Energy Efficient Virtual Machine Placement in Cloud Data Centers Using Modified Intelligent Water Drop Algorithm

Chandra Shekhar Verma
University of Hyderabad
Hyderabad, Telangana, India
chandra1em@gmail.com

V. Dinesh Reddy
Institute for Development and Research in Banking Technology
Hyderabad, Telangana, India
dineshvemula@gmail.com

G.R. Gangadharan
Institute for Development and Research in Banking Technology
Hyderabad, Telangana, India
grgangadharan@idrbt.ac.in

Atul Negi
University of Hyderabad
Hyderabad, Telangana, India
atulcs@uohyd.ernet.in

***Abstract*—Cloud Computing is an emerging distributed computing paradigm for the dynamic provisioning of computing services on demand over the internet. Due to heavy demand of various IT services over the cloud, energy consumption by data centers is growing significantly worldwide. The intense use of data centers leads to high energy consumptions, excessive CO_2 emission and increase in the operating cost of the data centers. Although many virtual machine (VM) placement approaches have been proposed to improve the resource utilization and energy efficiency, most of these works assume a homogeneous environment in the data centers. However, the physical server configurations in heterogeneous data centers lead to varying energy consumption characteristics. In this paper, we model and implement a modified Intelligent Water Drop algorithm (MIWD) algorithm for dynamic provisioning of virtual machines on hosts in homogeneous and heterogeneous environments such that total energy consumption of a data center in cloud computing environment can be minimized. Experimental results indicate that our proposed MIWD algorithm is giving superior results.**

I. Introduction

Cloud Computing is a new distributed computing paradigm for dynamic provisioning of computing services on demand over the internet. Cloud computing is evolving as everything as a service on the internet. Cloud providers utilize one or more data centers to host cloud services and resources. These data centers can be distributed geographically to safeguard data availability. The growth in hyper-scale cloud data centers is one of the major contributors for the increase in electricity consumption across the globe.

In the year 2014, US data centers consumed nearly 70 billion kWh of power which is approximately 2 percent of total power consumption of US [1]. Data Center Usage Report given by the U.S. Department of Energy stated that there is 4 percent increase in energy consumption of data center from 2010 to 2014. It is expected to continue with 4 percent increase during 2014-2020, and US data centers are projected to consume approximately 73 billion kWh of power in 2020 [1].

Efficient use of energy in data centers is a challenging task. The 2010 energy efficiency scenario [1] show‘ s that there were nearly 100 billion kWh of energy saving from 2010-2014 and 520 billion kWh of energy saving is expected with current trends from 2015-2020. The major part of these energy savings comes from servers and infrastructures. Further, the energy consumption in data centers can be reduced by efficient use of servers in data centers, that can be achieved by efficiently placing virtual machines on servers and switching off the unused servers.

Green cloud computing requires energy efficient use of data centers with minimum impact on environment [2]. There are several green metrics [3] defined to achieve greenness in data centers. A system is considered green when it consumes minimum energy to another system with the same goal. Green computing in the cloud can be achieved by implementing efficient policies and methods for energy aware resource management in virtualized data centers [4], [5].

The importance of dynamic VM placement is intensified by the high power consumption of data centers. The placement of Virtual Machines over physical hosts became a vital component of any cloud management framework. While provisioning resources to VMs, it is important to maximize resource utilization by using minimum number of physical hosts. This work is focused on energy savings by efficient utilization of the servers by effective placement of virtual machines (VMs) in cloud computing and switching off idle servers. We develop and discuss an efficient algorithm for virtual machine placement based on Intelligent Water Drop (IWD) algorithm to minimize power consumption of a cloud data center.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes our proposed

978-1-5386-4283-2/17 $31.00 © 2017 IEEE
DOI 10.1109/SITIS.2017.14

Table I
PROPOSED METHODS FOR VM PLACEMENT

Ref.	Problem Dealt	Methods Used	Energy Considered ?
[6]	VM placement and Migration	Modified Best Fit Decreasing, Minimum migration time (MMT)	Yes
[7]	VM placement	Mixed integer programing (MIP)	No
[8]	VM Migration	Single Threshold, highest potential growth and random choice policy	Yes
[9]	VM placement	Dynamic programing	Yes
[10]	VM consolidation	Virtual power management	Yes
[11]	VM placement	MMT, Random Choice (RC), Maximum correlation, Power aware best fit decreasing	Yes
[12]	VM placement	pMapper	No
[13]	VM placement	Two tier Cluster and Cut algorithm	No
[14]	VM placement	vGreen	Yes
[15]	VM placement	EAGLE	Yes
[16]	VM placement	Forecast based power aware best fit decreasing	Yes
[17]	VM placement	Hybrid Genetic Algorithm	No
[18]	VM consolidation	Simulated Annealing	No
[19]	VM placement and Migration	Minimum power VM placement (MinPow) and minimum virtual machine communication (MinCom)	Yes
[20]	VM placement	EnaCloud	Yes
[21]	VM placement	Best Fit with hierarchical clustering	Yes
[22]	VM consolidation	Binpacking and Timeseries forecasting	Yes
[23]	VM consolidation	PowerExpand MinMax	Yes

algorithm for energy aware VM placement. In Section 4, we discuss the performance analysis of our proposed algorithms, followed by concluding remarks in Section 5.

