ICIC Express Letters ICIC International ©2018 ISSN 1881-803X
Volume **12**, Number **9**, September **2018** pp. 863–870

PREDICTIVE VIRTUAL MACHINE PLACEMENT IN DECENTRALIZED CLOUD ENVIRONMENT

Suresh Baliram Rathod and Krishna Reddy

Department of Computer Science Engineering
Koneru Lakshmaiah Education Foundation
Vaddeswaram, Andhra Pradesh 522502, India
sureshrathod1@gmail.com; vkrishnareddy@kluniversity.ac.in

Received February 2018; accepted May 2018

Abstract. *In distributed cloud environment hosts configured with Local Resource Monitors (LRM). This LRM monitors the underlying hosts' resource usage, runs autonomous and balances underlying host's load by migrating Virtual Machine (VM) instance to other hosts. LRM shares its own information with another peer hosts after fixed interval. It takes decision for VM migration by own without considering decisions taken by another peer hosts. This results in multiple hosts to select the same destination host during VM migration. Placing multiple VM by multiple host to the destination host without considering its future behavior may lead to over utilization of the destination host or shuts down. Static threshold usage limit dynamic environment might be unfeasible solution. To address above problems, this paper proposes the predictive host selection and VM placement policy for decentralized cloud environment. Experimental results show that the proposed predictive host selection and placement policy reduces extra VM migration occurring due to over utilization of destination host, and does balances resource usage at source and destination hosts.*

Keywords: Virtual Machine (VM), Host Controller (HC), Controller Host (CH)

1. Introduction and Related Work.

In recent years, cloud computing is gaining popularity because of virtualization. Virtualized resources are deployed, provisioned and released with minimal management effort [2]. Virtualization addresses varying resource requirement by incorporating partitioning, isolation, and encapsulation [1]. VM is core element in cloud environment. It runs on top of the hypervisor and utilizes underlying host resources. Each VM differs from other VMs by resource, CPU architecture, operating system, storage type, network utilizations and the job it has. As a result, hosts in DC have multiple VMs running parallel with different job completion time. Static threshold limit on underlying hosts resources degrades the host's performance. VM migration helps to improve hosts performance. The migration is categorized as the task migration or VM migration. In task migration, tasks from current VM are migrated to other VMs running on the same host or other hosts. VM migration involves migrating VM and its associated memory pages to the other hosts.

Cloud computing on the structure of organizations categorized as, centralized or decentralized architecture. Several cloud providers like Google, Amazon, HP, and IBM provide services by adopting either centralized or decentralized cloud architecture. Various authors have discussed approaches for VM migration decisions considering hosts current CPU utilization. The workload on hosts and data center changes frequently, causing requirement for considering hosts future CPU utilization. This leads to the problem of selecting data center architecture such that the hosts in it consider adaptive threshold for resources and do decision for VM migration considering hosts future CPU utilization.

DOI: 10.24507/icicel.12.09.863

Problem formulation. The mapping of VS to the physical host gives the solution to the VS placement. Let C be the set of physical host represented as $C = \{CH_1, CH_2, CH_3, \ldots, CH_m\}$ and V be the set of virtual servers deployed on each physical server denoted as $V = \{VS_1, VS_2, \ldots, VS_n\}$. $V_{i,j}$ is the virtual server i deployed on the physical server j, such that $(1 < i < n)$ and $(1 < j < m)$. $X_{i,j}$ is the binary decision variable representing whether the VS_i selected from the host C_j. This requires VS placement to the host from the set of the host C_j. Let $t_1, t_2, \ldots, t_n$ be the time interval for VS CPU utilization. The host CPU utilization at time interval t is given by Equation (1).

$$CH_{(i,t)}(u) = CH_{(i,t)_{idle}} + \left(CH_{(i,t)_{\max}} - CH_{(i,t)_{idle}}\right) \tag{1}$$

Here, $CH_{(i,t)_{\max}}$ is the power consumption at maximum CPU utilization and $CH_{(i,t)_{idle}}$ is the power consumption when the server is active but is in an idle state. The mapping of V_i to the C_j such that the energy consumption of C_j at t is minimum. Let $P_{i_{util}}$ be the CH's past CPU utilization.

$$\forall \sum_{j=1}^{m} X_{i,j} = 1 \tag{2}$$

$$\forall_j \sum_{i=1}^{m} VS_{cpu} + P_{i_{util}}, \quad iX_{i,j} \leq C_{cpu,j} \tag{3}$$

$$\forall_j \sum_{i=1}^{m} VS_{mem}, \quad iX_{i,j} \leq C_{mem,j} \tag{4}$$

where i is the virtual server and j is the physical host. The above Equations (2) and (3) discuss the virtual server should not exceed the physical resources.

