University College of Takestan

Available online at **http://UCTjournals.com**

UCT Journal of Research in Science, Engineering and Technology

UCT. J. Resea. Scien. Engineer.Techno. (UJRSET)

21-29 (2016)

Volume 4 ,Issue 2

ISSN 2382-977X

New Method for Load Balancingin Cloud Computing

Younes Ranjbar haghighi and Ali Ghaffari*

Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran.

Original Article:

Received 03 Jan. 2015 Accepted 12 April. 2015 Published 29 June. 2016

ABSTRACT

Internet, since the beginning of its work, has undergone many changes one of the latest changes is how the internet cloud computing. New technologies cloud computing offers because of features all kinds of facilities to the users as a service. Each evolution, change and novel concept in the world of technologies has its own problems and complications. Accordingly, benefiting from cloud computing is no exception to this rule and it has challenged researchers and proponents in this research domain. Indeed, some major challenges in cloud computing are: load balancing, safety, reliability, ownership, data backup, data portability and supporting several platforms.

One challenge for such matters in the field of cloud computing is load balancing optimization in the cloud. The so-called cloud computing, including virtualization, distributed computing, networking, software and Web services. With respect to the ever increasing significance of load balancing in cloud computing the researchers in this paper intended to improve load balancing by using a novel method. The related studies were reviewed, evaluated and compared with each other. The efficiency of the proposed method was analyzed and compared with those of other studies. The results of the present study revealed that the proposed method is better than other dynamic virtual machine (VM) consolidation algorithms in terms of reducing SLA (service levels agreement) violation and the amount of transmitted data volume transmission has to present a better performance than other methods.

Keyword:

Cloud computing, load balancing, load balancing algorithms, SLA

*** Corresponding author: Ali Ghaffari**

Peer review under responsibility of **UCT Journal of Research in Science, Engineering and Technology**

INTRODUCTION

Cloud computing emerged a model for providing the resources required for customers, similar to other industries (water, electricity, gas and telephone).The user gain access according to their needs, regardless of where or how a service is delivered.The Sample a variety of computing systems has been proposed that some of these computing systems including: cluster computing, grid computing and cloud computing recently. Three fundamental services to provide by cloud computing architecture, according with customer requirements information technology.[1-3]

One outstanding challenge facing researchers in cloud computing is to maintain optimal load balance in cloud computing. Indeed, cloud computing refers to virtualization, distributed computation, network, software and web services. A cloud consists of several elements, namely clients, data centers and distributed servers. The following issues and challenges are involved in a cloud: error tolerance, high accessibility, scalability, flexibility, reduction of overload for users, reduction of ownership cost, providing services to requests, etc.[4-7]Focusing on these issues and trying to sort them out can result in an efficient load balancing algorithm. Load balancing refers to the process of distributing load among different nodes broadcast in a system in order to improve resource efficiency and job response time. As a result, conditions in which some nodes are overloadedwhile other nodes are unloaded or lightly-loaded should be avoided. Furthermore, it should be guaranteed that all the system processors and network nodes should have equal amounts of job at any moment. [8-12]

With respect to the explosion of cloud computing networks and their ever-increasing developments, energy consumption in data centers increases remarkably.hence, this condition can be regarded as a critical issue and concern for industry and society. The increase in energy consumption not only raises energy cost but also increases the amount of carbon distribution which can eventually reduce cloud providers' profits and can be harmful for the environment [13-14].It should be pointed out that since the scale and complexity of distributed cloud computing systems are overwhelmingly high, centralized assignment of jobs to specific servers is impossible. Hence, an effective distributed solution is required which should be proposed to service providers [15].Numerous scheduling algorithms have been proposed for taking load balancing into consideration. Namely, Honeybee Foraging algorithm, Biased Random Sampling, Active clustering, OLB+LBMM ،Min-Min, etc. are some algorithms which were aimed at maintaining load balance. Indeed, a specific algorithm is usually used based on the need and requirement. Nevertheless, it should be noted that cloud computing covers a wide range of areas; hence, methods are desirable which are appropriate for different contexts and environments and which can reduce cost.[16-19]Inasmuch as load balancing is of high significance in cloud data centers, different methods have been proposed. Nevertheless, it should be noted that each method considers only a select number of criteria and can only optimize particular parameters. Thus, the best approach is to use earlier methods, develop and optimize them so as to obtain more efficient results.[20]Over the past years, different methods have been proposed for maintaining load balance and numerous research papers have been published on this issue. In each proposed method, algorithms were designed based on specific features and certain criteria were taken into consideration by them[21]. For example, a load balance scheme was proposed in [22-23] which was based on virtual machine migration, it was aimed at ensuring error tolerance. An algorithm was proposed in [24] which was based on available methods, it was intended to reduce the response time. In [25-26] an algorithm was proposed where load balanced was used to maximize operational throughput. In [27], optimal load balancing was designed to prevent deadlock in cloud.

