QoS-aware Virtual Machine Consolidation in Cloud Datacenter

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

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

***Abstract*—With the rapid growth of the cloud industry in recent years, energy consumption of warehouse-scale datacenters 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 a bin packing problem which is NP-hard, hence, it is logical to apply a 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 multi-pass optimization in VM placement for an efficient VM consolidation. We have used CloudSim toolkit to simulate our experiments. To evaluate the performance of the proposed algorithms we used real-world workload traces from thousand VMs. Results demonstrate that our proposed methods outperform other existing methods.**

***Index Terms*—Virtual Machine Consolidation; VM placement; SLA; CloudSim**

I. Introduction

Cloud computing has revolutionized the IT industry with its pay-as-you-go services [4]. Many cloud datacenters are now being built to cater to greater computing needs. Concern for energy-efficient cloud infrastructure has risen worldwide because cloud datacenters consume a considerable amount of energy. Researchers have proposed several techniques for reduction of energy consumption of cloud datacenters and Virtual Machine (VM) consolidation is considered one of the most promising ones [1]. Virtualization is the key strength of cloud computing and one major advantage of this technology is the live migration of the VMs. For this reason, VM consolidation in cloud datacenters is an active field of research. The main idea of VM consolidation is to keep a minimal number of hosts active by consolidating VMs into active hosts, and the remaining unrequired hosts are kept in sleep or inactive mode. Since inactive or sleeping hosts cause minimal energy consumption, this technique reduces energy consumption considerably [5]. In the real world, the computational need is quite dynamic and therefore it needs to deploy efficient algorithms to ensure the reevaluation and resizing of VMs effectively with minimum time lags; otherwise, Quality of Service (QoS) is compromised. Algorithms need to ensure a balance between energy consumption and QoS.

In this research, we have divided the Dynamic Virtual Machine Consolidation (DVMC) problem into sub-problems and focused mostly on designing an effective VM placement approach. We propose that a VM placement algorithm should 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 and there should be an optimization phase which will rearrange the VMs between hosts so that activating a sleeping host becomes the last resort. Having this motivation, we have devised an efficient multi-pass BFD VM placement algorithm which ensures better energy-QoS balance compared to other research.

II. Related Works

Since VM consolidation is an active field of research, many researchers proposed techniques to design an effective DVMC system. In [3-5] Beloglazov et al. proposed VM consolidation algorithms based on different statistical measures. They treated the VM placement problem as a bin packing problem and devised a Power-aware Best fit decreasing algorithm for efficient placement. However, there is no optimization phase for their VM placement algorithm. In [2], Ferreto et al. VMs which consistently experience similar CPU usage pattern are not migrated and non-steady VMs are migrated to ensure better performance. This research does not engage all subproblems, rather focusing on only the VM placement problem. Farahnakian et al. [10] used an ant colony system to deduce a VM placement solution for VM consolidation and in [20], they proposed prediction-aware VM placement method and developed it in CloudSim. The prediction-aware VM placement which is a modification of the BFD algorithm does not rearrange the VMs in active hosts. In [19], Feller et al. also used ant colony based workload placement methods. Reference [6-7] describe CloudSim, an open-source cloud simulator which provides the opportunity to carry out experiments in a cloud environment of varying sizes and traffic cases. Mastroianni et al. [17] presented ecoCloud where decisions of assignments and migrations are made by using probabilistic processes and local information. However, all the sub-problems of DVMC are not addressed. Farahnakian et al. [21] used a Reinforcement Learning method(RL-DC) to design a VM consolidation algorithms. In [12], linear regression has been used to predict CPU utilization by the same author. These algorithms were developed in CloudSim and followed the distributed architecture which provides opportunities to

978-1-5090-5817-4/17 $31.00 © 2017 IEEE
DOI 10.1109/IC2E.2017.31

compare the results. However, the power-aware BFD algorithm they used for VM placement does not have an optimization phase after allocation is completed. Cao et al. [13] proposed an energy-aware framework for VM consolidation to achieve a better energy-performance trade-off. This research is based on CloudSim and used the Power-aware BFD VM placement approach.

