Scientific Journal of Impact Factor (SJIF): 5.71

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 5, Issue 04, April -2018

OPENSTACK BASED AUTO LIVE MIGRATION MECHANISM OF VIRTUAL MACHINE TO MANAGE LOAD BALANCING IN CLOUD COMPUTING

Mayuri B. Katosana[1], Prof. Hetal A. Joshiyara[2],

[1]*Department of Computer Engineering, L. D. College of Engineering*
[2] *Department of Computer Engineering, L. D. College of Engineering*

Abstract — *In cloud system, to achieve effective resource utilization to improve the system performance and to minimize the resource consumption, "Load Balancing" technique is used. Live Migration is used for load balancing. But, when the number of physical machines increases, manually initiated migrations can become problematic as it is also required to decide the factors like when to migrate and where to migrate. Therefore in this paper, we propose: "Virtual Machine Auto Live Migration for Load Balancing". We also take into account the user billing type (Hourly and Monthly) which highly affects on the cloud load of the datacenter. We have implemented our propose algorithm using OpenStack as it is an open-source, most used and rapidly developing cloud platform for the creation of IaaS platforms. We have tested our algorithm with real world work load traces.*

Keywords-*cloud computing; OpenStack; virtualization; load balancing; automatic VM migration*

I. INTRODUCTION

Cloud computing is a method of providing services to the user by using internet through web-based technology and applications. It offers services to users like infrastructure (IaaS-Infrastructure as a Service), platform (PaaS-Platform as a service) and application or software (SaaS-Software as a Service). Our main focus of this paper is IaaS. IaaS is provision model in which physical resources can be provided to organization to support operations such as network, storage, memory and CPU. Cloud computing provides scalable, on-demand, cost-effective, device independent and reliable services to its user. Organization purchased their computing infrastructures to run a real-time application, such as financial analysis, distributed data processing, real-time databases, etc. They can access their files and data on any computer or at any place through the Internet.

Cloud service provider allocates resources to customers in a way that specifies the required Quality of Services (QoS) determined by cloud providers through the Service Level Agreement (SLA). In cloud, SLA is defined as two-sided contract between the cloud service providers and its users. If requested CPU demand exceeds the available CPU capacity on host, then host is overloaded. High QoS and reduction the SLA violation depends on efficient management of load imbalance problem. So, some VM needs to be migrated to other host to minimize the SLA violation rate. Hence, a fit methodology is required by cloud service provider to manage the resource in dynamic way as to guarantee QoS for the users.

Cloud service providers provides options of billing of user's services. So in IaaS user can choose their virtual machine to be charged for their hourly usage or they can go with monthly plan. Usually user who have can keep their VM normally shutoff and start only when they have some need of it, chooses hourly plan. On the other side users who run website or something that needs to work 24x7 for whole month, goes with monthly plan. So hourly users are the ones who plays important role in cloud load imbalance.

In cloud computing because of the fickle nature of the cloud users, all cloud hosts do not have an equal amount of load. So it will degrade the performance of the system. Load balancing is a key technology to maintain the imbalance load of a particular host. It helps to distribute workload across multiple hosts to ensure that no single host gets overloaded. It is also helpful for proper utilization of resources. There are various types of load possible in cloud computing like CPU, memory, storage and network load. But, the question is how to improve utilization of resources and manage the imbalance load of the system.

Live Migration is useful for effective utilization of resources. Live migration is a technique which migrate the VM from one host to another host without disconnecting the user's application. But, the problem is that manually initiated migration become a problematic as it is also required to decide the factors like when to migrate and where to migrate. Also it can be inaccurate and untimely. So in this paper we propose an Auto Live Migration Mechanism for load balancing for cloud data centers.

OpenStack is open source cloud software tool for building and managing cloud computing platforms for public and private clouds. So, our proposed algorithm is implemented in OpenStack platform. So using this tool we can analysis how our algorithm acts in real cloud environment. Our Auto Live Migration algorithm consists an overloading host detection algorithm using Simple Linear Regression method to determine two utilizations threshold.

