*International Journal of Engineering & Technology*, 7 (2.7) (2018) 705-709

**International Journal of Engineering & Technology**

*Website: www.sciencepubco.com/index.php/IJET*

***Research Paper***

# Decision Making Framework for Decentralized Virtual Machine Placement in Cloud Computing

**Suresh B.Rathod[1]*, V.Krishna Reddy[2]**

*[1]Research Scholar, Department of computer Science and Engineering,*
*Koneru Lakshmaiah Education Foundation, Vaddeswaram, India.*
*[2]Professor, Department of computer Science and Engineering*
*Koneru Lakshmaiah Education Foundation, Vaddeswaram, India.*
**Corresponding author E-mail: sureshrathod1@gmail.com*

## Abstract

In distributed cloud environment hosts are configured with Local Resource Monitors (LRM). This LRM monitors the underlying hosts' resource usage, runs independently and balances the underling host's load by migrating Virtual Machine (VM) instance. For the dynamic environment, each hosts has varying resource requirement, hosts load cannot remain constant. LRM at each host takes decision for VM migration considering static threshold on its own and other hosts current CPU utilization. This result in chances of getting selected same host for VM placement by multiple hosts to reduce resource usage of underlying hosts. The decision making at each server causes the problem of same host identification by multiple hosts during VM placement and consumes extra CPU power and network bandwidth consumption towards each server. This paper addresses the above said issue by proposing decentralized decision making framework for cloud considering hybrid Peer to Peer (P2P) network topology. Proposed solution results avoiding above said issues and balances the load across servers in DC.

***Keywords***: *Cloud computing (CC), Virtual Machine (VM); Managing Host (MH); Acting Host (AH).*

## 1. Introduction

Virtualization considered as the default technology in cloud computing. In virtualization VM provides uninterruptable services to end user. Each VM differs with other VMs in resource it has that includes processor architecture, operating system (OS) it has, memory type, network bandwidth assigned and the tasks it has assigned during its executing. Resources in the cloud environment gets deployed, provisioned and released with minimal management effort [1]. This result the physical servers in Data Centre (DC) have various VM's running parallel having different job completion time. Cloud provider does frequent VM migration in order to maintain service up for long duration. In migration, VM neither assigned new IP address nor changes its location. The VM state and its associated memory pages migrated from one server to the other server. VM memory migration has several phases as listed below.

**Push phase:** Here pages of the selected VM from disk get pushed in the network. VM remains in running state towards source server. Consistency towards VM's memory page maintained by moving modified page from source server to the destination server.

**Stop and copy phase:** Here execution of the VM stopped at source server then the VM's stored pages get transferred towards destination server in rounds. In the last round, the VM transferred and does resume its execution at the destination server.

**Pull phase:** Here VM runs at destination server; if VM requires any page, the page fault get created across the network to retrieve the page from the destination server.

The VM migration dome in either considering Pre-copy or Post-copy.

**(1) Post-copy migration:** In this type of migration initially VM state transferred to the destination server followed by its memory contents. Memory contents get transferred in rounds [2].

**(2) Pre-copy migration:** In this initially the selected VM's memory pages transferred in rounds to the identified destination server. In last round, VM's state get transferred[2].

Hypervisors like XEN, KVM, Hyper-v and VMware's ESXi [17] do provide a facility for VM migration. The VM migration can be static or live migration.

In static VM migration, the VM's execution is suspended towards source and will get resumed towards destination.

In live migration execution of the selected VM is paused towards source and resumed at the destination. Memory pages migrated in rounds towards the destination in rounds. To achieve live VM migration, the VM need to be selected from one of the servers and need to be placed to the server such that placing VM on it should not violate the underlying server's resources limit.

Authors considered either centralized or decentralized cloud framework to achieve virtual machine's identification and placement. In centralized cloud environment the resources controlled by the single server. Cloud vendors provide its services adopting either centralize or decentralized cloud framework. Amazon had problem of service down due to S3 instance failure [3]. The regular downtimes due to instance failure avoided considering a modification to the cloud architecture.

