] focused on an architecture configuration to manage virtual machines in a data center to optimize the consumption of energy and meet SLA by grafting a tracing component of multiple consolidation plans which ensure minimum number of servers is switched on. In this research, the problem is seen as scheduling problem and all sub problems are not discussed.

In [8], we worked with VM selection algorithm and introduced migration control based VM selection approaches and developed those in CloudSim. In [9], we worked on a modified Overload detection methods based on different heuristic method. In [15], we have introduced a new overload detection algorithm based on mean, median and standard deviation of utilization of VMs. In this research, we are focused on VM placement and SLA aware VM selection methods. Moreover we have redesigned the Under-load detection method.

III. Motivation

In our previous research work [8,9,15], we worked with Fuzzy with migration control based VM selection method and Overload detection method. In this research work our main focus is to work on the VM placement, Underload detection and QoS aware VM Selection method. The motivation factors which are driving this research are given below.

- VM placement plays one of the most important role in VM consolidation. An effective VM placement approach will ensure that VMs will be placed in such a way so that it will not change a sleeping host to active mode unless it is absolutely necessary. There should be an optimization phase which will try to rearrange the VMs between hosts so that activating a sleeping host becomes the last resort.
- The Underload detection phase identifies the host which can be put to sleeping mode by transferring all the VMs to active hosts. While placing the VMs from the underloaded hosts, optimization by rearranging is needed to increase the probability of turning a host into sleep mode.
- VM migration causes SLA violation. When a host is found overloaded, VMs which are selected for migration should be balanced. It can happen that some VMs are selected for migration more than the other VMs. This scenario will cause more SLA violation to those VMs. To balance this impact, VM selection methods need to balance the number of migration to all VMs.
- Consolidating VMs into host cause host overload hence SLA violation take place and on the other hand, migration of VMs also cause SLA violation. There should be a control mechanism based on SLA violation. SLA violation can be set to a certain value and the rest of the algorithms should be adapted so that no VM experience more SLA violation that the set value.

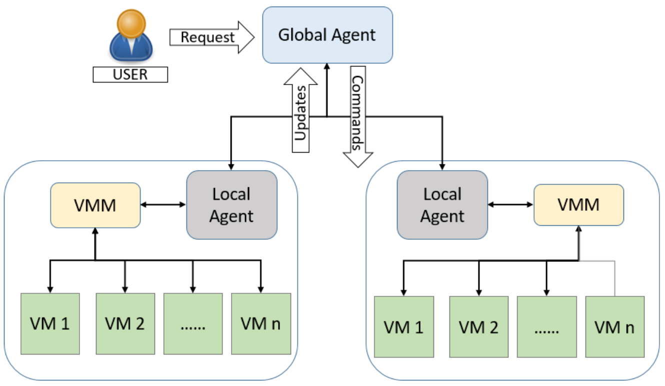

Fig. 1. System Architecture

IV. Proposed Methods

Dynamic Virtual Machine consolidation can be decomposed into several sub-problems[5]. The main advantage of decomposing the problem is, it gives us opportunity to devise separate algorithms for each sub problems.

A. Architecture and Flow

Our proposed cloud network architecture is depicted at Fig.1. The system has two major components, local agent(LA) and global agent(GA). The local agents reside on host side as a module of the Virtual Machine Monitor(VMM) or hypervisor. The objective of LA is continuous monitoring of the nodes utilization, re-sizing the VMs according to their resource needs, and send these information to global agent. The global agent resides on the master node and collects information from the local agents(LA) to maintain the overall view of the resources utilization. VM consolidation algorithm runs at global agent(GA) and it issues commands for the VMMs for VM migration.

The VM consolidation algorithm that runs at global agent is formulated at Algorithm 1. User puts workload on GA and GA instructs VMM to create VM and assign cloudlets. Based on dynamic consolidation technique, DVMC algorithm runs every scheduled interval which is represented by the while loop at step-2. For every scheduled interval, GA collects utilization and status information from LA at step-3 and prioritize VMs which crosses SLA violation threshold at step-4. Then overload detection is executed, and overloaded hosts are identified at step-5. To offload the overloaded hosts, at step-6, VMs are selected to migrate and this step is elaborated at Algorithm 5. Then at step-7, those VMs are placed into available hosts or if needed a host is switched on from sleeping mode and this step is described by Algorithm 2. Then at step-8 under-load detection algorithm is executed and less utilized hosts are put into sleeping mode by transferring all VMs to other active hosts. This step is elaborated at Algorithm 4. After each iteration QoS values are saved. At the end of the simulation, Energy consumption and QoS is shown. Now in the subsequent sections, each algorithms are detailed.

Algorithm 1: VM Consolidation

```
Data: Hosts, VMs
Result: QoS measurements
1  Assign VMs to Hosts;
2  while Run each DVMC interval do
3      Global Agent collects all data from Local Agents.
4      Priority assignment to VMs for QoS control
5      Overload detection
6      VM selection to selects VMs.
7      VM placement.
8      Underload detection and put hosts to sleep mode.
9      GA instructs the VMM to execute the changes.
10     Preserve history and calculate QoS.
11 end
12 Summarize the QoS parameters.
```

B. Pass-based BFD VM placement

In this segment we focus on the VM placement strategy. Since this is a bin packing problem, we have used best fit decreasing algorithm with a combination of an optimization layer. Making a sleeping host on and active has some cost and we have given least priority to wake a host from sleeping mode. In order to do that, we have used a pass based algorithm which will optimize the placement so that before waking up a host, the VMs are rearranged between the active hosts.

TABLE I
USED NOTATION

Description	Notation
Available CPU in Host H	A_{CPU}(H)
Available CPU in VM V	A_{CPU}(V)
Utilized CPU in Host H	U_{CPU}(H)
Utilized CPU in VM V	U_{CPU}(V)
Total CPU in Host H	T_{CPU}(H)
Total CPU in VM V	T_{CPU}(V)
Migration Map	M_{mig}(*H,VM*)
Overloaded Host	O_H
Sleeping Host	S_H
Migrating VM	v_M

Algorithm 2 represents the BFD VM placement algorithm. This algorithm starts with the knowledge of the overloaded hosts, sleeping hosts the and the VMs those need to be migrated. This VMs may can be from overloaded hosts or from the under-loaded host. This algorithm provides a migration map as a result. The migration map has two elements, destination host and the migrating VM. from steps 1-4, a host list is prepared where the overloaded hosts and sleeping hosts are excluded. These are the hosts where we will try to put VMs from the migrating VM list. At step-5 a loop is started which iterates for all the VMs in migrating VMs list and at step-6 a success variable is defined which will eventually provide a mark for successful placement. Now, the algorithm tries to find a suitable destination host for v_i. At step-8, for each host the Equation 1 is checked which ensures the host has enough resource to be the destination host of v_i. Here $A_{CPU}(h_i)$ indicates the available cpu of the host and $U_{CPU}(v_i$) indicates the needed cpu resource of the VM.

$$A_{CPU}(h_i) > U_{CPU}(v_i) \tag{1}$$

If the aforementioned condition satisfies then the VM is logically assigned to the this host. Logically assigning means the migration is not yet done but the the space is reserved in host. But this assignment is only done if the destination host is not overloaded by this assignment and this is checked at step-10. If the host is not overloaded by the assignment then it's a successful assignment and the migration map is updated with destination host and VM id. Now, if the the placement is not successful for the total loop of hosts then it means there is no viable space for this VM and then, optimize function is called at step-17 to rearrange the VMs in host to make room for the VM. The optimize algorithm is described in later portion. After the optimization, if the placement is unsuccessful then a sleeping host is activated and added to host list in steps 20-23. In later steps, the VM is assigned into the newly activated host.