The rest of this paper is organized as follows. Section 2 discusses related works. Section 3 introduce proposed system architecture, predictive method and proposed algorithm. We described the adjustment to the experimental environment, performance metrics and the results in section 4. Section 5 presents the conclusion and future work.

II. RELATED WORK

Tiwari et al. [1] presented Dynamic Weighted Live Migration (DWLM) to manage the load imbalance problem. In this approach, author has focused on maximizing the utilization of VMs CPU capacity. In DWLM algorithm, to checks the VMs availability and available jobs in queue which are not allocated in VMs due to overload condition of VMs. So, load imbalance management policy managed by threshold model. Generally, CPU capacity is between 0.0 to 1.0. But proposed approach used threshold value for CPU capacity is 0.8. If utilization of CPU higher than the threshold value, then DWLM migrate the VMs for load balancing.

Farahnakian et al. [2] LiRCUP proposed for energy-efficient consolidation of VM in cloud data centers. The proposed method predicted utilization of CPU based on Linear Regression technique. This technique used the last utilization over one hour ago and approximated a function. If the prediction value of CPU is greater than the current utilization capacity then the host will be marked as overloaded. Therefore, some VMs should migrated from overloaded host to maintain SLA level.

Abdelsamea et al. [3] proposed an algorithm using hybrid factors instead of single factor to enhance the VM consolidation. So, Multiple Regression Host Overload Detection (MRHOD) algorithm developed which uses multiple parameters such as CPU, memory and bandwidth for host overload detection. Regression based algorithms gives better predictions of host overloading since they are based on estimation of future CPU utilization. Once a host overload is detected, the next step is to select VMs to offload the host to avoid performance degradation. The virtual machine selection process uses three selection policies for effective utilization of virtual machine such as Minimum Migration Time (MMT), Random Choice (RC) and maximum correlation. After selecting VM for migration next step is where to place that VM. So, VM placement has been used for effective placement. This algorithm reduces the energy consumption and also utilizes all resources in an effective way. The reduction in energy consumption can be achieved by switching idle nodes to low power modes thus eliminating the idle power consumption.

Chen et al. [4] proposed Dynamic Resource Allocation (DRA) and Energy Saving algorithms. These two algorithms were implemented in an infrastructure platform based on cloud software OpenStack. OpenStack is open source cloud computing software platform. In order to status and power consumption of system were monitored through Power Distribution Unit (PDU). In DRA firstly upper limit and lower limit of resource utilization defined, when the resource utilization of a VM exceeds the upper limit then increase the resources of the VM. When the resource utilization of a VM is less than the lower limit than reduce the resources of the VM to release its idle resources. If the resources utilization is too high, we can merge the VMs to the other host via live migration and then shut down the host with no VMs to save energy.

Duggan et al. [5] focused on ideal time for a group of Virtual Machine to be migrated from an over-utilized host, depending on the current network traffic demand. Author have proposed a dynamic Reinforcement Learning (RL) network aware approach, enabling a single RL agent to learn the most opportune time to migrate a group of VMs, depending on the current utilization of a cloud's network resources. In this technique agent is getting rewarded or punished based on how the cloud system reacts to those action performed.

Forsman et al. [6] presented two strategies to balance the load in the system with multiple virtual machines (VMs) through automated live migration. Author has proposed *push* strategy and *pull* strategy based algorithm and also cost model for these algorithm has been proposed. When the *push* strategy is used, overloaded hosts try to migrate workload to less loaded nodes. On the other hand, when the *pull* strategy is employed, the light-loaded hosts take the initiative to offload overloaded nodes. A hot spot occurs when the performance of VMs degrades because the PM is unable to respond to the resource demand. The opposite situation, when resources on a PM are underutilized, is termed cold spot. This happens when the running VMs consume only a tiny fraction of the resources provided by the hosting PM. A hot spot is defined as a host that has a CPU load that exceeds a specific utilization threshold λ.