II. RELATED WORK

The seamless proliferation of internet users and digitization make the requirements for more data centers throughout the world. This requires high processing capacity and high storage capacity. According to Lawrence Berkeley National Laboratory (LBNL), the power consumption of data center was 70 billion kWh in 2014 which was 1.8% of total US energy consumption [2]. Several researchers focused on energy efficient operation of data centers.

Beloglazov et al. [6] proposed a threshold based modified best fit decreasing algorithm for virtual machines (VM) provisioning on hosts. This algorithm heuristically uses varying threshold level for CPU utilization. It performs live migration of VMs using minimum migration time method. However, dynamic consolidation may lead to more SLA violation. Wang et al. [7] proposed a mixed integer programming (MIP) approach to solve virtual machine placement problem and proposed a solution for linear and nonlinear power consumption models. To find the near optimal solution, they used a heuristic based iterative rounding technique. However, this takes more time with increasing virtual machines and servers.

Buyya et al. [8] proposed a non-power aware (NPA) policy and DVFS policies for energy saving. They proposed various policies such as single threshold (ST), Minimum migration (MM), Highest Potential growth (HPG) and random choice policies. For VM migration they illustrate that ST policy and MM policy consume nearly same energy, but MM policy shows very less VM migrations compared to ST policy. Goudarzi et al. [9] used dynamic programming to create multiple copies of VMs and to put them on servers and then used local search to find underutilized servers. Creating multiple copies of virtual machines favors that the smallest virtual machine will be fully utilized by the servers and minimizes the possibility of underutilized servers. In this approach, only the original VM serves the request while all other copies will remain ideal. However, SLA violation is not considered, and multiple copies of VMs create network overhead.

Nathuji et al. [10] proposed a Virtual Power Management (VPM) infrastructure framework which uses hardware based power scaling and software based approaches to control power consumption. This approach may lead to performance penalties due to application specific VM consolidation. For Virtual Machine Placement, Power Aware Best Fit Decreasing (PABFD) algorithm is proposed in [6], [11]. However, for strict SLA, this approach consumes more energy.

Verma et al. [12] proposed a power-aware pMapper framework, which works as an application placement controller and dynamically places the application to minimize power and migration cost while meeting the performance criteria. Meng et al. [13] defined the traffic aware virtual machine placement problem (TVMPP) and proposed a heuristic based two tier Cluster-and-Cut algorithm for TVMPP to improve the network scalability. This approach forms the clusters of VMs and hosts and then maps the VMs and hosts, at cluster level and individual level.

Dhiman et al. [14] proposed a vGreen software system for energy efficient VM scheduling on heterogeneous hosts. This model did not consider SLA violation. Li et al. [15] proposed an EAGLE algorithm for VM placement problem considering the efficient utilization of servers. This model has not considered SLA and VM migration. Dong et al. [16] proposed a Forecast based PABFD (F1PABFD) model that uses different forecast models for saving the energy. Tang et al. [17] proposed a hybrid genetic algorithm for VM placement considering network overhead. But it has not considered energy consumption by VM migration.

Wu et al. [18] proposed a Simulated Annealing (SA) based VM Placement algorithm It is suitable for static VM consolidation but not for dynamic VM consolidation. Dai et al. [19] proposed a minimum power (MinPow) and the minimum communication (MinCom) algorithms for energy efficient placement and migration of virtual machines, based on integer programming. However, it did not consider SLA violation. Li et. al. [20] proposed an EnaCloud architecture, which considers live application placement approach for saving the energy. But it has not considered multiple resources (CPU, memory, etc.) for energy savings.

Dong et al. [21] proposed a Best Fit with hierarchical clustering algorithm (BF-HC), based on the minimum-cut algorithm. But, authors have not considered the case of overload and network congestion in the network. Bobroff et al. [22] proposed a dynamic VM consolidation algorithm. They used time series forecasting and bin-packing heuristic for minimizing the active physical resources. However, multiple resources are not considered in this approach. Cardosa et al. [23] proposed a PowerExpandMinMax algorithm, for VM consolidation based on min-max and share features of VM technologies. This algorithm does not follow the restrict resource constraints and live VM migration. Table 2.1 summarizes the various methods proposed for VM placement in cloud computing.

To overcome the said problems, we propose a modified IWD (MIWD) algorithm for energy efficient VM placement in cloud data centers.