The decentralized cloud environment avoids centralized failure considering resource management at each peer server. Most of the decentralized cloud environment formed by establishing Peer to

Peer (P2P) network topology. Servers in P2P network, each peer server shares their detail with other peer server after fixed interval. This results in extra bandwidth consumption and processing power towards each server. This also has the problem for overutilization at destination server as the peer server does not shares its decision for destination host identification for VM placement. This cause multiple server to identify same destination server for VM placement. This lead to over utilization of destination server. It results the destination server initiates the process of VM migration to reduce its workload. To overcome the above issues in decentralized architecture, the decentralized hybrid approach needs to considered such that it would avoid the above mentioned limitation.

The paper is organized as follows: section 2 describes related work, section 3 proposed decentralized VM migration framework and section 4 discusses the Result analysis of the proposed system followed by a conclusion.

## 2. Related Work

Several authors considered network bandwidth, disk storage, No of I/O as the parameters for decision making. The author in [4], have proposed the distributed architecture for peer to peer network, Here the author explained how the peer servers in DC forwards its host utilization to the server and all other servers available in the DC. Such data transfer in DC floods network traffic to each server. The author in [4], have discussed the mechanism for a decentralized framework for controlling and monitoring virtual resources across servers in DC. The authors have discussed how the servers do exchange their information at a fixed interval of time. Authors [4] have considered CPU utilization as the parameter for VM selection and placement. The authors [4] have considered static threshold limit while making the decision for migration.

Authors in[5] proposed Ant Colony Optimization (ACO) based scheduling strategy; where they discussed ant colony based VS migration. The author proposed ant colony based path categorization, the author categorized path as positive or negative. The authors considered the positive path for VS migration. One of the servers was considered as the originating server whenever the server found finds its utilization is increased and crosses the threshold value[5]. The authors did not consider whether there is any other server already initiated the process of migration and selected the same server as destination server in its positive path. The author has considered present utilization as the parameter for VS placement. The author has not considered decision from another server.

The authors in [6] discussed how VM selection and VM placement policy using neighbourhood server information. The authors [6] have considered P2P unstructured network. They have considered CPU utilization as the base parameter for decision making.

Energy based VS placement approach was proposed by the authors in [7], where they have discussed VS migration considering penalty cost and energy consumption with each server as the parameters for the VM selection. The solution proposed by the author [7] has limitation like, if the energy cost and penalty cost increased, the overall performance will be degraded.

Distributed load balancing using CPU utilization is proposed by the authors in [8], where they proposed VS placement using hypercube. Here, the servers take decisions for VS migration without considering other server decision.

The author in [4], proposed optimum dynamic VS Placement policy using CPU consumption, but they have not considered the network topology for VM placement.

The authors in [9], discussed VM migration using the maximum processing power (MPP) and random selection (RS) approach AH, the solution proposed by [9] preserved same firewall rule for VM to the destination server after migration.

The authors in [10], have proposed Hierarchical Decentralized Dynamic VS Consolidation Framework, where they discussed how the global controller does take decision for VM migration. The author[10] has considered the future utilization of the server. The servers in the solution proposed by the author[10] do not consider decisions from another server before taking decision for VM migration.

The authors in [11] proposed adaptive VM migration, where they considered adaptive two-level threshold. They considered this threshold to identify the servers as a troublemaking server (over utilized). This server is categorized as trouble making at present instance or in future[11]. The author has considered the server as input and predicted its future regarding whether the server needs VM migration. The author[11] have considered fix set of VM.

Authors in [12], have proposed the distributed live VM migration wherein they have discussed VM migration using randomized probabilistic pair formation between the servers. The migration initiated between the selected server pair [12]. This random pair selection results in skipping the over utilized server.

Authors in [13] proposed correlation based VM placement. Here the authors have simulated there proposed VM placement using coudsim[13].

## 3. Decision Making Framework for Decentralized Virtual Machine Migration

Central server in the centralized cloud framework does task for VM identification and placement. This centralized framework suffers with centralize failure. If the central server goes down, all the hosted services on different servers will also goes down. End user could not access its various services hosted on servers in DC. To overcome this, the decentralized VM selection and placement policy need to be considered.