In [8], we worked with VM selection algorithms and introduced migration control based VM selection approaches. In [9], we worked on the Overload detection algorithm and we modified cloud framework in CloudSim using different heuristic methods. In [15], we have introduced a new overload detection algorithm(MMSD) based on the mean, median and standard deviation of utilization of VMs. In this research, we focus on VM placement in continuation of designing a complete VM consolidation solution.

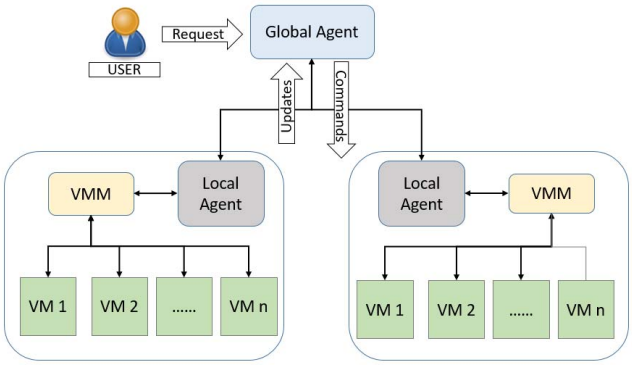

Fig. 1. System Architecture

III. 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 the opportunity to devise separate algorithms for each subproblem.

A. Architecture and Flow

A 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 the host side as a module of the Virtual Machine Monitor (VMM) or hypervisor. The objective of the LA is continuous monitoring of the nodes utilization. The global agent resides on the master node and collects information from the local agents (LA). The VM consolidation algorithm runs at a global agent (GA), and it issues commands for the VMMs for VM migration.

The VM consolidation algorithm is formulated at Algorithm 1. Users put the workload on GA and GA instructs VMM to create VM and assign cloudlets. The DVMC algorithm runs every scheduled interval which is represented step-2. GA collects utilization and status information from LA and prioritizes VMs which crosses SLA violation threshold. Then overload detection is executed to identify overloaded hosts.

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.

To offload the overloaded hosts, VMs are selected to migrate. Then at step-7, those VMs are placed into available hosts, or if needed a host is switched on from the sleeping mode. Then at step-8 the underload detection algorithm is executed and less utilized hosts are put into the sleeping mode by transferring all VMs to other active hosts. After each iteration, QoS values are saved. At the end of the simulation, Energy consumption, and QoS are shown. Now in the subsequent sections, each algorithm is detailed.

B. Multi-pass BFD VM placement

In this segment, we focus on the VM placement strategy. Since this is a bin packing problem, we have used the 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. To do that, we have used a multi-pass algorithm which will optimize the placement so that before waking up a host, the VMs are rearranged between the active hosts.

Algorithm 2 represents the Multi-pass BFD VM placement algorithm. This algorithm starts with the knowledge of the overloaded hosts, sleeping hosts and the VMs that need to be migrated. This VMs may be from overloaded hosts or the under-loaded host. This algorithm provides a migration map as a result. The migration map has two elements, the 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 on the migrating VM list. At step-5 a loop is started which iterates for all the VMs in the 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 available CPU is checked to ensure the host has enough resources 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.

Algorithm 2: Multi-pass 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 **if** *Success == 0* **then**
15 Optimize (v_i , *H*)
16 **if** *Success == 0* **then**
17 $h_{sleeping}$ ← Assign(S_H)
18 MakeActive($h_{sleeping}$)
19 Update *H* ← $h_{sleeping}$
20 *H* ← *Sort in decreasing order*(H_{CPU})
21 LogicalAssign v_i to $h_{sleeping}$
22 Success ← 1
23 Update: M_{mig} *(H,VM)*← ($h_{sleeping}$, v_i)
24 **end**
25 **return** M_{mig} *(H,VM)*

If the aforementioned condition satisfies then, the VM is logically assigned to this host. But this assignment is only done if this assignment does not overload the destination host and this is checked at step-10. For a successful assignment, the migration map is updated with the destination host and VM id. Now, if the placement is not successful for all the hosts then it means there is no viable space for this VM and then the optimize function is called at step-17 to rearrange the VMs in the host to make room for the VM. After the optimization, if the placement is unsuccessful then a sleeping host is activated and added to host list in steps 20-23. In subsequent steps, the VM is assigned to the newly activated host.