III. Proposed Work

In this section, we model virtual machine scheduling as an optimization problem and propose a modified IWD (MIWD) algorithm to solve energy efficient VM placement in cloud data centers. Efficient virtual machine placement increases the resource utilization and saves the energy in a cloud data center. Our aim is to find an optimal virtual machine placement solution that reduces the total energy consumption of a data center without violating SLAs.

A. Virtual Machine Scheduling Model

The process of placing the virtual machines onto each of the host (or physical servers) is known as virtual machine placement problem. For a given set of physical machines (hosts) and for a given set of the virtual machines (VMs), we describe the mathematical modeling of virtual machine placement problem as follows.

Let a set of physical machines be denoted as H and a set of virtual machines be denoted as V. Then

$H = \{H_1, H_2, H_3.....H_n\}$ where n is total number of hosts,

$V = \{V_1, V_2, V_3....., V_m\}$ where m is total number of VMs.

To deploy the set of VMs V to the physical machines H, we define a solution set S as follows.

$$S = \{S_1, S_2, S_3.....S_n\}$$

where S_i is the i^{th} solution set and represents a set of virtual machines to be placed on to a host.

If there are M virtual machines (VMs) and H servers (hosts), then the total number of ways to place VMs on Host is equal to M^H [24], [25]. This gives a polynomial time solution and it is not possible to enumerate all the solutions practically. Hence, we develop an algorithm to find an optimal solution in a reasonable time.

Our objective is to minimize the energy consumption by all the servers.

$$\text{Objective Function: } f = min \sum_{i=1}^{n} E_i \tag{1}$$

where E_i is energy consumption by the i^{th} server and each solution must satisfy the following constraints:

$$\sum_{i=1}^{H} r_i^{MIPS} \cdot v_{ij} < H_j^{MIPS} \tag{2}$$

$$\sum_{i=1}^{H} r_i^{mem} \cdot v_{ij} < H_j^{mem} \tag{3}$$

and

$$\sum_{j=1}^{n} v_{ij} = 1 \tag{4}$$

where

$$v_{ij} = \begin{cases} 0 & \text{if } i^{th} \text{ VM is not allocated to } j^{th} \text{ host} \\ 1 & \text{if } i^{th} \text{ VM is allocated to } j^{th} \text{ host} \end{cases}$$

H_j^{MIPS} and H_j^{mem} are the available capacity and memory respectively for the j_{th} server. r_i^{MIPS} and r_i^{mem} are capacity and memory required for the i^{th} virtual machine respectively.

B. Intelligent Water Drop Algorithm

Intelligent water drop algorithm [26], [27] is inspired by natural water flow which moves under the gravitational force of earth. The natural water ,drops follow a straight line path, if no obstacle is found, which is optimal. During the movement of water drops the following changes occur in the environment: (i) Soil is removed from the river bed and this removed soil is added to the water drop, (ii) the velocity of water drop changes based on the soil on the path it traversing. The path having less soil will increase the velocity of the water drop more. Soil removed from the path will be proportional to velocity of the water drop.

The initial soil on a water drop is denoted as $soil^{IWD}$ and the initial velocity of the water drop is denoted as Vel^{IWD}. We assume that the search space is discrete. For virtual machine placement, a fully connected graph is created with the number of nodes are equal to the number of virtual machines.

Let an IWD wants to move from a node i to node j and soil on the *arc (i, j)* is given by *soil (i, j)*. Then the velocity of IWD ($Vel^{\text{IWD}}(t+1)$) is calculated as follows:

$$= Vel^{\text{IWD}}(t) + \begin{cases} \frac{a_v}{b_v + c_v \cdot soil(i,j)}, & \text{if soil(i, j)} \neq \frac{b_v}{c_v} \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

where, $Vel^{IWD}(t+1)$ is the updated velocity and a_v, b_v, and c_v are constants.

According to the equation 3.5, Vel^{IWD} increases if the soil (i , j) is with in the range of $]-((b_v/c_v),\infty)[$ and Vel^{IWD} decreases if the soil(i, j) is with in the range of $]-(-\infty,-(b_v/c_v))[$.

We consider a local heuristic function called heuristic undesirability (HUD), that calculates the undesirability of an IWD to move from one node to another. For virtual machine placement problem, we define the HUD for a node i to node j as $HUD_{vmp}v(i,j)$:

$$HUD_{vmp}(i,j) = \frac{P_j}{r_j} \tag{3.8}$$

The HUD increases if the power consumption by a node j (P_j) increases and the resource requirement (r_j) decreases. The HUD decreases if the power consumption decreases and resource requirement increases. Hence a node j will be selected which consumes less power and require more resource.