In *push* migration strategies, an overloaded PM plays the role of the seller. The VMs hosted on this PM constitute the auctioned items. The PM multicasts information about these items to potential buyers. The buyers may bid on one, several, or all auctioned VMs. The seller selects the best candidate from the bids and performs migration. When *pull*

strategy is used the seller is an underutilized PM that advertises free resources. The other PMs in the system are potential resource buyers that bid with the characteristics of their VMs as described above. The seller decides which bid it accepts and initiates the VM migration.

Khoshkholghi et al. [7] proposed several novel algorithms for the dynamic consolidation of VMs in cloud data centers. The aim of this paper is improve the utilization of computing resources and reduce energy consumption under SLA constraints regarding CPU, RAM, and bandwidth. Overloading host detection algorithm uses IWLR (Iterative Weighted Linear Regression) algorithm to dynamically determine the utilization thresholds for each of the three resources. Regression is a statistical method for quantitative data analysis that is used to predict the future values of data. The proposed algorithm uses a simple weighted linear regression to predict future host utilizations. When all the overloaded hosts are detected. Then, one or several VMs will be selected from each detected host using VM selection algorithm so that host utilization drops below the threshold. After detecting the overloaded hosts and selecting the adequate VMs, then in this step, VMs are required to place to the best destination hosts.

Beloglazov et al [8] there are very few open-source software systems implementing dynamic VM migration. In this paper, author has proposed an architecture and open-source implementation of OpenStack Neat, a framework for dynamic VM migration in OpenStack clouds. OpenStack Neat can be configured to use custom VM migration algorithms and transparently integrates with existing OpenStack deployments without the necessity of altering their configuration. It mainly contains three system components like global manager, local manager and data collector. Local manager who invokes overload host detection and VM selection algorithm. Global manager invokes VM placement algorithm for efficient placement of selected VM.

III. SYSTEM DESIGN

3.1. Proposed System Architecture

The target system is an IaaS environment, the system architecture consists of N heterogeneous physical nodes. Each node i is characterized by the CPU, RAM and bandwidth. Each physical nodes consist M heterogeneous VMs that are characterized by the CPU, RAM and bandwidth. The system architecture for cloud OpenStack software that comprises two main components is: [8]

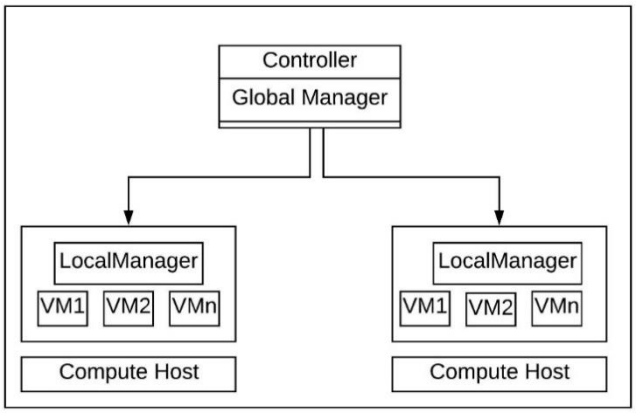

Figure 1. Proposed System Architecture [8]

3.1.1. Global Manager (Controller)

The global manager is installed on the controller host and it is responsible for making placement of VM decisions and initiating VM migrations. It takes requests from local managers then initiates the VMs migration process. [8]

3.1.2 Local Manager (Compute)

The local manager component is deployed on every compute host. it is responsible for monitoring the host's load continuously. It initiates overload host detection algorithm and VM selection algorithm. [8]

3.2. Proposed Structure and Algorithms

Our proposed Auto Live Migration mechanism includes three algorithms, as follows:

- **Overloading Host Detection Algorithm:** it defined as if host should be considered overloaded, in this case VMs are allocated to another host.
- **VM Selection Algorithm:** it selects the most suitable VMs to be migrated from overloaded host
- **VM Placement Algorithm:** it discovers the most suitable destination host for the selected VMs

3.2.1. Overloading Host Detection Algorithm

In the cloud environment, there are mainly two types of virtual machine users.