Now we will discuss about the optimizer function which we have formulated as a Single Pass and Double optimizer in Algorithm 3. The optimize function is called with the list of host which excludes overloaded and sleeping host and another input is the the VM that needs to be migrated. As an outcome it provides a migration map. At first we will discuss about single pass which means the attempt to find space for a VM can be created by passing a smaller VM in new destination host. At step-3 a loop is started for all hosts. Now in steps 4-7, VMs of the host is taken in one list and from the list a VM is searched which satisfies Equation 2 and Equation 3. Equation 2 represnts that the space made available by removing VM v_i from host is enough to insert v_M. Equation 3 is making sure the new VM is smaller than the migrating VM.

Algorithm 2: Pass-based BFD VM placement

```
   Data: Hosts H, VMs V, Sleeping Host $S_H$, Overloaded
         Host $O_H$, VMs to migrate $V_{Mig}$
   Result: Migration Map $M_{mig}$(H,VM)
 1 $M_{mig}$ (H,VM) ← NULL
 2 $host_{excluded}$←$O_H$+$S_H$
 3 H ← H- $host_{excluded}$
 4 H ← Sort in decreasing order($H_{CPU}$)
 5 foreach $v_i$ in VM list $V_{Mig}$ do
 6     Success ← 0
 7     foreach $h_i$ in Host list H do
 8         if $A_{CPU}(h_i) > U_{CPU}(v_i$ ) then
 9             LogicalAssign $v_i$ to $h_i$ And Check if $h_i$
                overloaded or not
10             if !Overloaded($h_i$) then
11                 Success ← 1
12                 Update: $M_{mig}$ (H,VM)← ($h_i$, $v_i$ )
13             end
14         end
15     end
16     if Success == 0 then
17         Optimize ($v_i$ , H)
18     end
19     if Success == 0 then
20         $h_{sleeping}$ ← Assign($S_H$)
21         MakeActive($h_{sleeping}$)
22         Update H ← $h_{sleeping}$
23         H ← Sort in decreasing order($H_{CPU}$)
24         LogicalAssign $v_i$ to $h_{sleeping}$
25         Success ← 1
26         Update: $M_{mig}$ (H,VM)← ($h_{sleeping}$, $v_i$ )
27     end
28 end
29 return $M_{mig}$ (H,VM)
```

$$T_{CPU}(h_i) > U_{CPU}(h_i) - U_{CPU}(v_i) + U_{CPU}(v_M) \tag{2}$$

$$U_{CPU}(v_i) < U_{CPU}(v_M) \tag{3}$$

If no such VM is found then single pass is not possible and we start double pass immediately. If found, we continue to single pass at step-11. Then, we have made the problem smaller and we need to find a new host for v_i. Just like VM placement the loop iterates over hosts and tries to match Equation 1. If match found, VMs are logically assigned to destination hosts only if they are not overloaded by the assignments. In steps 18-20, success variable and migration map is updated for both VMs.

Now if single pass is unsuccessful then from step-25 the double pass begins. At step-26, the algorithm iterates over all the hosts and at step-28, it tries to find a VM which satisfies the double pass equations which are Equation 4, Equation 5 and Equation 6.

$$T_{CPU}(h_j) > U_{CPU}(h_j) - U_{CPU}(v_j) + U_{CPU}(v_i) \tag{4}$$

$$U_{CPU}(v_j) < U_{CPU}(v_i) \tag{5}$$

$$T_{CPU}(h_i) > U_{CPU}(h_i) - U_{CPU}(v_i) + U_{CPU}(v_i) + U_{CPU}(v_M \tag{6}$$

These Three equations ensures that there are two such hosts and VMs, by interchanging those VMs, we can make space for the the migrating VM (v_M). If such VMs are not found then the optimizer returns as unsuccessful in step-33. If found, the three VMs are assigned in two hosts and check for overload in steps 35-39. If the hosts are not overloaded then placement is declared successful and migration map is updated in steps 41-44 and map is returned. Interesting take-away from this algorithm is that, every time there is a success, it prevents a sleeping host from waking up. This ensures considerable gains. If the optimizer runs for a large number migrating VMs then the hosts are rearranged more efficiently.

C. Underload Detection

Underload detection is another important step in Algorithm 1, which is at step-8. When all the overloaded hosts are offloaded by transferring some VMs then comes the part of Underload detection. The main objective is search if there is any host that can be switched off by transferring all it's VMs to active hosts. We formulated this strategy in Algorithm 4. The required data for this methods are the active hosts and it will provide a migration map at the end. At the beginning at step-4, all the hosts are sorted in increasing order as the target is to put the low utilized host in sleeping mode. At step-5 a loop is started for every host. In the loop in steps 8-10, the algorithms calls the VM placement method and counts the number of successful placement. In steps 13-18, the algorithm checks if all the VMs of a host are successfully placed or not, if placed the host is set sleeping mode. If placement for any VM is unsuccessful the all the placement of the host is revoked. This is how the Underload detection works.

D. Overload Detection

Overload detection plays it's role at step-5 in Algorithm 1. We have worked with overload detection in our previous works [9][15]. Overload detection method finds out the host