Most of the authors considered current CPU utilization, future CPU utilization of the VM, or two threshold resource limits as the parameter for VM migration. The authors in [4] discussed the mechanism where host shares its information after fixed interval and how VS selection and VS placement policy are using neighborhood host information. The authors in [5] proposed the solution for VM placement using Ant Colony Optimization (ACO) technique. The authors have tested their approach using P2P unstructured network topology and have considered CPU utilization as a parameter. Energy based VM placement approach proposed by [6] discussed VM migration by considering penalty cost and energy consumption as the parameters for VM selection. CPU utilization based distributed load balancing proposed by the authors in [7] considered hypercube based VM placement and migration. Authors in [8] have proposed optimum dynamic VM placement policy considering hosts CPU consumption; they have discussed the Maximum Processing Power (MPP) and Random host's Selection (RS) as an approach for VM migration. VM migrated to destination host by preserving VM's firewall rule. In [9] the authors have proposed hierarchical decentralized dynamic VM consolidation framework for VM migration, wherein they discussed how the global controller takes decision for VM migration by considering hosts future CPU utilization. Distributed load balancing using CPU utilization proposed by the authors in [10] considered hypercube based VM migration. Randomized probabilistic technique for distributed live VM migration proposed by [11] discussed hosts pair formation and initiating VM migration in the selected host pair. Correlation based VM placement on centralized cloud architecture was proposed by [12]. Cluster based VM consolidation was proposed by the authors in [13], where they have discussed batch oriented VM consolidation and on demand VM placement. Muti target based VM placement using genetics algorithm proposed by the authors in [14] considered SLA violation and CPU utilization as the parameter for VM migration decision making on centralized cloud architecture. In [15], authors have proposed reinforcement learning based VM placement wherein the authors discussed how centralized host learns VM deployment and puts host in sleep mode or in active mode considering the past traces.

In this work authors have contributed VM placement for decentralized cloud environment. Here, authors have considered destination host's future utilization to resolve the problem of the same host identification by multiple hosts during VM placement phase. The remaining portion of this paper is organized as follows: Section 2 discusses the proposed system followed by Section 3 to discuss the results of the proposed system and at last the conclusion of paper is given in Section 4.

2. Predictive Decentralized Virtual Machine Placement.

The proposed hybrid decentralized cloud architecture formed considering the distributed features like multi-tenant architecture, distributed storage, parallel processing and multithreading. Here, hosts are categorized as HC and CH. Host is termed as HC, if it does the decisions for VM placement and has running VM's instances and if host has running VM instances termed as CH. Hosts in the proposed architecture are configured with the agents as shown in Figure 1 and its details are below.

HC Resource Monitor (HCRM): This does decisions for VM migration and performs the tasks like collecting CH detail, storing CH detail to current and past utilization table and providing CHs information to the virtual host manager. This component activated when CH acts as HC.

Local Resource Monitor (LRM): This interacts with the underlying hypervisor and does share its details with HCRM. This component is available with every CH.

Virtual Host Manager (VHM): Unlike HCRM, this component activated whenever the CH acts as HC. This performs source and destination CH identification during VM migration, does predict destination CH's future CPU utilization, finds upper threshold usage for every CH, and finds next HC.