1. Instance with hourly billing
2. Instance with monthly billing

So, in general if user does not need the instance for all of the time in whole month, user would go with hourly plan as the instance will be charged as per hour usage. So this clearly indicates that hourly instance will not have constant uptime for whole month.

So we have exploited this behavior of instance type to predict the future host load condition.
The overloading host detection algorithm consists of mainly two type of regressions. First one is for hourly instance status prediction. This regression does the hourly instance status prediction from the past instance activity. This will be useful in host total load prediction later.

The second regression is for regression of prediction of monthly instances sum of load. We could have also used prediction of monthly instances separately but that is not that required as monthly instances does not have many start stops as the hourly users. So prediction will give good results for sum of all the monthly instances of host too.

1) **Simple Linear Regression (SLR)**

Regression is a statistical method for quantitative data analysis that is used to predict the future values of data. The proposed algorithm uses a simple Linear Regression to predict future host utilizations. The simple regression line is shown is as below: [7]

$$Y = \beta_0 + \beta_1 X \quad (1)$$

Where Y is the dependent variable and X is the independent variable. β_0 And β_1 are regression coefficients and are derived from the least squares technique as follows: **[7]**

$$\widehat{\beta_0} = \bar{Y} - \widehat{\beta_1}\bar{X} \quad (2)$$

$$\widehat{\beta_1} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - X)^2} \quad (3)$$

Where $\bar{X}$ and $\bar{Y}$ are the means of X and Y observations, and $\widehat{\beta_0}$ and $\widehat{\beta_1}$ are estimations of β_0 and β_1.
Where x_i and x_n are i^{th} and the last observations. SLR uses K iterations to detect K future values of the host utilization. In the proposed algorithm, we set k = 2 which means SLR predicts two future values. [7]

$$\widehat{y_1} = \beta 0 + \beta 1 x_n \quad (4)$$

$$\widehat{y_2} = \beta 0 + \beta 1 \widehat{y_1} \quad (5)$$

In this case, there is the following:
x_n is upper threshold, if $c \cdot \widehat{y_1} \geq 1$
x_n is pre threshold, if $c \cdot \widehat{y_2} \geq 1$
Where c is the intensity constant.

Algorithm Overloading Host Detection Algorithm
Input: host
Output: Prediction of Host (Overload, underpressure, normal)

```
first_pred_cpu, second_pred_cpu ← Linear_Regression (monthly vms last n cpu)
first_pred_ram, second_pred_ram ← Linear_Regressin (monthly vms last n ram)
for hourly_vms in host do
    get_instance_status
    get_instance_actions_list
    next_instance_action_with_time ← Linear_Regression (instance_action_list)
    if next_instance_status != instance_status then
        if next_instance_status == ACTIVE then
            positive_changed_instance ← get average of active utils for last n records
        else
            negative_changed_instance ← get last utils of instance
    else
        if instance_status == ACTIVE
            sudo_monthly_instances ← get_last_n_utils_records
```



```
    if positive_changed_instance then
        first_pred_cpu += positive_changed_instance [cpu]
        second_pred_cpu += positive_changed_instance [cpu]
        first_pred_ram += positive_changed_instance [ram]
        second_pred_ram += positive_changed_instance [ram]
    elif  negative_changes_instance
        sirst_pred_cpu   -= negative_changed_instance [cpu]
        second_pred_cpu -= negative_changed_instance [cpu]
        first_pred_ram -= negative_changed_instance [ram]
        second_pred_ram -= negative_changed_instance [ram]
if ( first_pred_cpu or first_pred_ram) >= 100 then
    return host_overload (2)
elif (second_pred_cpu or second_pred_ram) >= 100 then
    return host_under_pressure (1)
else
    return host_normal (0)
```

2) VM Selection Algorithm

Once overloaded hosts are detected. Then, it is necessary to determine which VMs are the best to be migrate from the host. This problem is solved by VM selection algorithm. So, that hosts utilization drops below the threshold. This algorithm selects VMs with the minimum amount of RAM and maximum CPU utilization so live migration time of VMs is minimized. [8]