### 3.1. Problem formulation

The mapping of VM to the physical host gives the solution to the VM placement. Let A is set of physical servers represented as $A=\{AH_1, AH2, \ldots\ldots AH_m\}$and V is set of virtual servers instances running at ach physical server denoted as $VS=\{VMI_1, VMI_2, VMI_3 \ldots, VMI_n\}$. $VS_{j,k}$ is the virtual server j running at physical server k, such that $(1<j<n)$ and$(1<k<m)$. $B_{j,k}$ is the binary decision making variable. It suggests whether the $VMI_j$ requires selection from the identified physical server $C_j$. This requires VM placement to the host from the set of host$C_j$ to be placed on one of the host from C hosts. The mapping of $VMI_j$ to the $C_j$ such that the energy consumption of $C_j$ at t is minimum.

$$\forall \sum_{j=1}^{m} B_{j,k}=1 \quad (1)$$

$$\forall_j \sum_{j=1}^{m} VM_{cpu}, iB_{i,j} \leq C_{cpu,k} \quad (2)$$

$$\forall_j \sum_{j=1}^{m} VM_{mem}, iB_{j,k} \leq C_{mem,k} \quad (3)$$

Here j is the virtual machine running instance and k is the physical server. As in (2) and (3) it found that the virtual server resource allocation should not exceed physical server's resources on VMI placement to the destination server$C_j$.

### 3.2. Proposed Architecture

This section discusses the proposed decentralized MC based virtual machine selection and placements policy. The proposed framework formed by interconnecting physical servers forming hybrid P2P network. Here, the physical servers categorized as managing host (MH) and Acting Host (AH). The physical hosts configured with following components.

**HC Resource Monitor (HCRM):** This is the component available at the CH. It gets activated only when the CH acts as HC and does make decisions for VM migration. It performs tasks like

collecting and storing peer hosts detail, providing host information to the Virtual Host Manager (VHM) as and when required.

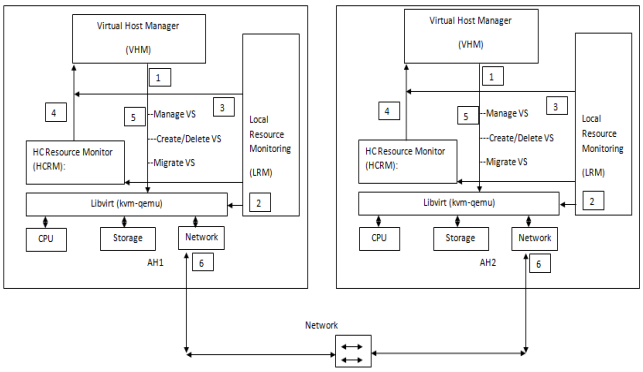


**Fig. 1:** Proposed physical servers configuration.

**Local Resource Monitor (LRM):** This component available at each CH. It interacts with the underlying hypervisor, collects underlying host detail and shares this collected information with the HCRM.

**Virtual Host Manager (VHM):** Unlike HCRM, it gets activated whenever the CH acts as HC. It identifies source and destination for VM migration

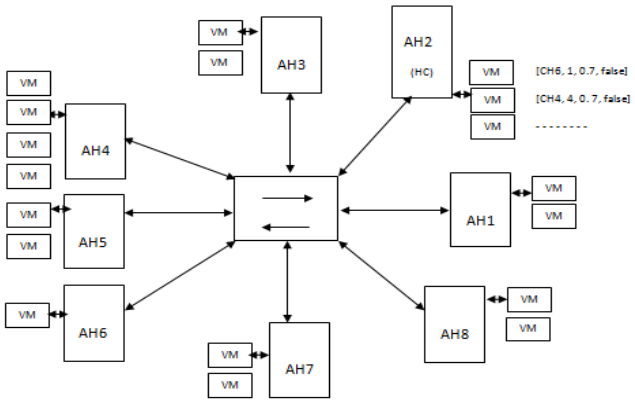


**Fig.2:** Decentralized peer to peer based network topology

Fig.1 shows the physical servers configuration with the component. Fig. 2 shows the generalized view of P2P network formation. In Fig.2, there are six servers AH1, AH2, AH3, AH4, AH5, and AH6 each configured with KVM/QEMU hypervisor.