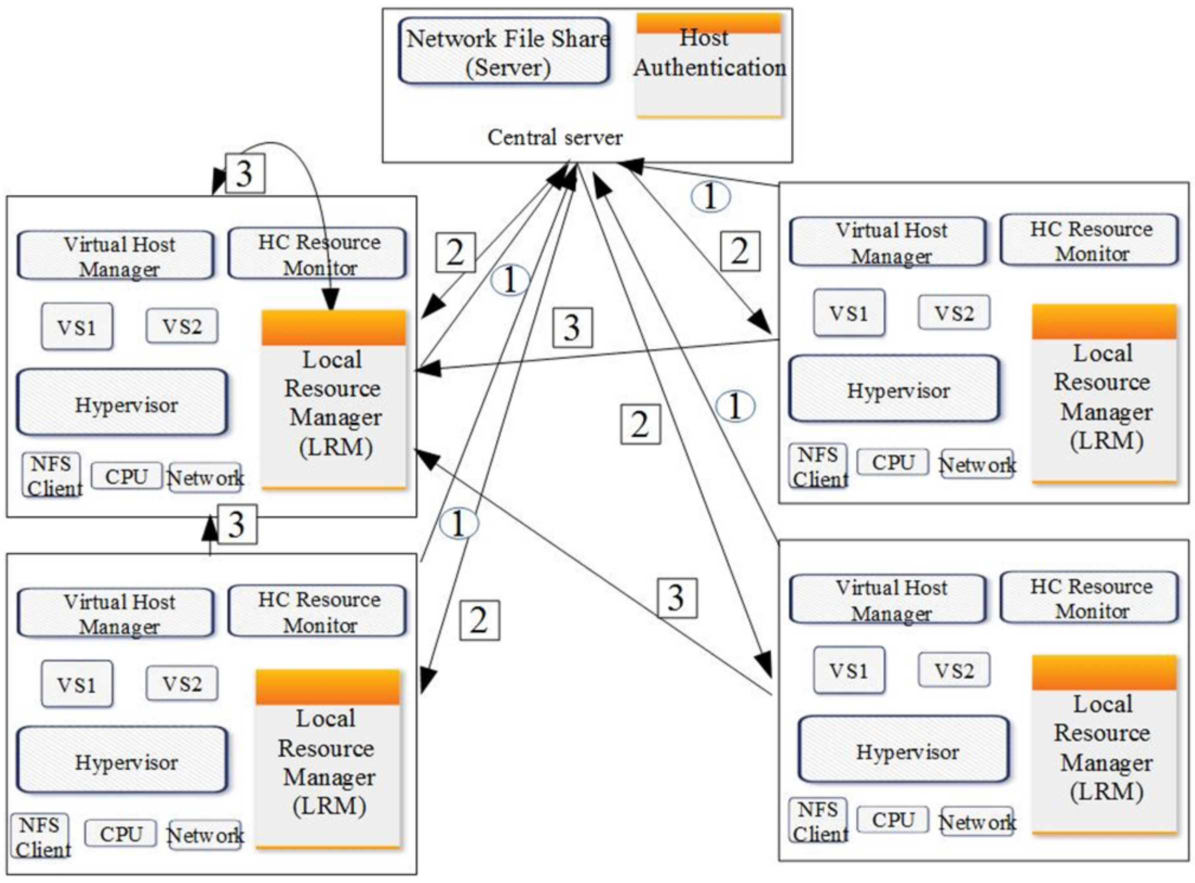

Figure 1. Decentralized hybrid host component diagram

When all the hosts (CH) powered on, the CH's starts connecting to HC. HC after fixed interval initiates daemon threads that will collect CH detail, manage remote VM, and identify next HC. CH shares its detail with HC in the form shown in Table 1.

Table 1. Host's information exchange form

Address	No. of VM	CPU utilization	Status	Time

Here, the status flag identifies whether the host currently acts as HC or CH. Address refers to CH address.