3) VM Placement Algorithm

VM Placement is done with Best Fit Decreasing algorithm. Algorithm first shorts the VMs into decresing order by their RAM and that available hosts into increasing order by their CPU. The algorithm runs over each VMs to find the best host (with best CPU) that specifies the VM's requirement. [8]

IV. PERFORMANCE EVALUATION

4.1. Experimental Setting

Table 1. Experiment Specification

Name	Details
Controller1	2 vCPUs, 7.5 GB memory, 40 GB disk
Compute 1	2 vCPUs, 4 GB memory, 40 GB disk
Compute 2	2 vCPU, 4.5 GB memory, 40 GB disk
Compute-3	2 vCPUs, 8 GB memory, 80 GB disk

The experiment setup is used for performance evaluation of the system, consisted of the hardware specifications as shown in Table 1. First, we installed Ubuntu 16.04.4 LTS (Xenial Xerus) operating system on all PMs. The Ubuntu installation followed the standard process described in detail in the OpenStack newton installation guide. We deployed the OpenStack component on a distributed model.

- **Controller Node:** Controller node represents the global state and interacts with all other components. An API Server acts as the web services front end for the compute nodes. In this node, we deployed the basic component as Nova-API, Nova-scheduler, Neutron, Keystone service, Glance service, MySQL server and others. We have to mention that the controller node runs all service except for Nova-compute. Therefore, controller node does not host any VMs. Then, we added the additional component that is responsible for making VM placement decision and initiating a VM migration. The global manager would call the VM placement algorithm for the requested list of migratory VMs. Hence, the algorithm produces an appropriate map between the VM and the PM. After that, the result's map would be sent to the OpenStack Nova-API to perform the migration. The global manager exposes a REST web service (REST-API) for processing VM migration requests that have been sent by local managers.

- **Compute Node:** Compute nodes are responsible for running the customers' instances and providing the physical requirements. We have installed the Nova-compute, Neutron packages in these nodes. The components in the proposed distribution system of compute nodes are responsible for making the local decision by executing periodically a function to determine whether it is necessary to reallocate VMs of the PMs to avoid overloaded machines.

- **Load Generation:** we are giving 2 types of CPU load.

1. Apache Benchmark
2. Stress-ng

So, the VM has stress-ng installed by default and also it has apache webserver running on it.
Now for load generation, we choose any one or two VM from the host and give a sudo random, almost increasing load by any of above load generation technique for every 30 minutes. And HOURLY VMs are being on and off as to mimic their actual behavior in cloud environment.

4.2. Performance Metrics

For effective performance evaluation, some performance metrics are used to evaluate the efficiency of proposed algorithm which are as below:

- **Overload Time Fraction (OTF):** It calculate the overall QoS delivered by the system.

$$\text{OTF} = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{s_i}}{T_{a_i}} \tag{6}$$

- Where N is the number of hosts,
- T_{s_i} is the total time during which host i has experienced the utilization of 100% leading to SLA violation
- T_{a_i} is the total of host i being in the active state (serving VMs)

4.3. Results and Analysis

The comparison of existing overload host detection values and proposed overload host detection values for the CPU are listed below.

Table 2. Comparison of Host Utilization Prediction

Actual Host Utilization	Prediction with Existing Algorithm	Prediction with Proposed Algorithm
73	73	73
82	80	80
85	84	76
72	87	87
76	75	75
73	77	77
87	80	88
86	83	90
82	84	88
86	81	85
100	79	100
100	83	98
98	86	88
89	88	90
90	89	95
97	89	99
94	93	93
100	94	100
100	97	98
92	98	91