Now we will discuss the optimizer function which we have formulated as a Single and a Double pass optimizer in Algorithm 3. The optimizer function is called with the list of the hosts and the VM to be migrated. Optimize function can be called during every VM placement attempt but this is computationally expensive. Rather we have used the optimizer function only when a normal allocation is not possible. At first, we will discuss single pass which means the attempt to find space for a VM can be created by moving a smaller VM to a new destination host. After starting, in steps 4-7, VMs of the host are taken in one list and from the list, a VM is searched which satisfies Equation 1 and Equation 2. Here T stands for total CPU, U stands for current CPU utilization and A stands for currently available CPU. Equation 1 represents that the space made available by removing VM v_i from the host is enough to insert v_M. Equation 2 is making sure the new VM is smaller than the migrating VM.

$$T_{CPU}(h_i) > U_{CPU}(h_i) - U_{CPU}(v_i) + U_{CPU}(v_M) \quad (1)$$

$$U_{CPU}(v_i) < U_{CPU}(v_M) \quad (2)$$

If no such VM is found then the single pass is not possible and we start double pass immediately. If found, we continue to single pass at step-10. 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 **??**. If a match is found, VMs are logically assigned to destination hosts only if they are not overloaded by the assignments. In steps 16-18, success variable and migration map are updated for both VMs.

$$T_{CPU}(h_j) > U_{CPU}(h_j) - U_{CPU}(v_j) + U_{CPU}(v_i) \quad (3)$$

$$U_{CPU}(v_j) < U_{CPU}(v_i) \quad (4)$$

$$T_{CPU}(h_i) > U_{CPU}(h_i) - U_{CPU}(v_i) + U_{CPU}(v_i) + U_{CPU}(v_M) \quad (5)$$

Now if the single pass is unsuccessful then from step-20 the double pass begins. At step-21, the algorithm iterates over all the hosts and at step-23, it tries to find a VM which satisfies the double pass equations which are Equation 3, Equation 4 and Equation 5.

These three equations ensure that there are two such hosts and VMs, by interchanging those VMs, we can make space for the migrating VM (v_M). If such VMs are not found then the optimizer returns as unsuccessful in step-27. If found, the three VMs are assigned to two hosts and check for overload in steps 29-30. If the hosts are not overloaded then placement is declared successful, and the migration map is updated in steps 32-34 and the map is returned. An interesting takeaway 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 and Overload Detection

Underload detection is another step in Algorithm 1, which is at step-8. The main objective is to search if there is any host that can be switched off by transferring all its VMs to active hosts. All the hosts are sorted in increasing order and using the VM placement algorithm, if all the VMs of a host can be migrated to active hosts then the host is switched to sleep mode otherwise keep as it is. Overload detection plays its role at step-5 in Algorithm 1. We have worked with overload detection in our previous works [9][15]. The Overload detection method finds the hosts which are overloaded. We used a Mean, Median and Standard Deviation (MMSD) based Overload detection method.