The time taken for an IWD with a velocity Vel^{IWD} *(t + 1)* to move from the node i to the node j is denoted by *time (i,j;Vel* ^{IWD} *(t + 1))* and calculated as follows:

$$time(i,j;Vel^{IWD}) = \frac{HUD}{Vel^{IWD}} \tag{3.6}$$

such that

$$Vel^{\text{IWD}} = Vel^{\text{IWD}}(t+1) + \begin{cases} \epsilon, & \text{if } \mid \text{Vel}^{\text{IWD}}\text{(t+1)}\mid < \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

where ϵ is a small positive value.

IWD is a population-based intelligent mechanism by which a population of IWDs construct a solution and an optimal or near optimal path emerges over time. The amount of soil removed by an IWD ($\Delta soil(i,j)$) while moving from node i to node j is calculated as follows:

$$\Delta soil(i,j) = \frac{a_s}{b_s + c_s \cdot time^{\alpha}(i,j,Vel^{IWD})} \tag{3.9}$$

where α is taken randomly from 1 to 20. The other parametrs a_s, b_s and c_s are positive numbers.

If an IWD moves from the node i to the node j, the *soil (i, j)* is reduced and soil of IWD is increased. These updations are given in equation 3.10 and 3.11.

$$soil(i,j) = (1-\mu(n))\cdot soil(i,j) - \mu(n)\cdot\Delta soil(i.j) \tag{3.10}$$

where μ(n) are positive number chosen between 0 and 1.

Soil on IWD will increase as it moves from the node i to the node j by the amount of soil it removes.

$$soil^{IWD} = soil^{IWD} + \Delta soil(i,j) \tag{3.11}$$

Now, we need to select the next node with less amount of soil. This is done by assigning a probability to each next valid node (which does not violate constraints as given in section 3.1.1) from the current node.

The probability $P_i^{IWD}(j)$ of selecting a valid node j from the node i is given in equation 3.12.

$$P_i^{IWD}(j) = \frac{f(soil(i,j))}{\sum_{k \notin vc(IWD)} f(soil(i,k))} \tag{3.12}$$

such that *f (soil (i, j))* computes the inverse of the soil between node i and j as given in equation 3.13. Where *g(soil(i, j))* is soil on edge joining node i to the node j and ϵ is small positive constant and *vc(IWD)* is a visited node list of IWD.

$$f(soil(i,j)) = \frac{1}{\epsilon + g(soil(i,j)} \tag{3.13}$$

IWD algorithm involves initialization of static and dynamic parameters. The static parameters do not change during the execution of a program. In our VM placement static parameters include number of Processing elements (Pes) for each host, RAM, bandwidth and network parameter for each host and virtual machine. The flow chart of IWD algorithm is given in Figure 1. IWD Algorithm is widely used in solving several combinatorial optimization problems like Set Covering Problem, Task Scheduling in Grid Environment, and other optimization problems [28]–[30].

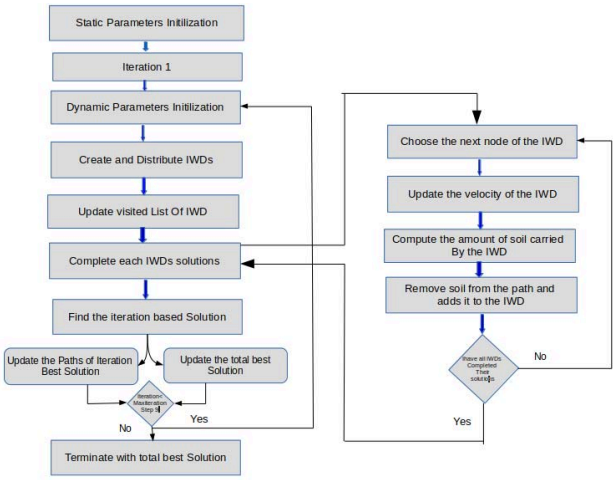

Figure 1. Flowchart of IWD algorithm

C. Modified IWD (MIWD) algorithm

We propose a modified intelligent water drop (MIWD) algorithm that only updates the soil on edge as we only consider the edge with minimum soil for selecting the next node of the path. Further, as the focus is on improving energy efficiency of the server, we consider HUD (which takes in account, the power consumption and resource requirement) along with soil on the edge in MIWD. After finding all the solutions, we calculate the power required for each of the solutions. We choose the solution which is taking minimum power as the optimal solution. The pseudo code for the proposed MIWD is given in Algorithm 1.

In our work, we create a fully connected graph which has a number of nodes equal to the number of virtual machines.

Each node represents a virtual machine which we call it as an intelligent water drop (IWD). The graph is created as a 2D array of size nxn, where n is the number of virtual machines. The virtual machines are distributed randomly on nodes. After creating the graph, we initialize each edge with some positive constant, which represent initial soil on the edge. Then, we randomly initialize the initial soil and the initial velocity of IWD with some positive constants and the algorithm terminates after 100 iterations, if the optimal solution is not found. An array is created for each IWD to track the visited node by that node. After the first inner iteration (line 7 to 16 in algorithm 1) each node will move to a next node based on soil on the edge. Then we update the visited array list of the node after each inner iteration. We choose the edge with minimum soil for selecting the next node of the path. If soil on the different edges are same, then the IWD moves heuristically, choosing the node which is not visited and requires the maximum CPU cycles. Visited array lists and soil on edges are updated accordingly.