Algorithm 3: SPDP Optimizer

```
Data: Hosts H, VM to migrate $v_M$
Result: Migration Map $M_{mig}$(H,VM)
1  Success ← 0
2  H ← Sort in decreasing order($H_{CPU}$)
3  foreach $h_i$ in Host list H do
4      $V_{iList}$ ← AllVms ($h_i$)
5      Find $V_i$ from $V_{iList}$ where,
6      $T_{CPU}(h_i) > U_{CPU}(h_i)$ - $U_{CPU}(v_i)$ + $U_{CPU}(v_M)$
7      AND $U_{CPU}(v_i) < U_{CPU}(v_M)$
8      if $V_i$ not found then
9          Break SinglePass and Start Doublepass
10     end
11     Begin Singlepass
12     foreach $h_j$ in Host list (H-$h_i$) do
13         if $A_{CPU}(h_j) > U_{CPU}(v_i)$ then
14             LogicalAssign $v_i$ to $h_j$
15             LogicalAssign $v_M$ to $h_i$
16             And Check if $h_i$ $h_j$ overloaded or not
17             if !Overloaded($h_i$) and !Overloaded($h_j$) then
18                 Success ← 1
19                 Update: $M_{mig}$ (H,VM)← ($h_i$, $v_M$ )
20                 Update: $M_{mig}$ (H,VM)← ($h_j$, $v_i$ )
21                 return $M_{mig}$ (H,VM)
22             end
23         end
24     end
25     Begin Doublepass
26     foreach $h_j$ in Host list (H-$h_i$) do
27         $V_{jList}$ ← AllVms ($h_j$)
28         Find $V_j$ from $V_{jList}$ where,
29         $T_{CPU}(h_j) > U_{CPU}(h_j)$ - $U_{CPU}(v_j)$ + $U_{CPU}(v_i)$
30         AND $U_{CPU}(v_j) < U_{CPU}(v_i)$
31         AND $T_{CPU}(h_i) > U_{CPU}(h_i)$ - $U_{CPU}(v_i)$ +
             $U_{CPU}(v_i)$ + $U_{CPU}(v_M)$
32         if $V_j$ not found then
33             return $M_{mig}$ (H,VM)
34         end
35         LogicalAssign $v_i$ to $h_j$
36         LogicalAssign $v_j$ to $h_i$
37         LogicalAssign $v_M$ to $h_i$
38         And Check if $h_i$ $h_j$ overloaded or not
39         if !Overloaded($h_i$) and !Overloaded($h_j$) then
40             Success ← 1
41             Update: $M_{mig}$ (H,VM)← ($h_i$, $v_M$ )
42             Update: $M_{mig}$ (H,VM)← ($h_j$, $v_i$ )
43             Update: $M_{mig}$ (H,VM)← ($h_i$, $v_j$ )
44             return $M_{mig}$ (H,VM)
45         end
46     end
47 end
48 return $M_{mig}$ (H,VM)
```

which are overloaded. We used a Mean, Median and Standard Deviation(MMSD) based Overload detection method.

Algorithm 4: Underload Detection

```
Data: Hosts $H$, VMs $V$, Sleeping Host $S_H$, Overloaded
      Host $O_H$
Result: Migration Map $M_{mig}$(H,VM)
1  $M_{mig}$ (H,VM) ← NULL
2  $host_{excluded}$ ← $O_H$ + $S_H$
3  H ← H- $host_{excluded}$
4  H ← Sort in increasing order($H_{CPU}$)
5  foreach $h_i$ in Host list H do
6      TotalSuccess ← 0
7      TotalNumVm ← NumOfVms($h_i$))
8      foreach $v_i$ in VM list $V_{Mig}$ do
9          $V_{Mig}$ ← $v_i$
10         Call VM Placement(H, $V_{Mig}$)
11         TotalSuccess ← TotalSuccess + Success
12     end
13     if TotalSuccess == TotalNumVm then
14         UnderloadSuccess($h_i$)
15         $S_H$ ← $h_i$
16     else
17         Underload Not Success
18         RevokeMap($M_{mig}$ (H,VM) upto TotalSuccess)
19     end
20 end
21 return $M_{mig}$ (H,VM)
```

E. SLA aware VM selection

VM selection is the step-6 in Algorithm 1. The VM consoldiation algorthim identifes the overloaded hosts, then VM selection method is deployed to select a VM, which should be migrated from overloaded host to offload it. The input of a VM seleciton method is the list of overloaded host and output is the list of VMs those should be migrated to offload the overloaded hosts. We have worked on VM selection methods in our previous works [8][15]. We designed fuzzy logic based VM selection. But those research works were not focused on SLA. While providing cloud services, it is expected that no particular VM is facing more SLA violation than the others. For example in [5] one VM selection method is designed as minimum migration time which selects a VM from the overloaded host that has the smallest amount of memory associated with it. It makes the transfer faster and hence reducing sla violation. But if we deploy this strategy then it is possible that low memory occupying VMs face more SLA violaiton due to excessive migration. For this reason, a SLA aware VM selection method is necessary. When a VM is in migration, the VM faces SLA voilation since at the time of migration the VM can not serve user needs. The Equation 11 provides the formula for SLA violation for migration (SLAM). If C_{d_j} is degradation for migration for a particular VM j and C_{r_j} is total requested CPU then the SLA violation for VM j can be determined by the Equation 11.

$$SLAM_j = \frac{C_{d_j}}{C_{r_j}} \tag{7}$$

$$v\,\epsilon\,V_j\,|\,\forall\,a\,\epsilon V_j\,,SLAM_a \geq SLAM_v \tag{8}$$

Equation 10. defines that if V_j is the set of all VMs in a host then VM v is selected for migration if the SLA violation of v is less then all other VMs a. This strategy is implemented in Algorithm 5. In the SLA aware VM selection algorithm the input is the overloaded hosts and in step-2 a loop is started to iterate over all the overloaded hosts. For each VM it is checked if the VM has high priority or not and in steps 7-12 minimum SLA violated VMs separately saved for high priority and low priority. If there are low priority VMs in the host then the minimum SLA violating low priority VM is selected for migration. If there are only high priority VMs then the lowest SLA violation VM is selected for migration. After selection, the VM is logically removed from host and tests whether the host is still overloaded or not. If the host is still overloaded then another iteration for VM selection takes place and if host is not overloaded then, next host is selected for offloading. In this way a list of migrating VMs is created from the overloaded hosts.

Algorithm 5: SLA aware VM selection

```
Data: Overloaded Host $O_H$
Result: VMs to migrate $V_{Mig}$
1  $V_{Mig}$ ← NULL
2  foreach $h_i$ in Overloaded Host list $O_H$ do
3      while Overloaded($h_i$) do
4          $VM_{list}$ ← VmList($h_i$ )
5          foreach $v_i$ $\epsilon$ VM list $VM_{list}$ do
6              $SLA(v_i)$ ← MeasureCurrentSLA($v_i$)
7              if $v_i$.Priority > 0 then
8                  CountThrViolated()
9                  UpdateMinSlaViolated($v_{minthrv}$)
10             else
11                 CountNonViolated()
12                 UpdateMinSlaViolated($v_{minnonv}$)
13             end
14         end
15         if CountThrViolated equals total counf of VMs
             then
16             $VM_{ToMigrate}$ ← $v_{minthrv}$
17         else
18             $VM_{ToMigrate}$ ← $v_{minthrv}$
19         end
20         Update: VMs to migrate $V_{Mig}$ ← $VM_{ToMigrate}$
21         LogicalRemove $VM_{ToMigrate}$ from $h_i$
22     end
23 end
24 return Priority(V)
```