### 3.3. Managing Host Selection

Initially all the AH configured as shown in table I. all AH on start, starts connecting with the central host. The central host selects one of the AH random from the set of AHs and marks such AH as the MH. LRM running at AH starts retrieving server detail from the underlying hypervisor. The LRM shares this collected details with HCRM after fixed interval. The HCRM at MH starts receiving server's details and updates this AH to the AH table. In Fig.2 MH represented as server AH6.

After fixed interval MH initiates the procedure for finding new MH on receiving all AHs server details. New MH identification required because in the P2P every server provides services to end user. As current MH has VM's running on it, it might get over utilized because of varying resource requirement from the VM.

The identification of new MH done by comparing current MH utilization with other AH's utilization. This done identification done is such that the new MH has the minimum CPU utilization compared with all other AH. AH with minimum CPU utilization marked as the MH. Once new MH identified, the current MH performs status check for newly identified MH. This check required because the selected AH will be taking care of future VM selection and placement. If it is shut down state, then new MH identified. Upon receiving reply from the newly selected AH, the current MH broadcasts newly identified MH address to all AH that are currently connected with MH. Following algorithm shows the new Managing Host selection.

ALGORITHM MH_SELECTION (HOSTS [])
//input number of server hosts []
//sort the servers in as per the workload
1) sort [HOSTS.workload)
//find the server with minimum workload
2) addr=null
3) Min.util = HOSTS [0]. util
4) boolean status=false;
5) for (i = low; i <= high; i++) {
6)if((HOSTS [i].util < min.util) &&( HOSTS [i].vm<min.vm))
7) Min.util = HOSTS [i].util
8) addr= HOSTS [i].addr;
9)}
10) if ((MH.util >min.util) )
11) status=true
12) else
13) status =false
14) if(status==true)
15) check=alive()
16) else if(status==false)
17) broadcast (MH.addr)
18) if(check==true)
19) MH.addr=addr
20) if (check==false)
21) MH_SELECTION (hosts)
22) BROADCAST (MH.addr);

From Fig. 2 the server AH4 is the MH starts receiving messages from all remaining servers AH1, AH3, AH4, AH5, AH2, and AH7. AH4 stores all AH details in as in Table 2. MH initiates the process for finding the next MH selection algorithm by traversing AH's details and marks one of the AH as new MH considering its current CPU utilization. Here, the AH3 has minimum CPU utilization. AH6 marks AH3 as new MH and broadcasts new MH address to all remaining servers in DC. Upon receiving a message from the MH, the servers AH1, AH2, AH3, AH4, AH5, AH6 updates MH address and starts forwarding their server detail to new MH.

### 3.4. VM selection and placement

The AH having running VM's instance has varying CPU utilization. Utilization of the server computed as the sum of all VM's CPU utilization. If any VM has change in requirement because of variation in application demands the underlying server's utilization gets changed. Each server has maximum threshold limit, beyond which it cannot fulfil the resource requirement from the VM. The performance of the underlying server degrades if the VM consumes large resources in such case server need to be shut off. This leads requirement for incorporating new method such that the VM with minimum utilization gets migrated to another servers that has low CPU utilization resolves above issue.

In this proposed work, VHM at MH does the job for VM identification from the set of AHs. This can be done by traversing the AH's information stored at MH and finding AH that has maximum CPU utilization and has running VM instances. The VHM marks such AH as the server from which VM needs to be migrated. Following VM_SELECTION finds the AH that has VM to be migrated.