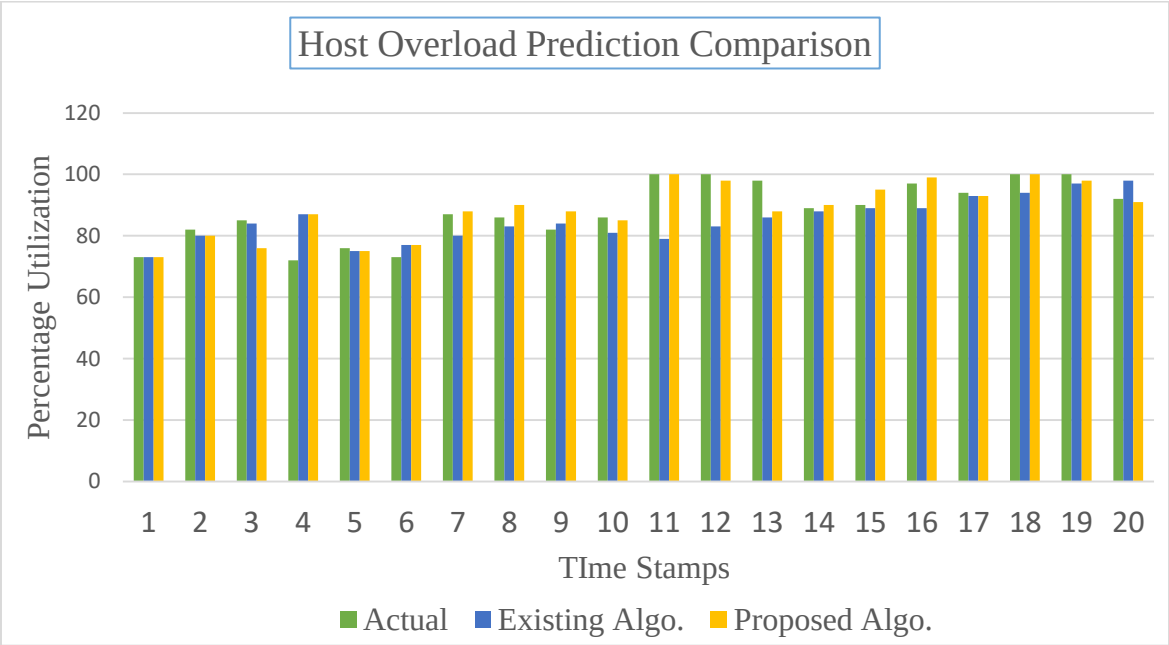

Figure 2. Chart of Host Overload Prediction Comparison

Here we have chosen a host utilization which has 4 VMs running on it. Two of them are MONTHLY and remaining two are HOURLY. The comparison graph shows our algorithm performs same normally, but in case of HOURLY VM being turned on or off, our algorithm gives good result, while existing algorithm [7] is performing poor at this.

Now for the performance evaluation, we have measured Overload Time Fraction (OTF) for each algorithm. It is defined by the fraction of time during which active hosts have experienced the CPU utilization of 100%. The reason behind the OTF metric is the observation that if a host serving applications is experiencing the 100% utilization, the performance of the applications is bounded by the host's capacity; therefore, VMs are not being provided with the required performance level.

Here we have used 3 different threshold values for automatic live migration. We can see in table 3, as we increase the threshold the OTF value increases rapidly and the user's SLA violation also increases by this. So one might think that keep the threshold low enough so user's SLA don't violate. But by doing so, we will end up using average low utilization of the host. So this costs more to the cloud provider. So only way to keep SLA of users at acceptable level along with high average utilization is to predict the host utilization more accurately. So for existing algorithm we can decrease the OTF value to 10.7% to 3.42% by reducing the threshold to 100% to 80%. Our proposed algorithm we have achieved 3.67% OTF with 90% Threshold and as low as 1.31% at the 80% Threshold.
Here is the comparison of two algorithms as per their OTF.

Table 3. Comparison of OTF value for different Threshold

Threshold	Existing Algorithm [7]	Proposed Algorithm
100%	10.70%	6.33%
90%	6.43%	3.67%
80%	3.42%	1.31%

V. CONCLUSION

Cloud Computing is an emerging field and demand of cloud computing is rapidly increasing. Along with this load balancing has become necessity for cloud computing. In this thesis we have studied auto live migration techniques for load balancing. For auto live migration host overload prediction is crucial task. We proposed an auto live migration technique that considers users billing type as it highly affects cloud load.