F. Priority-based SLA violation control

Priority-based SLA violation control is the step-6 in Algorithm 1. Here, our concern is to set a threshold on SLA violation so that the algorithm restricts the SLA violation for any VM. We propose a priority-based SLA violation control which will be executed in every consolidation cycle and set a priority based on the SLA violation status. When a VM will reach upto certain percentage of SLA violation threshold then maximum priority will be assigned so that this VM is not be migrated and moreover the VM will be re-sized to it's maximum size. So in the subsequent cycle SLA violation value will decreased. But the priority will not be set to lowest immediately rather it will be decreased periodically to ensure the VM is far from crossing SLA violation threshold. This is implemented in Algorithm 6. Here in step-4, 80% of threshold is considered as the trigger point and at step-11 when the priority is reset to 0 then the utilized cpu is made normal. In this way SLA violation constrain is imposed.

```
Algorithm 6: Priority-based SLA violation control
  Data: VMs V, SLA Limit Threshold
  Result: VMs Priority Priority(V)
1 $VM_{list} \leftarrow$ AllVms($V$)
2 foreach $v_i \in$ VM list $VM_{list}$ do
3     $SLA(v_i) \leftarrow$ MeasureCurrentSLA($v_i$)
4     if $SLA(v_i) >= 0.8 \times (Threshold)$ then
5         Priority($v_i$) $\leftarrow$ Priority 1
6         Set CPU to Max: $U_{CPU}(v_i) \leftarrow T_{CPU}(v_i)$
7     else
8         Priority($v_i$) $\leftarrow$ Priority - 0.1
9         if Priority($v_i$) $<=0$ then
10            Priority($v_i$) $\leftarrow$ Priority 0
11            Set CPU to Normal: $U_{CPU}(v_i) \leftarrow U_{CPU}(v_i)$
12        end
13    end
14 end
15 return Priority(V)
```

V. Experiment Setup

In this experiment, we have implemented our algorithms in CloudSim 3.0.3 to analyze performance and behaviour of our proposed methods.

A. Physical nodes and VM configuration

For the experiment of real cloud traces, we have considered 800 heterogeneous nodes, half of which are HP ProLiant G4 and the rest are HP ProLiant G5 servers. For Random load we have considered 200 hosts. Energy consumption is calculated based on HP ProLiant G4 and HP ProLiant G5 CPU usage and power consumption that is represented in Table II [5]. These servers are assigned with 1860MIPS (Million instruction per second) and 2660 MIPS for each core of G4 and G5 servers respectively. Network bandwidth is considered as 1GB/s. The VMs which were created were single core. VM were of 4 types, for example, 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).

TABLE II
POWER CONSUMPTION FOR DIFFERENT LEVEL OF UTILIZATION

Machine Type	0%	20%	40%	60%	80%	100%
HP G4 (Watt)	86	92.6	99	106	112	117
HP G5 (Watt)	93.7	101	93.7	121	129	135

TABLE III
DAY WISE PLANETLAB DATA

Date	Number of VM	Mean
3/3/2011	1052	12.31%
6/3/2011	898	11.44%
9/3/2011	1061	10.70%
22/03/2011	1516	9.26%
25/03/2011	1078	10.56%
3/4/2011	1463	12.39%
9/4/2011	1358	11.12%
11/4/2011	1233	11.56%
12/4/2011	1054	11.54%
20/04/2011	1033	10.43%

B. PlanetLab trace data

In this work we have used real world work load data that is provided from CoMon project, a monitoring infrastructure for PlanetLab [16]. This data is collected from more than thousand VMs of different servers that are located in 500 different locations. The workload is representative of an IaaS cloud environment such as Amazon EC2, where VMs are created and managed by several independent users. Table III presents the day wise VM number for this data. These real world traces contain VM utilization records in every 5-minute interval. 10 days data of year 2011 have been used in this experiment. Each VM contains 288 (=24*(5/60)) data of CPU utilization. The simulation checks CPU data every 5 minute interval and those trace data is plugged into dynamic VM consolidation.

C. Performance measurement metrics

The main target of VM consolidation is to reduce energy consumption and at the same time the QoS should be at an acceptable level. The energy consumption metric is discussed below and for QoS parameter, several metrics are stated which are used in different researches [4,5]. The main QoS is SLA violation. In VM consolidation, SLA violation occurs due to host overload and VM migration. Moreover, the number of VM migration indicates the efficiency of the consolidation method which is also described as a metric. But the main objective of our research is to obtain Energy-QoS trade off and that is defined by the metric ESV which is the product of energy consumption and SLA violation (SLAV). So the method providing the lowest ESV and at the same, the lowest energy consumption and the lowest SLA violation, is undoubtedly the best method. Based on these six metrics proposed method will be verified and they are described below.

1) Energy Consumption: This is the main metric as the target of VM consolidation is to reduce energy consumption. Energy consumption is computed by taking all hosts into account throughout the simulation and by mapping CPU value, energy consumption deduced from Table II. At each iteration, the CPU utilization is measured and power consumption is calculated from Table II. And at the end of the simulation energy consumption is measured by accumulating energy consumption all hosts. From Equation 9, power is a function of utilization.

$$E = \int P(U(t))dt \tag{9}$$

2) SLA violation: Service level agreement violation, SLAV, can take place for two reasons, 1. SLA violation due to overloaded host(SLAO) and 2. SLA violation for migration(SLAM). And SLA violation SLAV, is the combined impact of SLAO and SLAV. Equation 10,11 and 12 provides the calculation criteria for SLA violation.

$$SLAO = \frac{1}{M}\sum_{i=1}^{M}\frac{T_{si}}{T_{ai}} \tag{10}$$

$$SLAM = \frac{1}{N}\sum_{j=1}^{N}\frac{C_{d_j}}{C_{r_j}} \tag{11}$$

$$SLAV = SLAO * SLAM \tag{12}$$

At Equation 10, SLAO provides a measure of the fraction of time a host experienced 100% CPU utilization leading to SLA violation. Here N is the number of hosts, T_si is the total time when host i experienced 100% utilization leading SLA Violation, T_ai is the total active time of host i. At Equation 11, SLAM provides the measure of the total SLA violation due to VM migration. When a host is overloaded, VMs are migrated from that host to non-overloaded host. At the time of migration, it causes SLA violation. This metric calculates the SLA violation caused by migration. Here M is the number of total VMs adn C_{dj} stands for the CPU request at the time of migration of VM j and C_{rj} stands for total CPU requested by VM j.

3) ESV: Energy consumption and SLA are already defined. It is perceivable that if we try to reduce too much energy consumption the SLA violation will be increased, because consolidating many VMs in a host will increase the probability of overload. So it is desirable to obtain a method which will consume less power and still incur less SLA violation. To measure this, ESV is introduced in Equation 13. It is the combination of Energy consumption and SLA violation. So this can be treated as one metric to make an overall judgment.

$$ESV = Energy * SLAV \tag{13}$$

4) Number of Host Shutdown: This metric counts the number of host shutdown. Its a measure of defining how efficiently VM consolidation is working. If we have poor consolidation algorithm that will make the hosts on and off in great numbers which means the algorithms are not efficient.

5) Number of VM migration: This metric counts the number of VMs migrated during the simulation. VM migration is an important factor because unnecessary migration causes SLA violation and increases network traffic.

VI. Experimental Resutls

We have implemented our algorithms in CloudSim. To provide a fair comparison, we have carried out our experiments with real cloud workload and also with randomly generated workload data. We have also compared for different sizes of network and with different traffic scenarios.