In this paper,because ofvery importance process load balance in the cloud computing, using a new method to compare the existing methods and analysis methods is presented. The proposed method rather to other algorithmscombines dynamic virtual machines than both in terms of reducing the amount of violation of the SLA and the volume of data transmission, performance is better than other methods.The next discussion paper includes proposed algorithm calculations and performance of the proposed algorithm deal with, and finally conclusions and presented in the end of this article.

2. The Proposed Method

As mentioned above, load balance is one of the basic challenges in cloud computing. Sorting out this demanding issue requires dynamic and equal distribution of the local workload within all nodes so as achieve user satisfaction and high rate of resource utilization. As a result, equitable and efficient allocation of computational resources is guaranteed.

The purpose of the present study was to compare different load balance algorithms in cloud computing.As mentioned in the review of the related works, none of the above-mentioned algorithms can take all the criteria into consideration. Hence, due to the lack of such algorithms, designing an adaptive algorithm for heterogeneous environments which can reduce cost is of high significance. In this paper, all the important and effective factors in cloud computing systems were considered to design an algorithm which can reduce cost and system errors and enhance users' satisfaction.

When a decision is made to transmit a load on a network node, many important parameters should be taken into consideration. In case the objective is to appropriately distribute the main load on a cloud, this can be considered as a particular state of load distribution on a cloud. In general, algorithm can be used for distributing any type of load on the network such as program migration, code migration and even operating system migration. The concept behind algorithm is straightforward; that is, algorithm operates by considering a higher coefficient for those parameters which are more important.In some parameters, the nodes of a cloud network can simply be informed about the conditions of other nodes.

Through a simple signaling, for example, delay can be measured. In this algorithm, keeping only little information about neighboring nodes is required, when compared with the mass of information exchanged in the cloud context, the amount of information required about neighboring nodes is very little. In the proposed method, the focus will be more on information about neighboring nodes which is locally available and there is no need to have an overview of the network cloud. Cloud network is a vast network, if one wants to have comprehensive information about cloud network, huge memory space and large masses of data to be transmitted and received are required. Thus, in the algorithm proposed in this paper, local information about nodes, especially neighbors is used and each node transmits code or data to another node based on the conditions of the neighboring nodes. In case a node has a problem, other local nodes easily become informed of it and consider it in their decision-making process. Thus, it can be argued that each node does not need a lot of query and exploration in order to make a decision and easily obtains the required information for making the decision.

In the scheme proposed in this paper, the relations which will be mentioned below and effective factors in load balance and appropriate standard coefficients will be used to obtain desirable results. One factor in cloud computing which has a notable impact on load balance is the number of available neighbors in each node. According to the cloud model, it is assumed that the more the number of neighbors, the higher the load balance will be in the system. Hence, in the following equation (1), this factor (the number of neighbors) is considered with the coefficient a. Another important factorin cloud computing is related to the energy of each node; the higher is the energy of each node, it can be assumed that the node can be optimally used to maintain load balance. Hence, in the equation (1) below, node energy is denoted by b coefficient which is summed up by the previous factor (the number of neighbors). One more factor which can be used is the major load in each node. The lower the amount of major loads in each node, the higher the load balance. Consequently, this factor is denoted by the coefficient c in the denominator of the following equation (1).