Algorithm 3: SPDP Optimizer

Data: Hosts *H*, VM to migrate v_M
Result: Migration Map M_{mig}*(H,VM)*

Success ← 0
H ← *Sort in decreasing order*(H_{CPU})
foreach h_i *in Host list H* **do**
  V_{iList} ← AllVms (h_i)
  Find V_i from V_{iList} where,
  $T_{CPU}(h_i) > U_{CPU}(h_i)$ - $U_{CPU}(v_i)$ + $U_{CPU}(v_M)$
  AND $U_{CPU}(v_i) < U_{CPU}(v_M)$
  if V_i *not found* **then**
    Break SinglePass and Start Doublepass
  Begin Singlepass
  foreach h_j *in Host list (H-*h_i*)* **do**
    if $A_{CPU}(h_j) > U_{CPU}(v_i)$ **then**
      LogicalAssign v_i to h_j and v_M to h_i
      And Check if h_i h_j overloaded or not
      if *!Overloaded(*h_i*) and !Overloaded(*h_j*)* **then**
        Success ← 1
        Update: M_{mig} *(H,VM)*← (h_i, v_M) and (h_j, v_i)
        return M_{mig} *(H,VM)*
  end
  Begin Doublepass
  foreach h_j *in Host list (H-*h_i*)* **do**
    V_{jList} ← AllVms (h_j)
    Find V_j from V_{jList} where,
    $T_{CPU}(h_j) > U_{CPU}(h_j)$ - $U_{CPU}(v_j)$ + $U_{CPU}(v_i)$
    AND $U_{CPU}(v_j) < U_{CPU}(v_i)$
    AND $T_{CPU}(h_i) > U_{CPU}(h_i)$ - $U_{CPU}(v_i)$ + $U_{CPU}(v_i)$ + $U_{CPU}(v_M)$
    if V_j *not found* **then**
      return M_{mig} *(H,VM)*
    LogicalAssign v_i to h_j, v_j to h_i and v_M to h_i
    And Check if h_i h_j overloaded or not
    if *!Overloaded(*h_i*) and !Overloaded(*h_j*)* **then**
      Success ← 1
      Update: M_{mig} *(H,VM)*← (h_i, v_M), (h_j, v_i) and (h_i, v_j)
      return M_{mig} *(H,VM)*
  end
end
return M_{mig} *(H,VM)*

D. VM selection

VM selection is the step-6 in Algorithm 1. The VM consolidation algorithm identifies the overloaded hosts, and then the VM selection method is deployed to select a VM, which should be migrated from the overloaded host to offload it. We worked on VM selection methods in our previous works [8][15]. VM selection is done based on migration time, correlation and utilization steadiness of the VMs.

TABLE I
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 II
DAY WISE PLANETLAB DATA

Date	Number of VM	Mean Utilization
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%

IV. EXPERIMENT SETUP

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

A. Physical nodes and VM configuration

We have used two types of nodes which are HP ProLiant G4 and HP ProLiant G5 servers. For the real workload traces, we have chosen a network of 800 hosts. Energy consumption is calculated based on CPU usage of the nodes which is represented in Table I [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. More than 1000 VMs are considered in the simulation which are 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).

B. PlanetLab trace data

In this research, we have used real world workload traces provided by the CoMon project, a monitoring infrastructure for PlanetLab [16]. This data is taken from a cloud network which is geographically distributed in 500 different locations. Table II presents the day wise number of VM number and mean CPU utilization of those VMs. These traces contain VM utilization records of every 5-minute interval and for this reason, we have set the simulation iteration 5-minutes.

C. Performance measurement metrics

Several QoS metrics are used in different studies [4,5] and it is possible to implement measurement metrics in CloudSim. Based on five metrics our proposed method will be verified and they are described below.

1) Energy Consumption: Energy consumption is computed from the total cloud network that is used in the simulation. By mapping CPU value of every host, the energy consumption of a host is deduced from Table I. At each iteration, energy consumption is recorded and at the end of the simulation, total consumption is measured by accumulating energy consumption from all hosts.

2) SLA violation: Service level agreement violation can take place for two reasons, 1. SLA violation due to overloaded hosts (SLAO) and 2. SLA violation for migrations (SLAM). SLA violation (SLAV) is the combined impact of SLAO and SLAM. Equation 6 and 7 provides the calculation criteria for SLA violation.

$$SLAO = \frac{1}{M}\sum_{i=1}^{M}\frac{T_{si}}{T_{ai}} \tag{6}$$

$$SLAM = \frac{1}{N}\sum_{j=1}^{N}\frac{C_{d_j}}{C_{r_j}} \tag{7}$$

At Equation 6, SLAO provides a measure of the fraction of time a host experienced 100% CPU utilization leading to SLA violation. Here M is the number of hosts, T_{si} is the total time when host i experienced 100% utilization leading to SLA Violation, T_{ai} is the total active time of host i. At Equation 7, SLAM provides the measure of the total SLA violation due to VM migration. At the time of migration, it causes SLA violation. Here N is the number of total VMs and 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: One of the objectives of this research is to design a technique which will consume less power and still incur less SLA violation. To measure this, ESV is introduced which is the product of Energy consumption and SLA violation.