A. Comparison Benchmark

Every VM consolidation has four sub problems, 1. Overload Detection, 2. VM selection, 3. VM placement and 4. Underload detection. From different literature we have three VM selection approaches, 1. Minimum Migration time policy(MMT) [5] which selects a VM with minimum migration time, 2. Maximum Correlation(MC) [5,19] which selects a VM which has highest correlation and 3. Random Selection (RS) [4], where VMs are selected randomly. Moreover, in this research, we proposed one VM Selection approach which is SLA aware VM Selection. There are several Overload detection approaches, 1. Threshold Based Overload detection(THR) [4,5], it uses threshold to declare a host is overloaded, 2. Regression based overload detection (LR, LRR) where regression is used predict the host load [4, 12] 3. Interquartile range method (IQR) where quartile measures are used to declare a host is overloaded[5]. 4. Median Absolute Deviation [4,5] is used to determine a host is overloaded. From our previous research there is one overload detection method, MMSD,[15] which uses mean, median and standard deviation for overload detection. For VM placement, there are power-aware best fit decreasing algorithm proposed in [4,5,23] and there is underload detection algorithm in [4,5] without the optimization phase. Since we have not worked on overload detection in this research we will combine other proposed overload detection method with our algorithms. So we will compare with five Overload detection algorithms (IQR, LR, LRR, MAD and THR) and three VM selection (MC, MMT, RS) methods. So in combination there are 15 methods (IQR_MC, IQR_MMT, IQR_RS, LR_MC, LR_MMT, LR_RS, LRR_MC, LRR_MMT, LRR_RS, MAD_MC, MAD_MMT, MAD_RS, THR_MC, THR_MMT, THR_RS). With each combination, there are power-aware best fit decreasing method as VM placement and underload detection from [4,5] are be used to make them a complete VM consolidation packages and a set of benchmarks to compare with. On the other hand, we have Pass-based VM placement with optimization as VM placement algorithm, SLA-aware VM selection and our underload detection algorithm. We will combine our method with five Overload detection algorithms (IQR, LR, LRR,

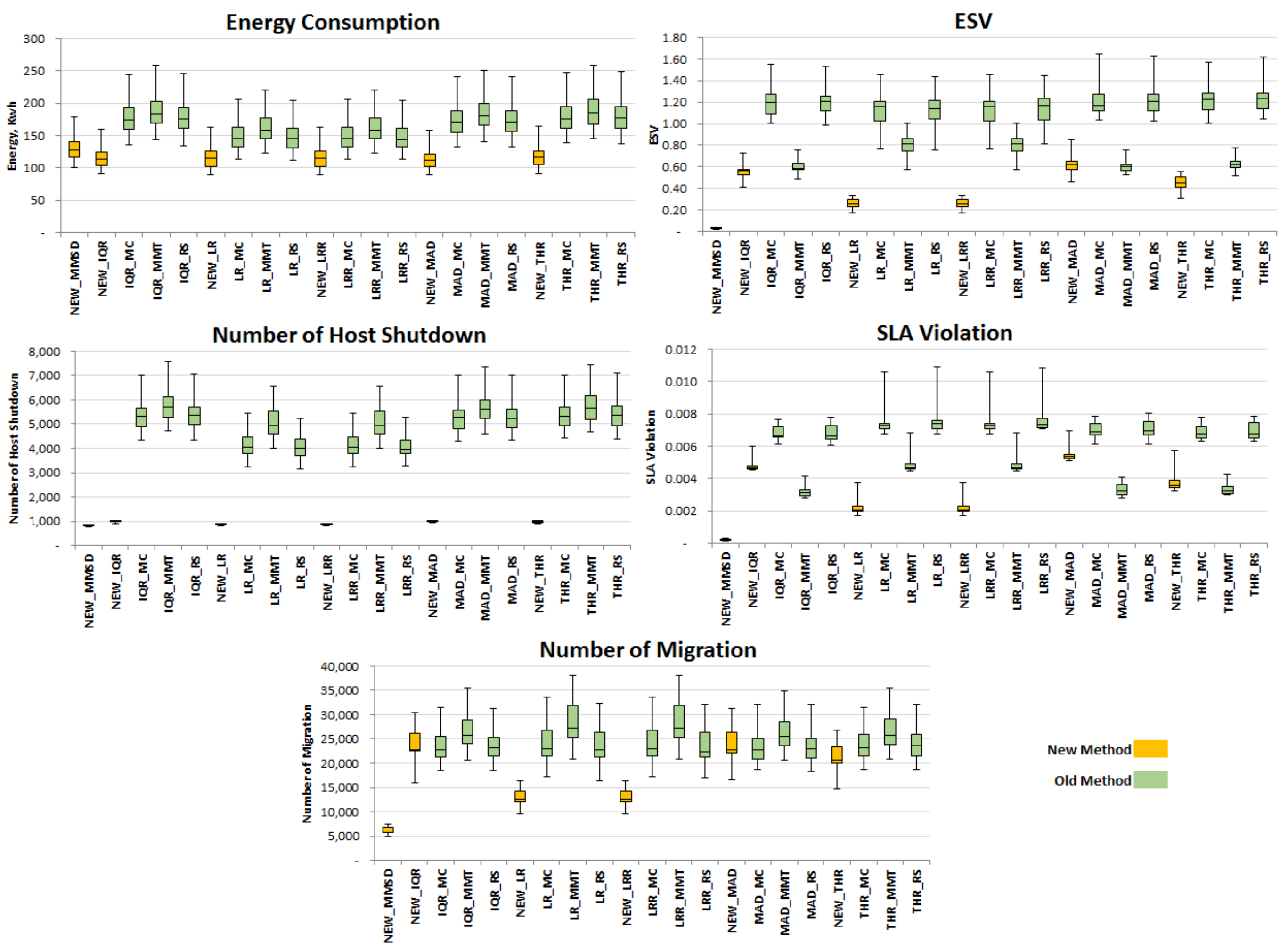

Fig. 2. Energy, ESV, Number of host shutdown, SLA, and Number of migration for planetlab traces

MAD and THR) from other research and with our MMSD algorithm. So we we will have six complete set of VM consolidation approaches (NEW_MMSD, NEW_IQR, NEW_THR, NEW_LR, NEW_LRR and NEW_MAD) against 15 complete set of consolidation approaches from other research. So We will compare NEW_IQR with IQR_MC, IQR_MMT, IQR_RS methods and NEW_MMSD method having all new approaches integrated and these can be compared to any method of 15 methods.

B. *Result with PlanetLab traces*

We have 10 days of PlanetLab data which we simulate for our 6 yellow colored consolidation approaches and 15 other consolidation approaches which are colored green. Based on the result for 10 days Box plots have been prepared to compare the results in Fig.2. In this experiment we have used 800 hosts and more than 1000 VMs. We have compared our results for five metrics.