After one complete iteration, each IWD completes a path i.e. it has visited all the nodes. Complete path of each IWD is shown in Figure 2. The path traversed by each IWD during this process will give a solution for that particular IWD. Each IWD will give one solution, so there will be many solutions which will be equal to the number of total IWDs (VMs).

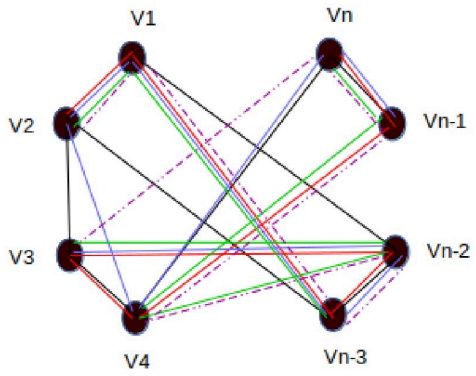

Figure 2. Final path by each iwd

IV. Performance Evaluation

A. Experiment Setup

For our experiments, we simulated a virtualized data center using Cloudsim [31]. We perform simulations with different number of Virtual Machines (50, 100, 150, 200, 250, and 300) in homogeneous and heterogeneous environments. In all the cases the number of hosts are fixed to 100.

To simulate a homogeneous data center, we use HP ProLiantMl110G4 Xeon3040 servers. For the heterogeneous environment, we use IBM X3550 XeonX5675, HP ProLiantMl110G4 Xeon3040, and HP ProLiant Ml110G5 Xeon3075 servers each having 1 processing element (Pe) with 3000 Million instructions per second (MIPS), 8.5 GB of Random Acess Memory (RAM) and 125 GB of storage. We use three types of Virtual Machines with 100, 200, and 300 MIPS each with 1 Pe respectively, having a RAM of 870, 1740, 613 MB respectively. All these VMs have 100 Mbit/s bandwidth and 2.5GB of storage.

We compared our proposed MIWD with IWD [26] and MBFD [6]. The initial parameters for MIWD and IWD are set as follows: The initial soil and initial velocity are taken as 1000. The constants a_v, b_v, c_v, a_s, b_s and c_s are taken as 3000, 0.01, 1, 1000, 0.01, and 1 respectively. The value of $\mu(n)$ is taken as 0.7 and the value of parameter α is taken as 1.

Algorithm 1 Virtual machine placement using MIWD

1: Input: List of virtual machines
2: Output: List of Host mapped with virtual machines
3: Initialize the fully connected graph of virtual machines
4: Initialize the visited node and next visited node list for each node of the graph
5: Initialize velocity of IWD and soil on IWD
6: **for** $n = 0$ to total no. of VMs **do**
7: **for** each VM **do**
8: $i \leftarrow$last visited node
9: **for** $j = 0$ to total no. of VMs **do**
10: **if** Edge soil for the edge is less and node j is not visited and
11: *VmMips(j)* $\leq$ *to Available HostMips* **then**
12: $k \leftarrow j$ ▷ j is next node to move
13: **if** soil on all edge are equal **then**
14: **if** *VmMips(j)¿VmMips(k)* and node j $\neq$ visited and
15: *VmMips(j)*$\leq$ *Available HostMips* **then**
16: $k \leftarrow j$ ▷ j is next node to move
17: *total allocated VmMips = VmMips + VmMips*$[k]$
18: next visited node for node $i \leftarrow k$
19: update next visited node in solution of ith node
20: **for** each VM **do**
21: update visited array list
22: update edgesoil for each edge as shown in equation 3.10
23: go to next host
24: **for** each solution i **do**
25: Compute power and take solution with minimum power
26: return (minmumPowerSolution)

V. Results

We evaluated the performance of our approach with 50, 100, 150, 200, 250, and 300 virtual machines in homogeneous and heterogeneous environments. The results are illustrated in Table 2 - Table 4, shows the results of the energy consumption, SLA Violation, and VM Migrations for the varying number of VMs with homogeneous and heterogeneous hosts for MBFD,

Table II
RESULTS OF MBFD ALGORITHM WITH HOMOGENEOUS & HETEROGENEOUS HOSTS

	No. of VMs	Energy (in kW)		SLA Violation		VM Migrations	
		Homogeneous	Heterogeneous	Homogeneous	Heterogeneous	Homogeneous	Heterogeneous
1	50	0.36	0.37	10	9.31	41	23
2	100	0.44	0.51	9.4	10.29	59	103
3	150	0.66	0.66	9.7	9.0	139	130
4	200	0.8	0.92	9.51	9.67	171	171
5	250	0.96	1.0	12.62	16.37	281	253
6	300	1.12	1.26	13.21	9.9	319	291