We have designed an algorithm for overloaded host prediction which takes advantage of the user billing type. We have implemented our algorithm in python language and test it for OpenStack environment. OpenStack deployment is done on

real hardware. We have also implemented the auto live migration technique proposed in [7]. By the testing we have shown that our proposed algorithm gives 3.67% OTF value compare to 6.43% for existing algorithm.

In future, we can also use RNN LSTM or ARIMA model to more accurately predict the host utilization instead of linear regression in our proposed method. We can also do host underutilization prediction more accurately with using concept of our proposed method.

REFERENCES

[1] P. K. Tiwari and S. Joshi, "Dynamic weighted virtual machine live migration mechanism to manages load balancing in cloud computing," IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Chennai, pp. 1-5, 2016.

[2] F. Farahnakian, P. Liljeberg and J. Plosila, "LiRCUP: Linear Regression Based CPU Usage Prediction Algorithm for Live Migration of Virtual Machines in Data Centers," 2013 39th Euromicro Conference on Software Engineering and Advanced Applications, Santander, pp. 357-364, 2013

[3] Abdelsamea, A., A. A. El-Moursy, E. E. Hemayed, and H. Eldeeb, "Virtual machine consolidation enhancement using hybrid regression algorithms", Egyptian Informatics Journal: Elsevier, 2017.

[4] C. C. Chen, P. L. Sun, C. T. Yang, J. C. Liu, S. T. Chen and Z. Y. Wan, "Implementation of a Cloud Energy Saving System with Virtual Machine Dynamic Resource Allocation Method Based on OpenStack," 2015 Seventh International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), Nanjing, pp. 190-196, 2015.

[5] M. Duggan, J. Duggan, E. Howley and E. Barrett, "An Autonomous Network Aware VM Migration Strategy in Cloud Data Centres," 2016 International Conference on Cloud and Autonomic Computing (ICCAC), Augsburg, , pp. 24-32, 2016

[6] Mattias Forsman, Andreas Glad, Lars Lundberg, Dragos Ilie, Algorithms for automated live migration of virtual machines, In Journal of Systems and Software, Volume 101, Pages 110-126, 2015.

[7] M. A. Khoshkholghi, M. N. Derahman, A. Abdullah, S. Subramaniam and M. Othman, "Energy-Efficient Algorithms for Dynamic Virtual Machine Consolidation in Cloud Data Centers," in IEEE Access, vol. 5, pp. 10709-10722, 2017.

[8] Beloglazov, Anton & Buyya, Rajkumar. (2014). OpenStack Neat: A framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds. Concurrency and Computation: Practice and Experience. 27. . 10.1002/cpe.3314.

[9] Vikas Malik C R Barde. Article: Live Migration of Virtual Machines in Cloud Environment using Prediction of CPU Usage. International Journal of Computer Applications 117(23):1-5, May 2015.

[10] C. T. Yang, Y. T. Liu, J. C. Liu, C. L. Chuang and F. C. Jiang, "Implementation of a Cloud IaaS with Dynamic Resource Allocation Method Using OpenStack," 2013 International Conference on Parallel and Distributed Computing, Applications and Technologies, Taipei, 2013, pp. 71-78.

[11] Al-moalmi Ammar, Juan Luo and Zhuo Tang, "An Anti-Overload Model for OpenStack Based on an Effective Dynamic Migration," KSII Transactions on Internet and Information Systems, vol. 10, no. 9, pp. 4165-4187, 2016.

[12] A. Upadhyay and P. Lakkadwala, "Migration of over loaded process and schedule for resource utilization in Cloud Computing," 2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), Noida, India, pp. 1-4, 2015

[13] Jasmin James, Dr. Bhupendra Verma , "Efficient VM Load Balancing Algorithm for a Cloud Computing Environment," International Journal on Computer Science and Engineering (IJCSE),2012.