1) Energy Consumption: Main objective of this research is to design a VM consolidation algorithm so that the energy consumption is reduced. By comparing the proposed and existing methods in the Fig.2, it is found that energy consumption is significantly reduced in all six combinations in comparison to their counterparts and as well as in total. Therefore, we can infer, the basic objective of this research is achieved by saving energy consumption.

2) SLA Violation: SLA violation is one of the key indicators of QoS. SLA Violation is calculated by keeping two scenarios into consideration, i) SLA violation for overload, and ii) The SLA violation incurred for migration. A method having low SLA violation ensures the desired QoS. From Fig.2, SLA violation is decreased in three methods NEW_MMSD, NEW_LR and NEW_LRR. Remaining three approaches are threshold based overload detection methods and the improved ones are prediction based methods. It means, the overload detection method we have used, predicted the overloaded host efficiently and as an outcome, SLA violation has dropped significantly. If host overload is predicted successfully then there will be less number of migration which will reduce SLA violation like a cycle.

3) ESV: As energy consumption has been successfully reduced by the proposed method, now energy-QoS trade off

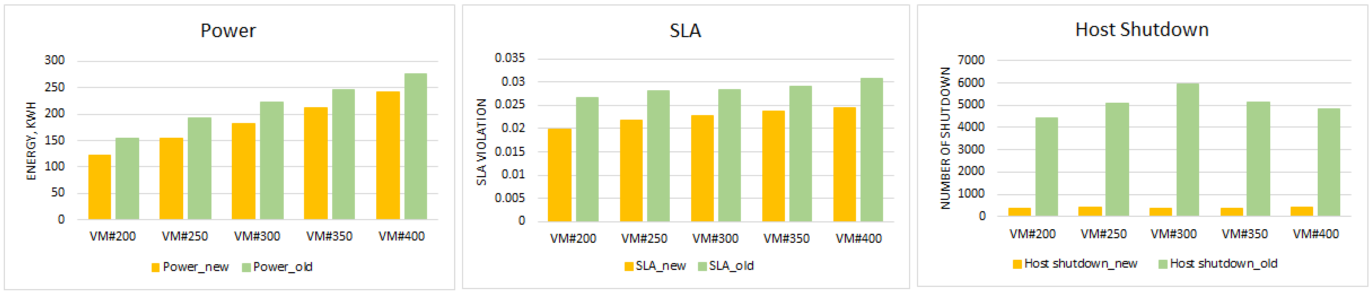

Fig. 3. Power, SLA and Host shutdown with Random traces

needs to be checked. ESV is the metric which is a product of Energy consumption and SLA violation; hence, provides a trade-off visibility of the proposed method with other existing methods. From previous two sub-sections, we have observed that both energy and SLA violation is reduced for most of the proposed methods, so it is inevitable that ESV will also be reduced. From the Fig.2, ESV is found to be reduced which is clearly visible for all six new methods. If ESV reduces it means that this approach saves energy and at the same time SLA violation is controlled. As ESV is reduced significantly, it means that Energy and SLA trade-off has been achieved.

4) Number of Host Shutdown: From Fig.2, it is easy to observe that the number of host shutdown in the new methods reduced significant. That means when a host is put to sleep mode it stays there for long time hence proves the efficiency of our algorithms.

5) Number of Migration: Less number of VM migrations means efficient consolidation, less traffic in cloud network and less SLA violation for VM migration. Reduction in Number of VM migration is also visible from Fig.2 for three new methods which is just an echo of the SLAV graph. Again it is proven that, prediction based overload detection works better. However our proposed solution beats all other. From this results it is evident that the proposed method provides better VM consolidation compared to the existing VM consolidation approaches.

C. Result random traces

For PlanetLab real workload data it has been proven that our methods outperforms other VM Consolidation strategies. Now we want to compare the performance for randomly generated load and we also want to see the performance when we increase the number VMs in the same size network. For this section we are using a network of 200 hosts and there are five different number of VMs allocated to it (200, 250, 300, 350, 400). For this experiment the Old method is considered IQR_MC method and the New Method is NEW_IQR with new VM placement, VM selection and Underload detection algorithm. The result is given at Fig.3.

1) Energy Consumption for Random load: From Fig.3, power graph, we observe the energy consumption is increasing with the number of VMs. This is logical as more VMs will keep more host active and active hosts consumes power. But the main take away is that, even if the power consumption is increasing the new methods is outperforming the old method for every size of the VM population.

2) SLA for Random load: From Fig.3, we observe that the SLA violation is increasing with the number of VMs in a slow rate. However for all number of VMs, the new method has less SLA violation from old one which just supports previous graphs of PlanetLab data.

3) Number of Host shutdown: From Fig.3, we observe the the number of Host shutdown is increasing in the beginning and is decreasing when the number of VMs reaches near 400. However for every number of VMs the host shutdown has a high difference where the new method is outperforming the old one by large margin.

Considering the results of PlanetLab load and the result of Random load we can infer that, these algorithms will perform better in different kinds of load with respect to the old methods.

D. A deep dive

From the above analysis, we have found that the proposed methods are sowing better results in all metrics and for some metrics new methods are showing significant improvement. Drastic improvement found for SLA violation when prediction based overload methods are used and also significant improvement observed for the number of host shutdown which provides testimony of an efficient VM consolidation. Now we want to investigate what is the reason for this improvement and what is happening in root level

For this experiment, we have used 200 hosts and 200 VMs and random load. We have generated two heat maps for the new method and the old method and that is presented at Fig.4. For the old method, IQR_MC is used and for the new method NEW_IQR is used. So the Overload detection method is same but methods for other three sub problems are different. In the heat map, if a host is in sleeping mode i.e., 0% utilization, then it is marked by blue color and red color is for highest utilization. In the X-axis time is portrayed. As iteration