Table III
RESULTS OF IWD ALGORITHM WITH HOMOGENEOUS & HETEROGENEOUS HOSTS

	No. of VMs	Energy (in kW)		SLA Violation		VM Migrations	
		Homogeneous	Heterogeneous	Homogeneous	Heterogeneous	Homogeneous	Heterogeneous
1	50	0.35	0.36	10	10	42	42
2	100	0.49	0.55	9.78	10.72	60	121
3	150	0.71	0.67	10	10.26	145	192
4	200	0.9	0.91	10	10.03	218	214
5	250	1.07	1.12	10	9.65	308	298
6	300	1.25	1.22	8.93	8.56	305	340

Table IV
RESULTS OF MIWD ALGORITHM WITH HOMOGENEOUS & HETEROGENEOUS HOSTS

	No. of VMs	Energy (in kW)		SLA Violation		VM Migrations	
		Homogeneous	Heterogeneous	Homogeneous	Heterogeneous	Homogeneous	Heterogeneous
1	50	0.35	0.36	10	10	42	42
2	100	0.52	0.56	10	10.14	83	86
3	150	0.61	0.64	10.74	9.14	151	116
4	200	0.6	0.62	8.87	10	124	140
5	250	0.69	0.66	9.29	10	185	150
6	300	0.74	0.69	10.32	9.17	215	158

IWD, and MIWD respectively. Our proposed approaches use the minimum number of hosts while satisfying the SLA.

In the homogeneous environment,

- we observe that IWD is giving similar energy consumption with 100, 150, and 200 VMs. While, it is consuming 10% more energy with 300 VMs compared to MBFD.
- With MIWD, there is no significant difference in energy consumption for 50, 100, and 150 virtual machines but achieved 75%, 71%, and 66% energy savings compared to MBFD for 200, 250, and 300 VMs respectively.

In the heterogeneous environment,

- IWD is giving 3% of energy savings with 300 hosts compared to MBFD.
- MIWD achieved energy savings of 67%, 66%, and 54% with 200, 250, and 300 VMs respectively compared to MBFD.

The number of VM migrations for 200, 250, and 300 VMs significantly less in case of MIWD algorithm with homogeneous and heterogeneous hosts. Hence we are able to achieve better energy savings and less number of VM migrations for the MIWD algorithm with increasing number of VMs .

The comparisons of energy consumption of our proposed method (MIWD) with IWD and MBFD in homogeneous and heterogeneous environments are presented in Figure 3 and Figure 4. With increasing VMs, MIWD algorithm shows a slight increase in energy consumption, whereas IWD and MBFD shows linearly increase in energy consumption (see Figure 3 and Figure 4). The SLA violations are also less in the case of the IWD and MIWD algorithm as shown in Figure 5 and Figure 6 for homogeneous and heterogeneous hosts respectively compared to MBFD. We can observe a sudden increase in SLA Violation (16.37%) for 250 VMS in heterogeneous environment in case of MBFD algorithm as shown in Figure 6, whereas it is 9.65% and 10% in case of the IWD and MIWD respectively. Further, we observe that average VM migrations are less compared to other algorithms in all the cases for proposed MIWD algorithm as shown in Figure 7 and Figure 8.

Advantages of our approach include the flexibility and the ability to self-adapt the search for optimum solutions on the fly. Further, our algorithm search parallel from a population of Intelligent Water Drops. Therefore, our algorithm prevents premature convergence to local optimal solutions.

VI. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we investigated the problem of energy aware virtual machine placement in cloud computing. We proposed a modified Intelligent water drop algorithm for