Algorithm 1 DPPVP

```
1:  procedure
2:    Host ← GETCONNECTIONDETAIL ()
3:    for i ← 1 to length [HOSTLIST] do
4:    MED ← FINDMEDIANHOST [i]
5:    CURRUTIL [i] ← HOSTLIST [i]
6:    FHOST [i] ← FINDFUTURE (id)
7:     GETMIN ← (CURRUTIL)
8:    dest ← GETMAX (CURRUTIL)
9:    id ← GETVICTIMID (src)
10:   putil ← CURRUTIL [id]
11:   if (putil ≥ MED [dest])
12:   dest ← FINDNEXT (dest, HOST)
13:   else for i ← 1 to length [HOST] do
14:    if HOST [i] == dest
15:   if MED [i] ≥ 0.9
16:    MED [id] ← 0.9
17:    else if MED [i] ≥ FHOST [i]
18:   INITMIGRATION (src, dest)
19:    return
20:   else if MED [i] ≤ FHOST [i]
21:    address ← FINDNEXT (dest, src, CURRUTIL)
22:   INITMIGRATION (src, dest)
23:    return
```

VHM at HC initiates the procedure for VM migration using DPPVP algorithm and next HC identification. DPPVP algorithm is shown in Algorithm 1. CH shares its CPU utilization to HC at fixed interval. The CH's current CPU utilization is computed using Equation (5).

$$\mathrm{H_U} = \sum_{i=0}^{n} \mathrm{VM}_i \tag{5}$$

Here, $\mathrm{H_U}$ is the host utilization of host u, and it is the sum of all VM_i running on the host u at time interval t. Upon receiving CH detail, VHM at HC creates daemon thread that will store received CH's information in current and past utilization table. The HC stores all active CH's address by making a call to GETCONNECTION. HC refers HOST to find source and destination host address during migration. Each host's dynamic threshold limit computed using MAD [9] is computed using Equation (7).

$$\mathrm{MAD} = \frac{\sum_t^n y_{t-\hat{y}}}{n} \tag{6}$$

$$\mathrm{UpperThreshold} = 1 - \mathrm{MAD} \tag{7}$$

Here, y_t represents actual CH's utilization and n represents a number of observations and $\hat{y}$ represents fitted value at time t. The VHM, after finding upper threshold limit for the hosts, it initiates the thread to find the CH that has maximum and minimum CPU utilization. DPPVP applies GETMIN and GETMAX to finding the CH that has minimum and maximum CPU utilization.

The CPU utilization of all the CHs is obtained referring current CPU utilization table. CH with maximum CPU utilization is selected as source and the VM that has maximum resource consumption is marked as VM to migrate to the CH (destination) having minimum CPU utilization at current instance of time. The VM is placed to the identified destination CH if the identified destination CH has its future CPU utilization less than its upper threshold and the upper threshold is less than or equal to 0.9.

The VM placement discarded to the identified destination host if the CH's current utilization of destination host is greater than 0.9 and the CH's future utilization greater than upper threshold. If the computed upper threshold value of CH is greater than 0.9, the new upper threshold would set to 0.9. The new CH is identified if the VM placement failed to place on the destination host. On CH identification over the VM from source CH migrated to the new identified CH. The CH's future CPU utilization computed by Doubles Exponential Smoothing (DES) [16], the smoothed value of CH computed using Equation (10).

$$S_t = \alpha y_t + (1-\alpha)(s_{t-1} + b_{t-1}), \quad 0 \le \alpha \le 1 \tag{8}$$

$$b_t = \gamma(s_t - s_{t-1}) + (1-\gamma)b_{t-1}, \quad 0 \le \gamma \le 1 \tag{9}$$

$$f_{t+m} = s_t + mb_t \tag{10}$$

Here, S_t represents smooth values at time t, the y_t represents observed values over a time period t. b_t represents trend factor over time period t values for the previous period b_{t-1}. The pseudo code for FINDNEXT algorithms is as shown in Algorithm 2. The CH's is said to be in normal state, if it has its CPU utilization less than 0.7. If the CH has utilization greater than 0.7 it is considered as over utilized.

Algorithm 2 FINDNEXT

```
1: procedure
2:    for i ← 1 to length [HOSTLIST]
3:    for j ← 1 to length [HOSTLIST]
4:    if (HOSTLIST [j] ≤ HOSTLIST [j − 1])
5:    exchange (HOSTLIST [j], HOSTLIST [j − 1])
6:    for i ← 1 to length [HOSTLIST]
7:    if (haddr ≠ HOSTLIST [i])
8:     return HOSTLIST [i]
```

3. Results and Discussion.

The proposed framework is developed considering hybrid peer to peer network topology. Each CH is configured with KVM/QEMU hypervisor, OpenJDK 1.6, Libvirt, JNA, python panda and Network File Share (NFS) client. Central host is configured with NFS server and CH with NFS client. Every CH is configured with central host address such that on startup CH connects with central host and retrieves HC address from central host after fixed interval. CH on receiving HC address, they start sharing their own detail with HC. HC refers the stored CHs detail from the past and current utilization table during host identification phase, for next HC identification and to find CH's future CPU utilization. Current utilization table at HC is shown in Table 2.