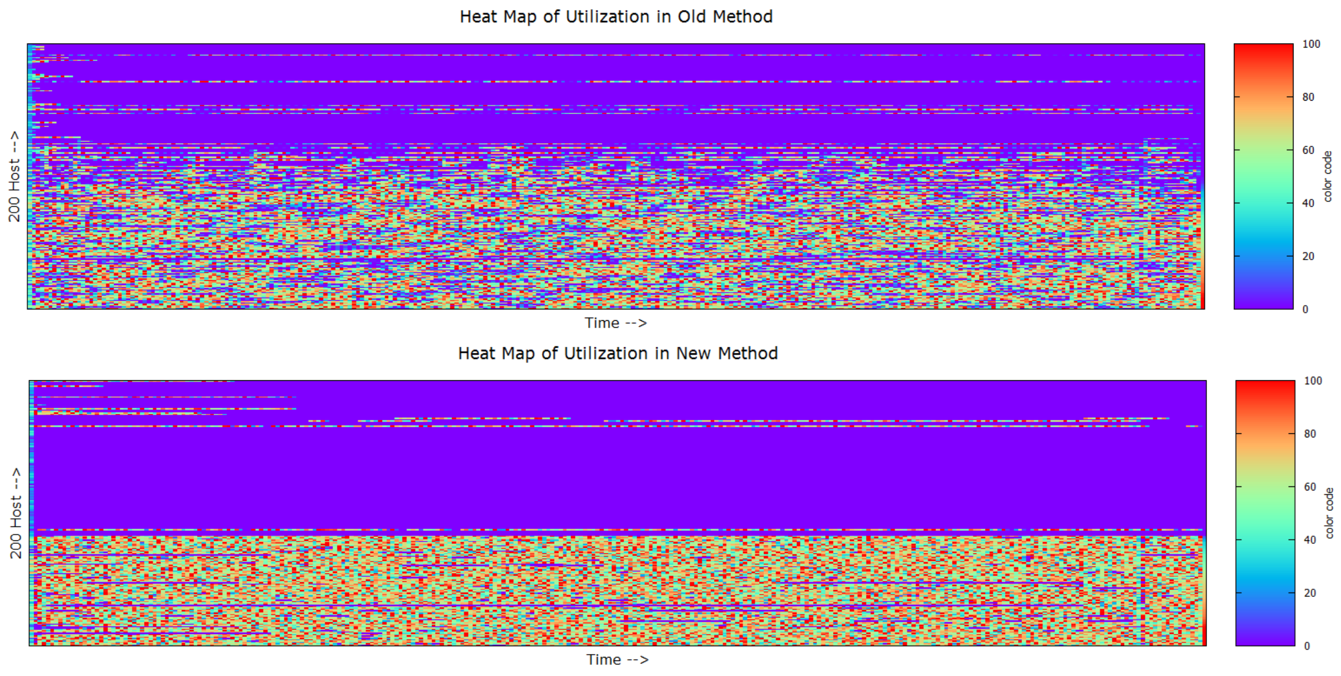

Fig. 4. Heat map of old and new method

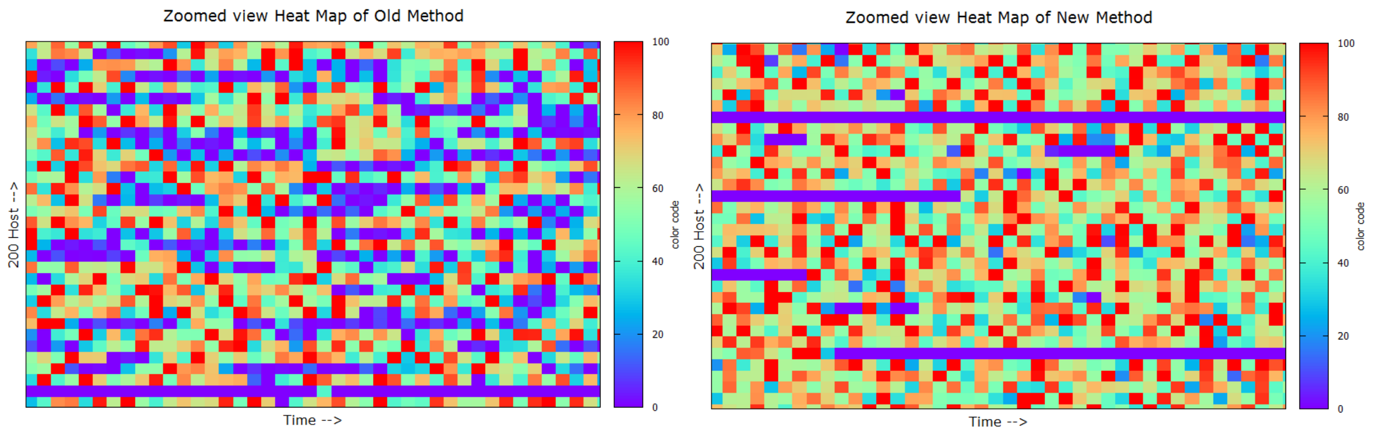

Fig. 5. Zoomed view of Heat maps

duration is 5 minutes, there are total 288(starting from 0 sec to 8600 sec) iterations are there since the simulation is done for one-day data. Y-axis represents 200 hosts. Each horizontal line in the graph represents a host in different time. From Fig.4, it is visible that for the old method, hosts are used in a scattered fasione and occupying most of the graph. On the other hand, the new method shows the heat map not much scattered and the hosts which are active are remaining active and the hosts are in sleeping mode are still in sleeping. Overall we can observe a consistency in new method.

We have seen the bigger picture, now we will zoom in to get more insight. The Fig.5 shows some zoomed portion of Fig.4. The left map depicts the zoomed view of the old method whereas the right map is portraying the new method. From the zoomed map of the old method shows that the hosts re experiencing frequent ON/OFF scenario. This means, a host is kept in sleep mode and waked up again for VM placement and this keeps going for all over the simulation and resulting high number of host shutdown value, more SLA violation and more energy consumption. On the other hand, the right map shows completely different picture. The new method shows that when a host is in active mode it stays in active mode for longer time and when a host is put to sleeping mode than it stays in sleeping mode for long. For these reasons, the number of host shutdown, SLA violation and energy consumption is lower than the old method.We can deduce that main reason behind this consistent performance is the VM placement and Underload detection method with optimization phase.