ALGORITHM VM_SELECTION (HOSTS [])
//input number of servers HOSTS []
//sort the servers in as per the workload
1) SORT [HOSTS.util)

```
//find the server with maximum workload
3)  max_util = HOSTS [0].util
4)  adddr=null,index=0;
4) for (i = 0; i <= high; i++) {
6)    if(HOSTS [i].util > max. util)
7)         max.util = HOSTS [i].util
8)        addr= HOSTS [i]. util
8)     }
9)  SORT (HOSTS [addr].VM)
10) max_util =HOSTS[addr].VM[0]. util
11) for (int j=0;j<HOSTS[addr].VM.length;i++)
12)      {
13)     if (HOSTS [addr].VM[i].util < max_util)
14)         {
15)         Max_util = HOSTS [addr].VM[i].util
16)          index=HOSTS [addr].VM[i]
17)       }
18)    }
19)  return (index);
```

Fig. 2 shows AH1, AH2, AH3, AH4, AH5, AH6, and AH7 has the running VM's instances. After receiving AH details from all AHs, the MH here, AH6 checks its own CPU utilization with the remaining AHs utilization and initiates the process of VM migration considering all AHs current CPU utilization. The VHM at AH after receiving AH detail initiates the process of finding AH that has maximum CPU utilization and minimum CPU utilization from the set of AHs. From Table II it found that the AH4 has maximum CPU utilization[18-19].

The VM placement is done on the server if it satisfies following conditions.

- The server has minimum CPU utilization.
- The selected VM should not exceed the resource capacity on placing to the server having minimum CPU utilization.

The VHM imitates the trigger to find the minimum and maximum CPU utilized AH. Once the AH with the highest utilization found; the VHM establishes the connection with such AH and retrieves a list of VM instances. The list of VM instances sorted as per resource consumption, the VM with maximum resource consumption at current instance get migrated to the minimum CPU utilized server.

## 4. Result and Discussion

The proposed solution is tested considering hardware configuration as in Table I. The KVM/QEMU hypervisor as the default hypervisor. Ubuntu 14.04 as an operating system, Java 1.6 as a software platform and the libraries JNA, Libvirt for developing proposed systems. Network files sharing (NFS) used to share disk image of VM. Each AH installed with the VM as in Table 2. The random server from the set of AHs selected as the MH. Each AH configured with MH address. The AH starts sharing its detail information to MH using the blow form.

| Address | No.of VM | CPU utilization | Status | Time |
|---|---|---|---|---|

The status flag provides identification of current MH. From Table 3, it founds that the AH1 currently acts as the MH. It starts receiving server detail from each AH. After receiving AHs detail, MH initiates the demon thread to store this AH information in the table. This stored information gets referred by other threads running at MH for future references.

**Table 1.** Hardware configuration of Servers

| Server Name | Core | RAM | operating system |
|---|---|---|---|
| AH1 | I5 1st gen. | 4GB DDR3, RAM. | Ubuntu 14.04 |
| AH2 | I5,6th gen | 4GB DDR3 ,RAM | Ubuntu 14.04 |
| AH3 | I5, 5th gen. | 4GB DDR3,RAM | Ubuntu 14.04 |

The MH initiates the call to VM placement after receiving all AHs detail. This VM placement thread called by MH at fixed interval to maintaining the AH in normal state. Here, the interval set to 2 minutes.

**Table 2.** Initial server configuration

| Server Name | No of VM | RAM(MB) | VCPU | Disk(GB) |
|---|---|---|---|---|
| AH1 | 1 | 512 | 1 | 1 |
| | 2 | 1024 | 1 | 2 |
| AH2 | 1 | 1024 | 1 | 2 |
| | 2 | 512 | 1 | 5 |
| AH3 | 1 | 512 | 1 | 2 |
| | 2 | 512 | 1 | 2 |
| | 3 | 1024 | 1 | 3 |

Table 3 shows the initial AH detail received at MH from the AH's. To do VM placement minimum and maximum CPU utilized AH need to identify.

**Table 3.** MH CPU utilization details towards MH.

| Server Name. | CPU utilization | Flag | No of VM |
|---|---|---|---|
| AH1 | 0.134 | 1 | 2 |
| AH2 | 0.176 | 0 | 2 |
| AH3 | 0.234 | 0 | 2 |

Fig. 3 shows the CPU utilization of the AH against the time period. From Fig.3, it found that over the time period the CPU utilization is reduced.