In non-predictive VM placement, the VHM at HC applies the VM placement after fixed interval referring current CPU utilization table. From Table 2, it is found that 10.0.0.3 has maximum CPU utilization and 10.0.0.1 has minimum CPU utilization. DPPVP at

HC starts searching VM from host 10.0.0.3 having minimum CPU utilization. If VM having CPU utilization is identified, HC initiates the VM migration from CH with address 10.0.0.3 to the CH with address 10.0.0.1. This approach leads to some host's to over utilized or underutilized and requires shutdown due to migration. Result for non-predictive VM placement is shown in Figure 2(a) and the time required to migrate VM instance is in Figure 2(b). Referring Figure 2(a) it is found initial utilization of hosts 10.0.01, 10.0.02, and 10.0.0.3 minimum utilization but after several minute the load at 10.0.01, 10.0.02, and 10.0.0.3 increased over time. To reduce over utilization due to migration, the predictive VM placement is incorporated. In the predictive VM placement HC refers both current and past utilization table. HC refers current utilization table to find CH address for migration and to find next HC. Past CPU utilization table is used to find destination CH's future utilization. From Table 2, it is found that the host with address 10.0.0.3 has the maximum CPU utilization at current instant of time and the host with address 10.0.0.1 has minimum CPU utilization. The HC marks 10.0.0.1 as the destination host and 10.0.0.3 as the source host. Before VM placement to CH 10.0.0.1, the DPPVP performs the check and finds whether the future utilization of the host is less than the upper threshold limit. VM migrated from 10.0.0.3 to 10.0.0.1 if the future utilization is less than upper threshold limit. DPPVP starts thread to find new CH and initiates VM migration from 10.0.0.3 to new CH address, if the future utilization of new CH is greater

Table 2. Current utilization table at HC

Host address	CPU utilization	NO.VM	Status
10.0.0.3	0.212	4	FALSE
10.0.0.2	0.098	2	FALSE
10.0.0.1	0.13	1	TRUE
10.0.0.3	0.168	3	FALSE
10.0.0.2	0.128	3	FALSE
10.0.0.1	0.133	1	TRUE
10.0.0.3	0.125	2	FALSE
10.0.0.2	0.165	4	FALSE
10.0.0.1	0.135	1	TRUE
10.0.0.3	0.153	3	FALSE