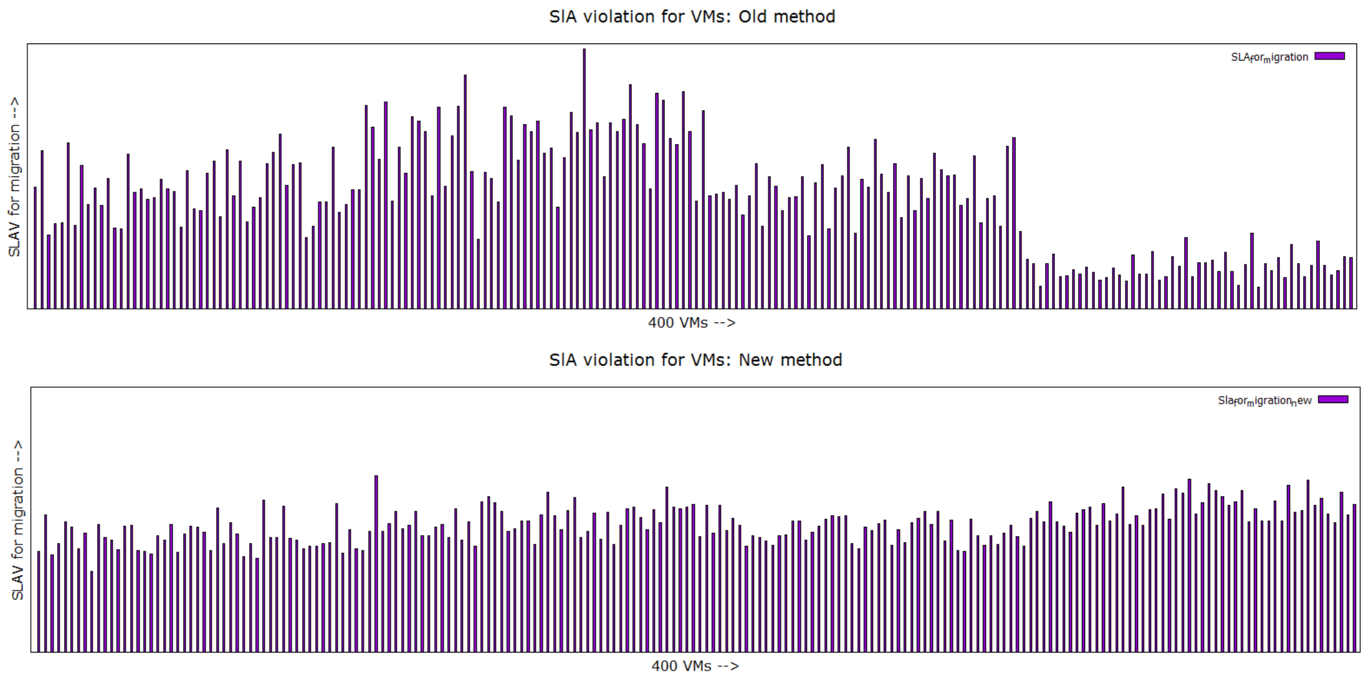

Fig. 6. SLA violation control

E. *SLA Aware VM selection method*

The objective for this section is to design a SLA aware VM selection method which will distribute the SLA violation among the VMs in such way so that the no VMs suffer from extra SLA violation while some VMs are not affected. For this reason we compare the Maximum correlation VM selection method which the old method with our SLA aware VM selection method. For this experiment we have chosen a network size of 200 hosts and 400 VMs and we used random load.

From Fig.6, the graph for old method shows different range of SLA violation for different VMs. In some portion the SLA violation is very high and in some portion the SLA violation is very small. So for this VM selection approach different user will face different SLA violation regardless the CPU load that the user is generating. On the other hand, the graph for new methods shows a different picture where all the VMs shows consistent behavior in terms of SLA violation and it seems the SLA violation due to migration is distributed by All. Hence objective of the SLA aware VM selection method is fulfilled.

VII. Conclusion

In this research we have devised an algorithm for pass-based VM Placement method using best fit decreasing strategy and introduced an extra phase which will optimize the placement further. The optimizer algorithm improves VM placement using single and double pass method. Then we designed Underload detection algorithm which also uses the VM placement and the SPDP optimizer function to improve it's performance. Then we have devised algorithms for SLA aware VM selection mechanism to ensure control on SLA violation so that no VM suffer from high SLA violation where others do not. After simulation and making comparison with existing methods, it has been found that the proposed methods outperformed other previous methods in both perspectives, i.e., more energy saving and less SLA violation. Therefore, it can be inferred that the objective, energy-SLA tradeoff has been achieved in this work in an efficient manner. As a future work we have plan to improve the overload detection mechanism and develop in CloudSim to achieve more energy saving and less SLA violation.

References

[1] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, *A taxonomy and survey of energy-efficient data centers and Cloud computing systems*, Advances in Computers, M. Zelkowitz (ed.), vol. 82, pp. 47111, 2011.

[2] T. C. Ferreto , M. A. S. Netto , R. N. Calheiros and C. A. F. De Rose, *Server consolidation with migration control for virtualized data centers*, Future Generation Computer Systems, v.27 n.8, pp.1027-1034, 2011.

[3] 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 (FGCS), vol. 28, no. 5, pp. 755768, 2011.

[4] 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 (CCPE), vol. 24, no. 13, pp. 13971420, 2012.

[5] A. A. Beloglazov, *PhD Thesis: Energy-Efficient Management of Virtual Machines in Data Centers for Cloud Computing*, 2013. Link: http://beloglazov.info/thesis.pdf.

[6] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. 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. 2350, 2011.

[7] CloudSim link:, http://code.google.com/p/CloudSim/

[8] M. A. H. Monil, R. Qasim and R. M. Rahman, *Energy-aware VM consolidation approach using combination of heuristics and migration control*, 9th IEEE International Conference on Digital Information Management, pp. 74-79, 2014.

[9] M. A. H. Monil and R. M. Rahman, *Implementation of modified overload detection technique with VM selection strategies based on heuristics and migration control*, IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS), pp. 74-79, 2015.

[10] F. Farahnakian, A. Ashraf, P. Liljeberg, T. Pahikkala, J. Plosila, I. Porres and H. Tenhunen,, *Energy-Aware Dynamic VM Consolidation in Cloud Data Centers Using Ant Colony System*, IEEE 7th International Conference on Cloud Computing (CLOUD), pp. 104 - 111, 2014.

[11] S. Di, D. Kondo, W. Cirne, *Host load prediction in a Google compute cloud with a Bayesian model*, International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Salt Lake City, UT, Nov. pp. 10-16, 2012.

[12] F. Farahnakian, P. Liljeberg, and J. Plosila, *LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers*, 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), pp. 357364, 2013.

[13] Z. Cao and S Dong, *Energy-Aware framework for virtual vachine vonsolidation in cloud computing*, IEEE 10th International Conference on High Performance Computing and Communications and 2013 IEEE International Conference on Embedded and Ubiquitous Computing, pp. 1890 1895, 2013.

[14] G. S. Akula and A. Potluri, *Heuristics for migration with consolidation of ensembles of Virtual Machines*, Communication Systems and Networks (COMSNETS), pp. 1 4, 2014.

[15] M. A. H. Monil and Rashedur M Rahman, *Fuzzy Logic Based Energy Aware VM Consolidation*, 8th International Conference on Internet and Distributed Computing Systems (IDCS 2015), Windsor, U.K., September 2-4, pp.223-227, 2015.

[16] K. S. Park and V. S. Pai, *CoMon: a mostly-scalable monitoring system for Planet- Lab*, ACM SIGOPS Operating Systems Review, vol. 40, no. 1, pp. 6574, 2006.

[17] N. Madani, A. Lebbat, S. Tallal and H. Medromi, *New cloud consolidation architecture for electrical energy consumption management*, AFRICON, pp. 1 3, 2013.

[18] C. Mastroianni, M. Meo adn G. Papuzzo, *Probabilistic Consolidation of Virtual Machines in Self-Organizing Cloud Data Centers*, IEEE Transactions on Cloud Computing,(Volume:1 , Issue: 2), July-December 2013, pp. 215 228, 2013.

[19] 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 USENIX Annual Technical Conference, pp. 2828, 2009.

[20] E. Feller, L. Rilling and C. Morin, *Energy-aware ant colony based workload placement in clouds, International Conference on Grid Computing*, International Conference on Grid Computing, 2011, pp.26-33.

[21] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila and H. Tenhunen, *Utilization Prediction Aware VM Consolidation Approach for Green Cloud Computing*, 2015 IEEE 8th International Conference on Cloud Computing, Athens, pp.381 - 388, 2015.

[22] F. Farahnakian, P. Liljeberg and J. Plosila, *Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning*, Parallel, Distributed and Network-Based Processing (PDP), pp. 500 507, 2014.