4) Number of Host Shutdowns: This metric counts the number of host shutdowns. A poor consolidation algorithm will make the hosts turn on and off in great numbers.

5) Number of VM migrations: Migration of a VM means SLA violation and additional network traffic. It is desirable to have a method which will efficiently organize all the VMs in the active hosts using least number of migrations.

V. Experimental Resutls

To provide a fair comparison, we have carried out our experiments with different combinations of VM consolidation algorithms.

A. Comparison Benchmark

VM consolidation has four subproblems, 1. Overload Detection, 2. VM selection, 3. VM placement and 4. Underload detection. From the 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) [5,18] which selects a VM that has the highest correlation and 3. Random Selection (RS) [4], where VMs are selected randomly. There are several Overload detection strategies, 1. Threshold Based Overload detection (THR) [4,5], it uses a threshold to declare a host is overloaded, 2. Regression-based overload detection (LR, LRR) where regression is used to 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. So there are 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, power-aware best fit decreasing VM placement methods [4,5,23] and underload detection method from [4,5] are used to make them 15 complete VM consolidation packages. These 15 combinations are considered as a set of benchmarks to compare with.

On the other hand, we have multi-pass VM placement, VM selection method from our previous work and underload detection from reference [4]. We will combine these methods with five Overload detection algorithms (IQR, LR, LRR, MAD and THR) from other research and with overload detection algorithm from our previous work (MMSD) [15]. So we have six complete sets of VM consolidation approaches (NEW_MMSD, NEW_IQR, NEW_THR, NEW_LR, NEW_LRR and NEW_MAD) against 15 complete sets of consolidation approaches from other research.

B. Result with PlanetLab traces

We have ten 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 of 10 days, box plots have been prepared to compare the results in Fig.2. We have compared our results using five metrics.

1) Energy Consumption: The 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 Fig.2, it is evident 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.

2) SLA Violation: SLA violation is one of the key indicators of QoS. SLA Violation is calculated by taking 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. The 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.