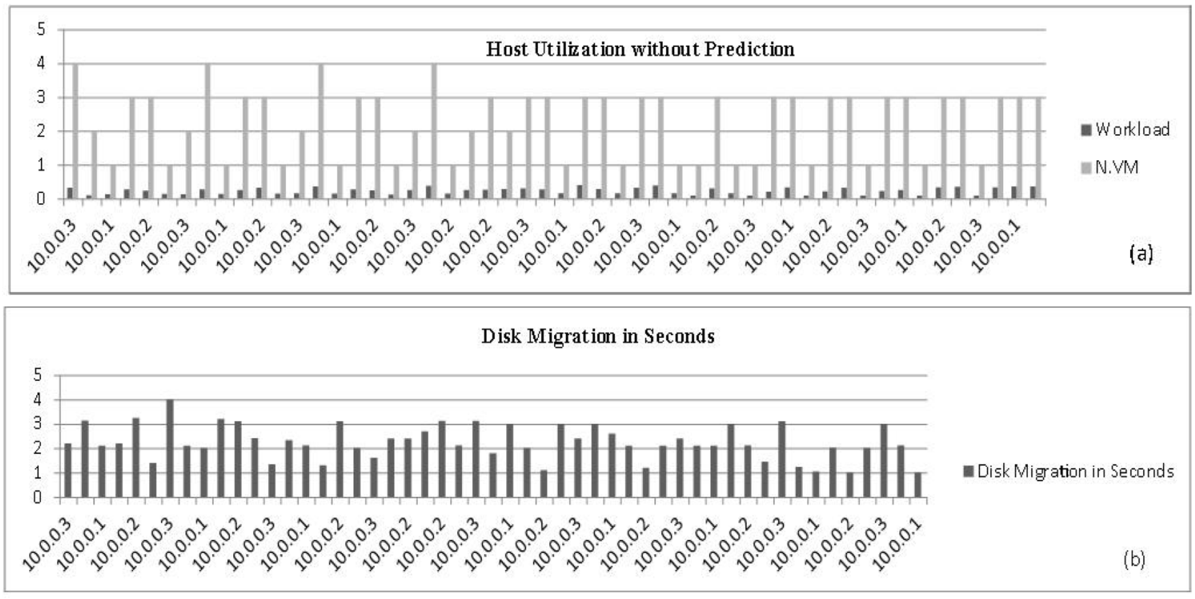

Figure 2. Non predictive VM placement

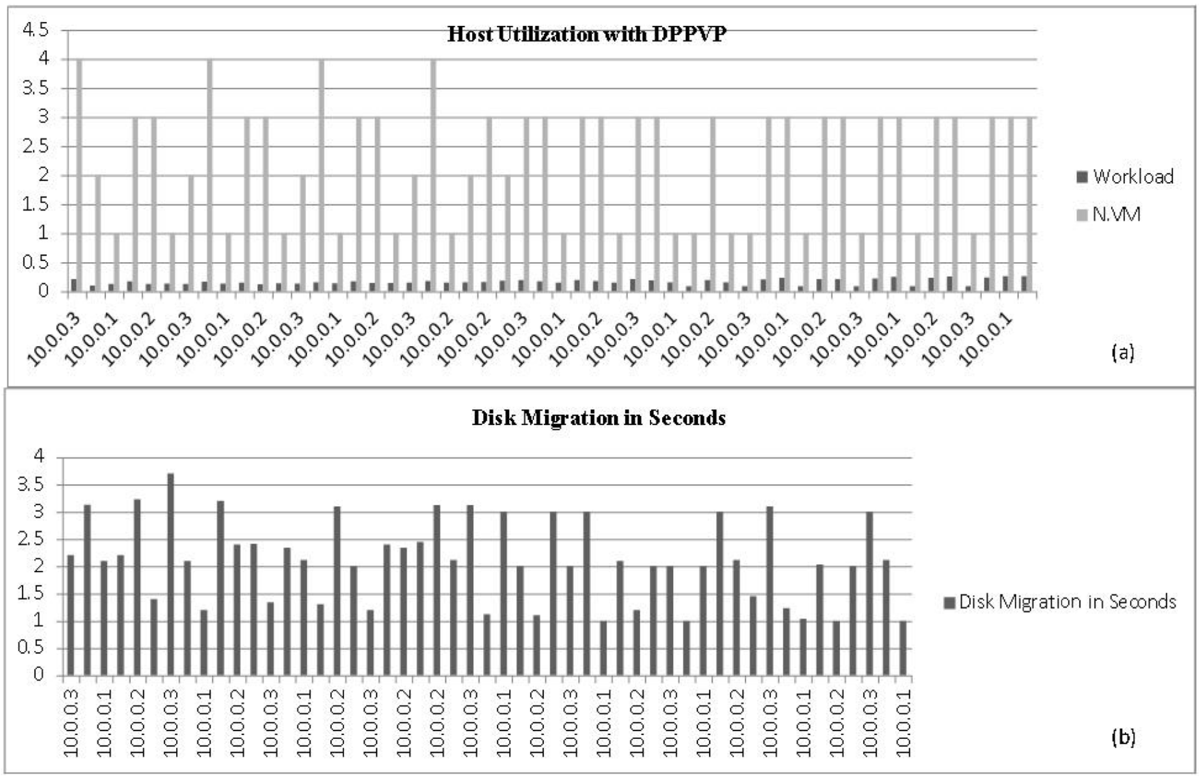

FIGURE 3. Predictive VM placement using DPPVP

than upper threshold limit. Upper threshold for CH is computed using Equation (7) and future utilization using Equation (10).