Another factor which can play a significant role in load balance is the response time of each node to the transmitted request. For measuring response time, the elapsed time from transmitting a data to the related node and its reception by the node should be measured. This procedure is conducted for all the available nodes in the cloud system and its different neighbors. By comparing the response times for all the nodes, the node with lower response time will be used for load balancing. This factor is denoted by the coefficient d and is included in the denominator of the equation (1)so that it can contribute to maintaining a better load balance in cloud computing.The proposed equation (1) is as follows:

$$\frac{nn*a+en*b*x1}{ld*c*x2+rt*d*x3} \text{(equation 1)}$$

a+b+c+d=1 (equation 2)

In this equation, nn refers to the number of nodes, en stands for the amount of energy, ld denotes the load mass in each node and rt refers to the response time of each node in the network. Also, a, b, c and d stand for the impact factor of each parameter. Each impact factor or coefficient is a number ranging from zero to one and the sum of all coefficients should be equal to one. (equation (2)). Moreover, x1, x2 and x3 are parameters which included in the equation (1), they are aimed at maintaining a balance among the available parameters. In this paper, the parameter related to the number of neighbors is regarded as the basic parameter and the other parameters are measured in relation to this basic parameter. In other words, this basic parameter is used for integrating different available parameters in this equation. The values of these parameters are determined based on the expected values for the main parameters.

The equation mentioned above is highly flexible since other criteria can be added to it. Expectations from cloud computing vary in different environments and under different conditions. Thus, the modifications made in the above-mentioned parameters can be added to the equation or new parameters can be added to it. In any case, initially, the impact factors of different parameters are determined. However, for determining the precise magnitude of integration parameters, limits of parameters should be specified.

Other parameters are parallel to customer response time. For example, energy or the amount of load is not directly related to response time. Nevertheless, it should be noted that, in the long run, considering them might have a desirable impact on response time.

3. Computationalefficiencyof the proposed method

For finding hosts which are in the overload condition, the local regression proposed in the paper [21] was used. The rationale behind local regression is attributed to simple models in which local data subsets are used to produce a curve to estimate major data. For each observations such as (x_i ,y_i), weight function (tricube) is used to obtain neighboring weight (equation 4).

$$\omega_i(x) = T\left(\frac{\Delta_i(x)}{\Delta_{(q)}(x)}\right) \qquad \text{(equation 4)}$$

In this equation, $\Delta_i(x)$ denotes the distance from x to x_I and $\Delta_{(q)}(x)$ refers to the sum of these distances in the ascending manner. Hence, the neighborhood weight for(x_i,y_i) is defined as $\omega_i(x)$ which is obtained through equation (5) below.

$$\omega_i(x) = T\left(\frac{\Delta_i(x)}{\Delta_{(q)}(x)}\right) \qquad \text{(equation 5)}$$

For each x_i, $\Delta_i(x) < \Delta_q(x)$ and q determines the number of local data subsets around x. The size of the subset is defined by a parameter known as bandwidth. For example, if the value of the polynomial is equal to one, then the function will be of the family y=a+bx. The values of a and b are obtained by minimizing the equation (6) given below:

$$\sum_{i=1}^{n} \omega_i(x)\left(y_i - a - bx_i\right)^2 \text{ (equation 6)}$$

This method was used for obtaining a polynomial for observing k observations and the productivity of the processor was used. The polynomial was fitted only for one point, i.e. the last observation. According to the argument given in [22], first-degree polynomial leads to the distortion of the observation configuration. However, second-degree polynomial results in the loss of these distortions but results

in this state is higher than border bias. Hence, for reducing bias in borders, a first-degree polynomial was used. Assume that $x_k$is the last observation and x_1 is the K[th] observation from the border in the right side; in this condition, x_i is selected so that $x_1 < x_i < x_k$ and $\Delta_i(x_k) = x_k - x_i$. Thus, the weight function Tricube can be simplified as follows: $T^*(u).(1-|u|^3)^3$ for $1 \le u \le 0$ and the weight function is obtained as follows:

$$\omega_i(x) = T^*\left(\frac{\Delta_i(x)}{\Delta_{(q)}(x)}\right) = \left(1-\left(\frac{x_k - x_i}{x_k - x_1}\right)^3\right)^3 \quad \text{(equation 7)}$$