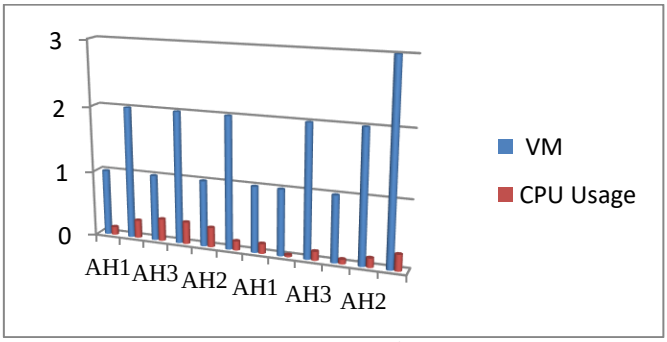


**Fig.3:** CPU utilization of server

From Table 4 the VM placement thread finds the AH3 and AH2 as the maximum utilized server and minimum utilized server. The VHM running at MH, instructs LRM running at AH3 to list a number of running VM instances. VHM uses this list to find the VM with maximum utilization and gives instruction to of AH3's LRM to establish a connection with AH2. The AH3 migrate VM to AH2. Table 4 shows updated AH utilization of each host after migrating VM from the host table associated to MH.

**Table 4.** MH CPU utilization details after VM migration.

| Server Address. | CPU utilization | Flag | NVM |
|---|---|---|---|
| AH1 | 0.136 | 1 | 2 |
| AH2 | 0.212 | 0 | 3 |
| AH3 | 0.194 | 0 | 1 |

In comparison with the proposed solution with existing solution it finds that it avoids central failure at same time the proposed ap-

proach AH repeatedly makes call to VM placement to avoid over utilization of the any AH. Here using this approach we are able to resolve the case of same host identification by multiple hosts.

## 5. Conclusion

The paper discussed the live VM migration for the decentralized cloud environment wherein each server configuration with LRM and VHM to manage local and other server resources. The VHM gets activated only when the current AH acts as MH. This categorization of servers in AH and MH helps in avoiding centralized server failure. The proposed hybrid decision policy towards MH avoids the problem of same server identification by multiple servers during VM placement.

## Acknowledgement

We would like thanks to the Doctoral committee and all the faculty members at Koneru Lakshmaiah Education Foundation and Sinhgad Academy of Engineering to support this work.

## References

[1] "National Institute of Standards and Technology | NIST." [Online]. Available: https://www.nist.gov/. [Accessed: 29-Dec-2017].

[2] A. Shribman and B. Hudzia, "Pre-copy and post-copy VM live migration for memory intensive applications," *Lect. Notes Computer. Science (including Subser. Lecture Notes Artificial Intelligence. Lecture Notes Bioinformatics)*, vol. 7640 LNCS, 2013, pp. 539–547, available online: https://link.springer.com/chapter/10.1007/978-3-642-36949-0_63: 28.02.2015.

[3] "AWS suffers a five-hour outage in the US News Datacenter Dynaimcs."[Online].Available:http://www.datacenterdynamics.com/content-tracks/colo-cloud/aws-suffers-a-five-hour-outage-in-the-us/94841.fullarticle. [Accessed: 04-Jan-2018].

[4] R. Benali, H. Teyeb, A. Balma, S. Tata, and N. Ben Hadj Alouane, "Evaluation of traffic-aware VM placement policies in distributed cloud using Cloud Sim", *Proceedings of the conference Proc. - 25th IEEE Int. Conf. Enabling Technology Infrastructure. Collab. Enterp. WETICE 2016*,pp.95–100, http://ieeexplore.ieee.org/document/7536438/.

[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", *Proceedings of the conference Ninth Int. Conf. Front. Comput. Sci. Technol.*, pp. 364–369, https://doi.org/10.1109/FCST.2015.41.