The predictive VM placement considers destination host's future utilization during VM placement phase and at the same time it ensures that the destination host remains in normal after VM placement. Here, the proposed predictive VM placement avoids over utilization of the destination host.

4. Conclusions.

In this proposed predictive VM migration for distributed cloud the categorization of hosts into CH and HC helps to reduce network bandwidth consumption and CPU power by restring the network traffic between CH and HC. Facilitating decisions for VM migration at HC helps avoid the same host selection by multiple hosts for VM placement. Predicting future CPU utilization towards destination host avoids unnecessary VM placement and over utilization due to VM placement. This work further can be extended to provide security during migration.

REFERENCES

[1] *National Institute of Standards and Technology*, https://www.nist.gov/.

[2] *Cloud Computing Defined Characteristics Service Levels*, Cloud Computing News, https://www.ibm.com/blogs/cloud-computing/2014/01/cloud-computing-defined-characteristics-service-levels/.

[3] F. Paraiso and P. Merle, A study of virtual machine placement optimization in data centers, *The 7th International Conference on Cloud Computing and Services Science*, Porto, Portugal, pp.343-350, 2017.

[4] E. Feller, C. Morin and A. Esnault, A case for fully decentralized dynamic VM consolidation in clouds, *Proc. of the 4th IEEE International Conference on Cloud Computing Technology and Science*, Taiwan, pp.26-33, 2012.

[5] W.-T. Wen, C.-D. Wang, D.-S. Wu and Y.-Y. Xie, An ACO-based scheduling strategy on load balancing in cloud computing environment, *The 9th International Conference on Frontier of Computer Science and Technology*, China, pp.364-369, 2015.

[6] D. Grygorenko, S. Farokhi and I. Brandic, Cost-aware VM placement across distributed DCs using Bayesian networks, *Economics of Grids, Clouds, Systems, and Services*, pp.32-48, 2016.

[7] R. Benali, H. Teyeb, A. Balma, S. Tata and N. B. Hadj-Alouane, Evaluation of traffic-aware VM placement policies in distributed cloud using CloudSim, *Proc. of the 25th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises*, France, pp.95-100, 2016.

[8] Z. Bagheri and K. Zamanifar, Enhancing energy efficiency in resource allocation for real-time cloud services, *The 7th International Symposium on Telecommunications*, Iran, pp.701-706, 2014.

[9] M. H. Ferdaus, M. Murshed, R. N. Calheiros and R. Buyya, An algorithm for network and data aware placement of multitier applications in cloud data centers, *Journal of Network and Computer Applications*, vol.98, pp.65-83, 2017.

[10] M. Pantazoglou, G. Tzortzakis and A. Delis, Decentralized and energy-efficient workload management in enterprise clouds, *IEEE Trans. Cloud Computing*, vol.4, no.2, pp.196-209, 2015.

[11] S. Nikzad, An approach for energy efficient dynamic virtual machine consolidation in cloud environment, *International Journal of Advanced Computer Science and Applications*, vol.7, no.9, pp.1-9, 2016.

[12] Y. Zhao and W. Huang, Adaptive distributed load balancing algorithm based on live migration of virtual machines in cloud, *The 5th International Joint Conference on INC, IMS and IDC*, Nexus, pp.170-175, 2009.

[13] X. Fu and C. Zhou, Virtual machine selection and placement for dynamic consolidation in cloud computing environment, *Front Computer Science*, vol.9, no.2, pp.322-330, 2015.

[14] F. Teng, L. Yu, T. Li, D. Deng and F. Magouls, Energy efficiency of VM consolidation in IaaS clouds, *Journal of Supercomputing*, vol.73, no.2, pp.782-809, 2017.

[15] E. Arianyan, H. Taheri and S. Sharifian, Multi target dynamic VM consolidation in cloud data centers using genetic algorithm, *Journal of Information Science and Engineering*, vol.32, no.4, pp.1575-1593, 2016.

[16] *Double Exponential Smoothing Insight Central*, https://analysights.wordpress.com/tag/double-exponential-smoothing/, 2017.