In the proposed LR algorithm, the method derived from Loess was used to find a line, i.e. $\hat{g}(x) = \hat{a} + \hat{b}x$ for each new observation. This line is used to estimate the next observation, namely, $\hat{g}(x_{k+1})$. If equation (8) is established, the host algorithm is identified as overload.

$$s.\hat{g}(x_{k+1}) \ge 1 \qquad x_{k+1} - x_k \le t_m \quad \text{(equation 8)}$$

In this equation, $s \in R^+$ standsfor the assurance parameter and t_m refers to the maximum required time for conducting a migration in the host. MMT method was used along this purpose.In MMT, the virtual machine V will be selected for migration provided that it relatively has the least migration time. The migration time is estimated by dividing the amount of used memory (RAM) on the available network bandwidth for the available host j. If V_i refers to the set of available virtual machines in the host j, then, the MMT method will select a virtual machine such as v for migration; this virtual machine should have the following conditions:

$$v \in V_j \left| \forall a \in V_{j'} \right| \frac{RAM_u(u)}{BW_j} \le \frac{RAM_u(a)}{BW_j} \quad \text{(equation 9)}$$

In this equation, $RAM_u(u)$refers to the amount of memory currently used by the virtual machine a andBW_idetermines the network bandwidth for the host j.

4.Result proposal algorithm

In this article, we analyzed the results of the algorithm for this purpose the following should be considered:

1. The live migration of virtual machines is one of the beneficial capabilities which is used for energy-efficient management of resources in the data center. In any way, this type of migration has a negative impact on the SLA of applications which are run. In this paper, algorithm number(1-5)was aimed at reducing the amount of SLA violation, reducing the number of migrations and reducing the amount of data to be transmitted.

For finding lightly loaded hosts, at first, equation (10) should be used to measure the threshold value for the host (UT_{HI}) H_I , this host is a combination of host processor utilization (U_{HI}) and the number of virtual machines within the host (VM_{SHI})HI.

$$UT_{Hi} = \alpha.U_{Hi} + \beta.VMS_{Hi}$$
$$\alpha + \beta = 1, 0 < \alpha < 1, 0 < \beta < 1$$

equation (10)

In equation (10), α stands for the processor utilization weight and β refers to the virtual machines within the host.

2. In each stage of the VDT (vm-based dynamic threshold) algorithm, the host with the minimum UT_{HI} value is selected, all the virtual machines within the selected host try to migrate to other hosts. The host goes into the sleep mode. This procedure is carried out for all the hosts which are not in the lightly loaded mode.

3. For selecting a new host for virtual machines, the UMC (utilization and minimum correlation) algorithm is proposed in this section. UMC is based on multiple correlation coefficient and host processor utilization(algorithm 2-5)[26]. UMC is implemented in two steps: In the first step, a number of hosts are selected as candidates based on processor utilization for finding a new host for the virtual machine. This algorithm assures that a lightly loaded host is not selected for a new destination (algorithm 3-5). In the second step, in the UMC algorithm, for selecting a new host for the virtual machine i, a host among the candidate hosts, will be selected where the virtual machines within it have the least dependency on the virtual machine I in terms of processor utilization history. Hence, in the new host, the virtual machine i has the least similarity to other virtual machines within the host with respect to resource utilization. Thus, it can be argued that UMC reduces SLA violation, the number of migrations and also the amount of transferred data. This algorithm reduces the number hosts which have reached the peak of their load. Consequently, the number of overloaded hosts is reduced(algorithm 4-5). Thus, this is regarded as another reason and justification for the reduction of migrations in the proposed method.