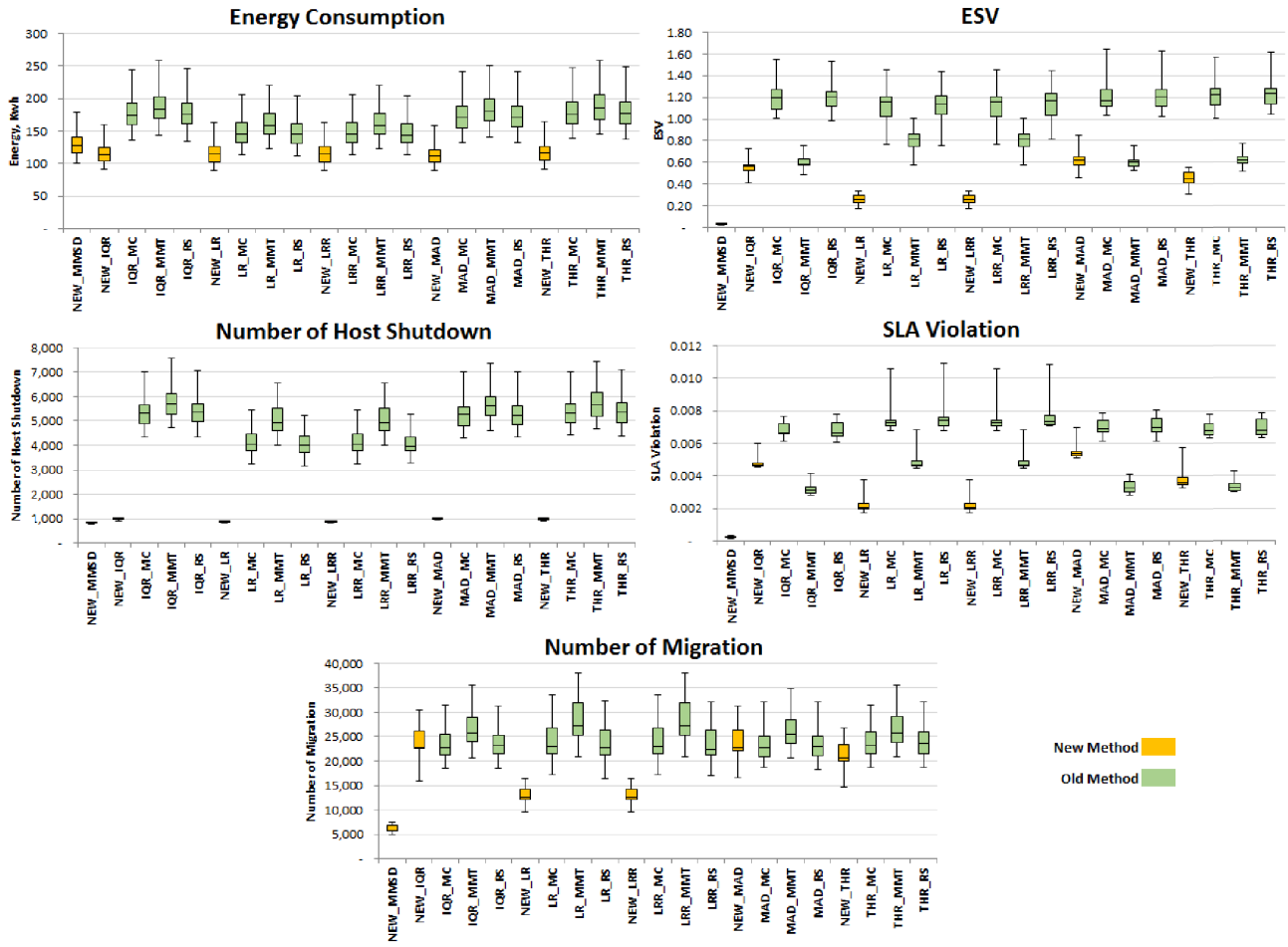

Fig. 2. Energy, ESV, Number of host shutdown, SLA, and Number of migration for planetlab traces

If host overload is predicted successfully then there will be fewer migrations which will reduce SLA violation.

3) ESV: ESV is the metric which is a product of Energy consumption and SLA violation; hence it provides a trade-off visibility of the proposed method with other existing methods. From the previous two sub-sections, we have observed that both energy and SLA violation are reduced for most of the proposed methods, so it is inevitable that ESV will also be reduced. From Fig.2, ESV is found to be reduced which is clearly visible for all six new methods. If ESV decreases, 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 Shutdowns: From Fig.2, it is easy to observe that the number of host shutdowns in the new methods reduced significantly. That means when a host is put into sleep mode it stays there for a long time and hence proves the efficiency of our algorithms.

5) Number of Migrations: Fewer VM migrations mean efficient consolidation, less traffic in the cloud network and less SLA violation for VM migration. Reduction in the Number of VM migrations is also visible from Fig.2 for three new methods which are just an echo of the SLAV graph. Again it is proven that prediction based overload detection works better. From this result, it is evident that the proposed method provides better VM consolidation compared to the existing VM consolidation approaches.

VI. Conclusion

In this research, we have devised an algorithm for a multi-pass VM Placement method using a Best Fit Decreasing strategy and introduced an additional phase which will optimize the placement further. The SPDP optimizer algorithm improves VM placement using a multi-pass method. After simulation and making a 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.

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. 47-111, 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. 755-768, 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. 1397-1420, 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. Last Access Date:31-July-2016.

[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. 23-50, 2011.

[7] CloudSim link:, http://www.cloudbus.org/cloudsim/ Last Access Date:31-July-2016.

[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. 357-364, 2013.

[13] Z. Cao and S Dong, *Energy-Aware framework for virtual machine consolidation 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. 65-74, 2006.

[17] C. Mastroianni, M. Meo and 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.

[18] 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. 28-42, 2009.

[19] 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, pp. 26-33, 2011,

[20] 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.

[21] 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.