[6] E. Feller, C. Morin, and A. Esnault, "A case for fully decentralized dynamic VM consolidation in clouds",*Proceedings of the conference Cloud Com 2012 - Proc. 2012 4th IEEE Int. Conf. Cloud Comput. Technol. Sci.*,pp. 26–33, https://doi.org/10.1109/CloudCom.2012.6427585.

[7] D. Grygorenko, S. Farokhi, and I. Brandic, "Cost-aware VM placement across distributed DCs using Bayesian networks", *Proceedings of the conference Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9512, pp. 32–48, https://doi.org/10.1007/978-3-319-43177-2_3.

[8] M. Pantazoglou, G. Tzortzakis, and A. Delis, "Decentralized and Energy-Efficient Workload Management in Enterprise Clouds", *IEEE Transactions on Cloud Computing*, vol. 4, no. 2, 2016,pp. 196–209,available online: http://ieeexplore.ieee.org/document/7180318/,last vist:21:7:2017.

[9] Z. Bagheri and K. Zamanifar, "Enhancing energy efficiency in resource allocation for real-time cloud services", *2014 7th Int. Symp. Telecommun. IST 2014*, pp. 701–706, https://doi.org/10.1109/ISTEL.2014.7000793.

[10] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya, "An algorithm for network and data-aware placement of multi-tier applications in cloud data centers", *Journal of Network and Computer Applications*, vol. 98, no. September 2017, 2017,pp. 65–83, available online:https://dl.acm.org/citation.cfm?id=3164305,last visit:11:02:2018.

[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. 9,2016, pp.1–9,available online:http://thesai.org/Publications/ViewPaper?Volume=7&Issue=9&Code=ijacsa&SerialNo=1,last visit:12:02:2018.

[12] Y. Zhao and W. Huang, "Adaptive Distributed Load Balancing Algorithm Based on Live Migration of Virtual Machines in Cloud", *Proceedings of the conference 2009 Fifth Int. Jt. Conf. INC, IMS IDC*, pp. 170–175, https://doi.org/10.1109/NCM.2009.350.

[13] X. Fu and C. Zhou, "Virtual machine selection and placement for dynamic consolidation in Cloud computing environment", *Frontiers of Computer Science*, vol. 9, no. 2, 2015,pp. 322–330, available online:https://link.springer.com/article/10.1007/s11704-015-4286-8,last visit:11:2:2018.

[14] X. Y. Wang, X. J. Liu, L. H. Fan, and X. H. Jia, "A Decentralized Virtual Machine Migration Approach of Data Centers for Cloud Computing", *Math. Probl. Eng*., vol. 2013, pp. 10, available online: https://www.hindawi.com/journals/mpe/2013/878542/,last vist: 12:02:2014.

[15] X. Meng, C. Isci, J. O. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via VM multiplexing", *Proceedings of the conference 7th Int. Conf. Auton. Comput. - ICAC'10*, pp. 11, https://doi.org/10.1145/1809049.1809052.

[16] T. Daradkeh and A. Agarwal, "Distributed shared memory based live VM migration", *Proceedings of the conference IEEE Int. Conf. Cloud Comput. CLOUD*, pp. 826–830, https://doi.org/10.1109/CLOUD.2016.0116.

[17] Petter Sard, Benoit Hudzia, Steve Walsh, Johan Tordsson, Erik Elmroth. "Principles and Performance Characteristics of Algorithms for Live VM Migration", *ACM SIGOPS Operating Systems Review*, vol.49, no.1, 2015, pp.142-155, available online: https://dl.acm.org/citation.cfm?id=2723894, last visit: 12:02:2018.

[18] ANNABATTULA, J., KOTESWARA RAO, S., SAMPATH DAKSHINA MURTHY, A., SRIKANTH, K.S. and DAS, R.P., 2015. Underwater passive target tracking in constrained environment. Indian Journal of Science and Technology, 8(35), pp. 1-4.

[19] HUSSAIN, S.N. and KISHORE, K.H., 2016. Computational Optimization of Placement and Routing using Genetic Algorithm. Indian Journal of Science and Technology, 9(47),.