Algorithm (1-5) : Finding VDT(VM-based dynamic threshold)
Input:host list
Output:migration virtual machine (VM) list

```
1: while (Has host that not checked)
2:   minScore ← MAX
3:   underutilizedHost ← NULL
4:   foreach (host_i ∈ hostList) do
5:     hostScore ← calculateScoreOf(host_i) // UT_Hi = α.U_Hi + β.VMs_Hi
6:     if (hostScore < minScore) then
7:       minScore ← hostScore
8:       underutilizedHost ← host_i
9:     end if
10:  end for
11:  Add VMs of underutilizedHost to migrationList
12:  Add host_i to checked hosts
13: end while
14: return migrationList
```

Algorithm(2-5) :Power – Aware Best Fit Decreasing
Input: virtual machines List (vm list)and Host List (hostList). Output:virtual machinesallocation

```
1: Sort VMs based on utilization in decreasing order
2: foreach (vm) in (vmList) do
3:   minPower ← MAX
4:   allocatedHost ← NULL
5:   foreach (host) in (hostList) do
6:     if (host has enough resource for vm) then
7:       Power ← estimatePower(host, vm)
8:       if (power < minPower) then
9:         allocatedHost ← host
10:        minPower ← power
11:      end if
12:    end if
13:  end foreach
14:  if (allocatedHost != NULL) then
15:    allocate vm to allocatedHost
16:  end if
17: end foreach
18: return new VMs placement
```

Algorithm (3-5). Selection of a new location for virtual machines (UMC) Input: virtual machines which should be migrated and moved (migratable VMsList). Output: migration map.

```
1: foreach (VM_i ∈ migratableVMsList) do
2:   candidateHostList ← getCandidateHostList(VM_i) // Algorithm 4-4
3:   minCorrelation ← MAX
4:   destinationHost ← NULL
5:   if (candidateHostList == NULL) then
6:     destinationHost ← get new placement for VM_i by PABFD // Algoritm 4-2
7:     Set destinationHost as new placement for VM_i
8:     Continue
9:   end if
10:  foreach (host_i ∈ candidateHostList) do
11:    metric ← getCorrelation(vm, host.vmList())
12:    if (metric < minCorrelation) then
13:      minCorrelation ← metric
14:      destinationHost ← host
15:    end if
16:  end for
17:  Set destinationHost as new placement for VM_i
18: end for
19: return migrationMap
```

Algorithm (4-5) : Finding a candidate host for virtual machines Input: virtual machine (VM)
Output: candidate host list

```
1: hostList ← getHostList()
2: candidateHostList ← NULL
3: foreach (host_i ∈ hostList)
4:   utilization ← getUtilization(host)
5:   if (host_i.isSuitable(vm) == false or isSwitchOff(host_i) == true) then
6:     Continue
7:   end if
8:   if (utilization > utilizationThreshold) then
9:     Add host_i to candidateHostList
```

Cloud computing service providers face a trade-off between power and efficiency (improvement of equitable relation along with SLA reduction). For maximizing profit, service providers need energy-efficient methods. In such methods, virtual machines are combined and idle servers are put into the sleep mode so as to optimize equitable relations (Fig. 1). However, this combination increases SLA violation which is signed between service provide and user at the outset of the contract (Fig. 2).The results of the present study revealed that the proposed method is better than other dynamic VM algorithmsboth in terms of reducing SLA violation and reducing the amount of transmitted data (Fig.3) and reduce the number migration between virtual machines for sightly finding hosts (Fig. 4) that Performance is better than other algorithms.