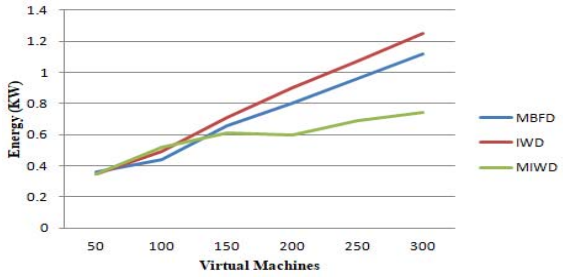

Figure 3. Comparison of Energy Consumption with homogeneous hosts

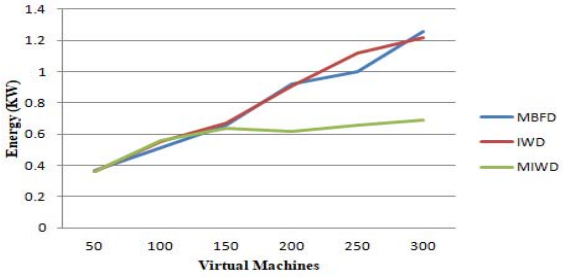

Figure 4. Comparison of Energy Consumption with heterogeneous hosts

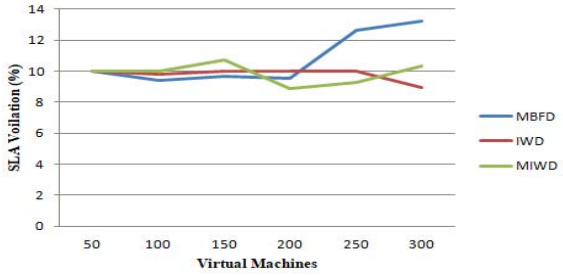

Figure 5. Comparison of SLA Violation with homogeneous hosts

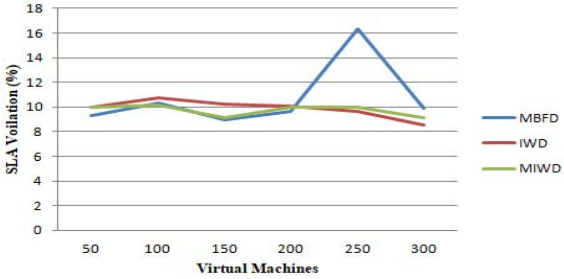

Figure 6. Comparison of SLA Violation with heterogeneous hosts

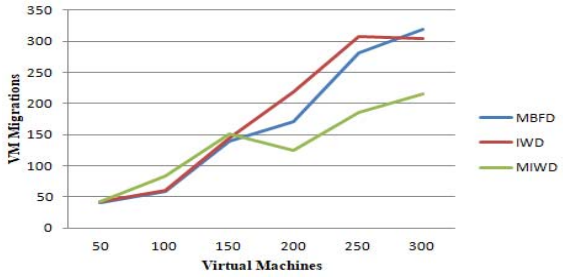

Figure 7. Comparison of VM Migrations with homogeneous hosts

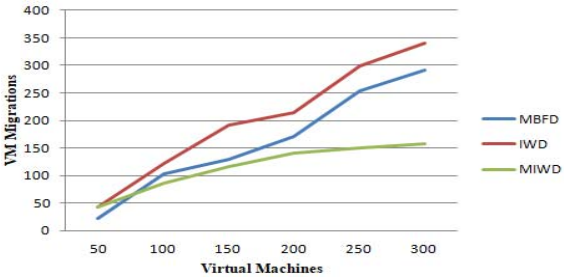

Figure 8. Comparison of VM Migrations with heterogeneous hosts

efficient placement of virtual machines considering energy saving. We found that MIWD is giving significant saving in energy when compared with MBFD and IWD. We simulated our proposed algorithm on Cloudsim for 100 homogeneous hosts and 100 heterogeneous hosts with a varying number of virtual machines. We found that there is 66% saving of energy in case of homogeneous host and 55% saving of energy in case of heterogeneous hosts with 300 Virtual machines. Particularly, we observed that our proposed algorithm is giving better energy saving and less number of VM migrations with the increase in the number of virtual machines.

Migration of VM instances between two servers need both servers to be on and causes performance overhead for applications running on those servers. In our future work, we work on VM migration considering both time and energy. Also, we will investigate other evolutionary algorithms and swarm intelligence based algorithms for energy efficient VM placement in cloud data centers .

REFERENCES

[1] Shehabi A, Smith SJ, Horner N, Azevedo I, Brown R, Koomey J, Masanet E, Sartor D, Herrlin M, Lintner W. United States data center energy usage report. Lawrence Berkeley National Laboratory, Berkeley, California. LBNL-1005775. 2016.

[2] Murugesan S, Gangadharan GR. Harnessing green IT: Principles and practices. 2012, Wiley IEEE Publishing.

[3] Reddy VD, Setz B, Rao GSVRK, Gangadharan GR, and Aiello M. Metrics for Sustainable Data Centers, IEEE Transactions on Sustainable Computing, Vol. 2, No. 3, pp. 290-303, 2017, IEEE, doi: 10.1109/TSUSC.2017.2701883.

[4] Pernici B, Aiello M, vom Brocke J, Donnellan B, Gelenbe E, Kretsis M. What IS can do for environmental sustainability: a report from CAiSE11 panel on Green and sustainable IS. Vol. 30, No. 18, 2012.

[5] Reddy VD, Setz B, Rao GSVRK, Gangadharan GR, Aiello M. Best Practices for Sustainable Data Centers. IT Professional, 2017, IEEE (In press).

[6] Beloglazov A, Buyya R. Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In Proceedings of the 8th International Workshop on Middleware for Grids, Cloud and e-Science, MGC'10, Vol. 4, No. 6, pp. 4, 2010, ACM.

[7] Wang Y, Xia Y. Energy Optimal VM Placement in the Cloud. In Proceedings of the Cloud Computing (CLOUD), 2016 IEEE 9th International Conference. pp. 84-91, 2016, IEEE.