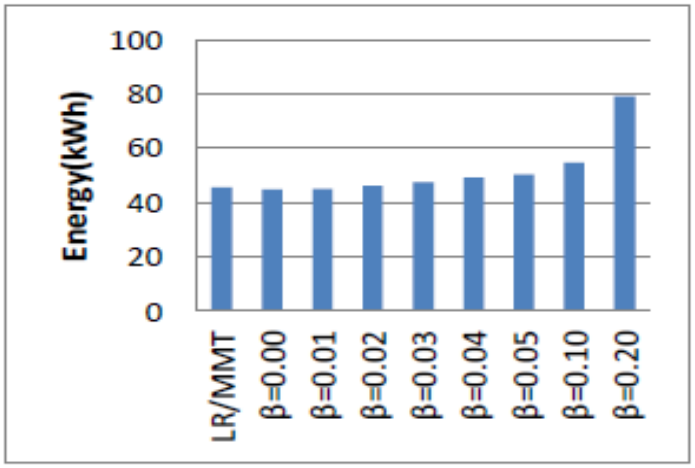

Figure.1. Energy consumption

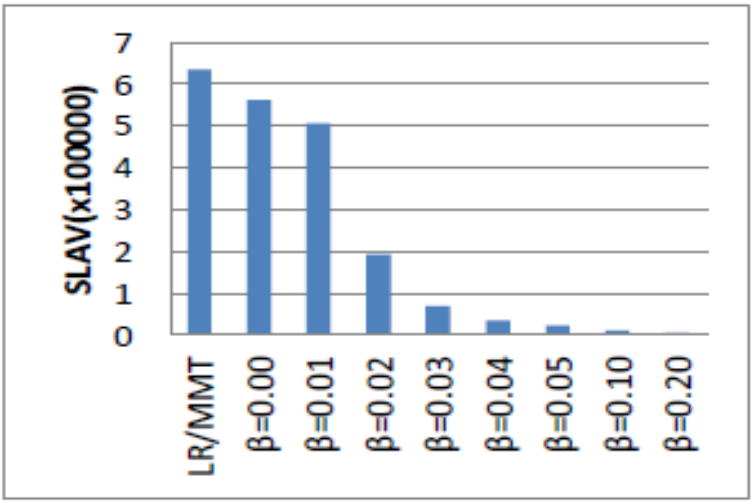

Figure.2. The violation of SLA

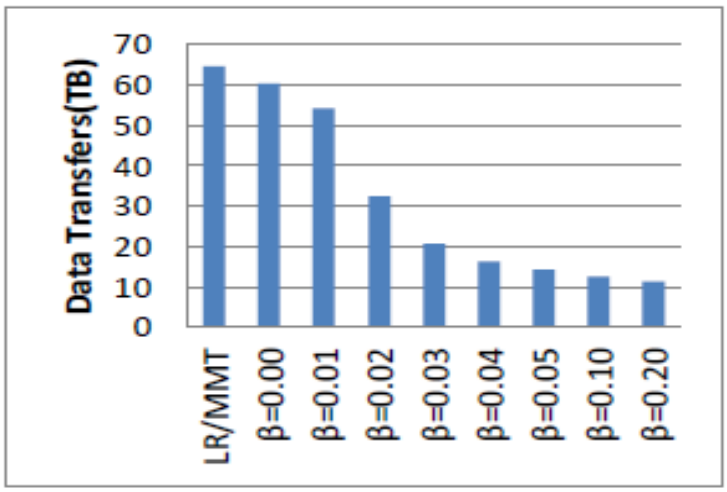

Figure.3. The amount of transmitted data

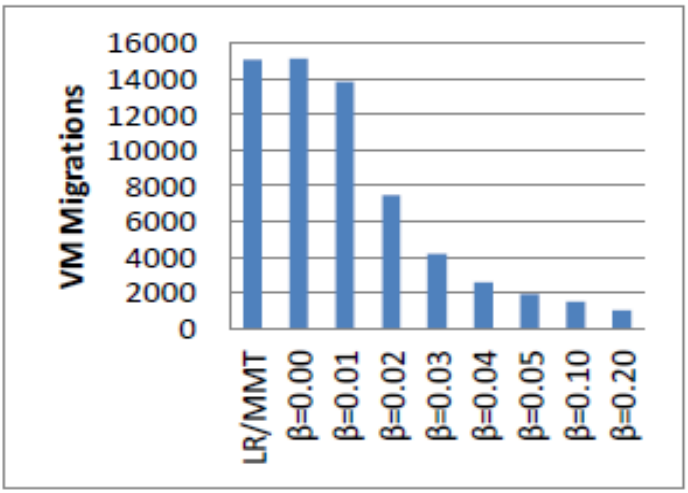

Figure.4. Migrated List

5. Conclusion:

In this paper, a novel method was proposed for finding lightly-loaded hosts and new hosts for virtual machines.For identifying lightly loaded hosts, the number of VMs was considered as well as processor utilization. So, in determining the new host, a host will be selected as the new destination for VM which has the least similarity with the VMs within the host. According to the algorithm presented in this article to find hosts lightly-load and find a new host for virtual machines simulation results using a large-scale data and center with thousands of virtual machine data is evaluated on Planet-Lab performance results is provided better than other algorithms.It is mentioned in this article in the simulation is not considered in the calculation of costs (energy consumed by the network due to immigration).The use of statistical methods and data gathered from the past virtual machines in various stages of virtual machinescombined can be very useful in my future work will be in the future.

References

[1] S. Abramson, W. Horka, and L. Wisniewski, "A Hybrid Cloud Architecture for a Social Science Research Computing Data Center," in Distributed Computing Systems Workshops (ICDCSW), 2014 IEEE 34th International Conference on, 2014, pp. 45–50.A. Author 1 and B. Author 2, "Title of the conference paper," *Proc. Int. Conf. on Power System Reliability*. Singapore, pp. 100-105, 1999.

[2] J. Adhikari and S. Patil, "Double threshold energy aware load balancing in cloud computing," in Computing, Communications and Networking Technologies (ICCCNT),2013 Fourth International Conference on, 2013, pp. 1–6.A. Author 1 and B. Author 2, "Title of the journal paper" *IEEE Trans. Antennas and Propagation*, Vol. 55, No. 1, pp. 12-23, 2007.