[8] Buyya R, Beloglazov A, Abawajy J. Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. 2010, arXiv.
[9] Goudarzi H, Pedram M. Energy-efficient virtual machine replication and placement in a cloud computing system. In Proceedings of the Cloud Computing (CLOUD), IEEE 5th International Conference. pp. 750-757, 2012, IEEE.
[10] Nathuji R, Schwan K. Virtualpower: coordinated power management in virtualized enterprise systems. ACM SIGOPS Operating Systems Vol. 41, No. 6, pp. 265-278, 2007, ACM.
[11] Beloglazov A, Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurrency and Computation: Practice and Experience. Vol. 24, No. 13, pp. 1397-420, 2012, wiley.
[12] Verma A, Ahuja P, Neogi A. pMapper: power and migration cost aware application placement in virtualized systems. In Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware. pp. 243-264, 2008, Springer-Verlag New York.
[13] Meng X, Pappas V, Zhang L. Improving the scalability of data center networks with traffic-aware virtual machine placement. In Proceedings of the IEEE INFOCOM. pp. 1-9, 2010, IEEE.
[14] Dhiman G, Marchetti G, Rosing T. vGreen: a system for energy efficient computing in virtualized environments. In Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design. pp. 243-248, ACM.
[15] Li X, Qian Z, Lu S, Wu J. Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center. Mathematical and Computer Modelling. Vol. 58, No. 5, pp. 1222-35, 2013, Elsevier.
[16] Dong D, Herbert J. Energy efficient vm placement supported by data analytic service. In Proceedings of the Cluster, Cloud and Grid Computing (CCGrid), 13th IEEE/ACM International Symposium. pp. 648-655, 2013, IEEE.
[17] Tang M, Pan S. A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers. Neural Processing Letters. Vol. 41, No. 2, pp. 211-21, 2015, Springer.
[18] Wu Y, Tang M, Fraser W. A simulated annealing algorithm for energy efficient virtual machine placement. In Proceedings of the Systems, Man, and Cybernetics (SMC). pp. 1245-1250, 2012, IEEE.
[19] Dai X, Wang JM, Bensaou B. Energy-efficient virtual machine placement in data centers with heterogeneous requirements. In Proceedings of the Cloud Networking (CloudNet), IEEE 3rd International Conference. pp. 161-166, 2014, IEEE.
[20] Li B, Li J, Huai J, Wo T, Li Q, Zhong L. Enacloud: An energy-saving application live placement approach for cloud computing environments. InProceedings of the Cloud Computing, CLOUD'09. pp. 17-24, 2009, IEEE.
[21] Dong J, Jin X, Wang H, Li Y, Zhang P, Cheng S. Energy-saving virtual machine placement in cloud data centers. In Proceedings of the Cluster, Cloud and Grid Computing (CCGrid), 13th IEEE/ACM International Symposium. pp. 618-624, 2013, IEEE.
[22] Bobroff N, Kochut A, Beaty K. Dynamic placement of virtual machines for managing sla violations. In Proceedings of the Integrated Network Management. 10th IFIP/IEEE International Symposium. pp. 119-128, 2007, IEEE.
[23] Cardosa M, Korupolu MR, Singh A. Shares and utilities based power consolidation in virtualized server environments. In Proceedings of the Integrated Network Management, IM'09. IFIP/IEEE International Symposium. pp. 327-334, 2009, IEEE.
[24] Xu J, Fortes JA. Multi-objective virtual machine placement in virtualized data center environments. In Proceedings of IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing. pp. 179-188, 2010, IEEE.
[25] Reddy VD, Gangadharan GR, Rao GSVRK. Energy aware Resource Allocation and Selection in Cloud Data Centers, Soft Computing, Springer, 2017 (In Press).
[26] Shah-Hosseini H. The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. International Journal of Bio-Inspired Computation. Vol. 1, No. 1, pp. 71-9, 2009.
[27] Hosseini HS. Problem solving by intelligent water drops. In Proceedings of the Evolutionary Computation, 2007. pp. 3226-3231, 2007, IEEE.
[28] Crawford B, Soto R, Crdova J, Olgun E. A Nature Inspired Intelligent Water Drop Algorithm and Its Application for Solving The Set Covering Problem. In Artificial Intelligence Perspectives in Intelligent Systems: Proceedings of the 5th Computer Science On-line Conference, pp. 437-447, 2016, Springer, Cham.
[29] Selvarani S, Sadhasivam G. An intelligent water drop algorithm for optimizing task scheduling in grid environment. International Arab Journal of Information Technology. Vol. 13, No. 6, 2016.
[30] Alijla BO, Wong LP, Lim CP, Khader AT, Al-Betar MA. A modified intelligent water drops algorithm and its application to optimization problems. Expert Systems with Applications. Vol. 41, No.15, pp. 6555-69, 2014.
[31] Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R. 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, Wiley.