[3] M. Ajit and G. Vidya, "VM level load balancing in cloud environment," in Computing, Communications and Networking Technologies (ICCCNT),2013 Fourth International Conference on, 2013, pp. 1–5

[4] E. Al-Rayis and H. Kurdi, "Performance Analysis of Load Balancing Architectures in Cloud Computing," in Modelling Symposium (EMS), 2013 European, 2013, pp. 520–524

[5] J. Bhatia, T. Patel, H. Trivedi, and V. Majmudar, "HTV Dynamic Load Balancing Algorithm for Virtual Machine Instances in Cloud," in Cloud and Services Computing (ISCOS), 2012International Symposium on, 2012, pp. 15–20.

[6] H. Chen, F. Wang, N. Helian, and G. Akanmu, "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing," in *Parallel Computing Technologies (PARCOMPTECH), 2013 National Conference on*,2013, pp. 1–8.

[7] D. Goutam, A. Verma, and N. Agrawal, "The performance evaluation of proactive fault tolerant scheme over cloud using CloudSim simulator," in *Applications of Digital Information and Web Technologies (ICADIWT), 2014 Fifth International Conference on the*, 2014, pp. 171–176.

[8] J. Grover and S. Katiyar, "Agent based dynamic load balancing in Cloud Computing," in *Human Computer Interactions (ICHCI), 2013 International Conference on*, 2013, pp. 1–6.

[9] R. A. Haidri, C. P. Katti, and P. C. Saxena, "A load balancing strategy for Cloud Computing environment," in *Signal Propagation and Computer Technology (ICSPCT), 2014 International Conference on*, 2014, pp. 636–641.

[10] C.-H. Hsu, S.-C. Chen, C.-C. Lee, H.-Y. Chang, K.-C. Lai, K.-C. Li, and C. Rong, "Energy-Aware Task Consolidation Technique for Cloud Computing," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, 2011, pp. 115–121.

[11] C.-C. Lin, H.-H. Chin, and D.-J. Deng, "Dynamic Multiservice Load Balancing in Cloud-Based Multimedia System," *Syst. Journal, IEEE*, vol. 8, no. 1, pp. 225–234, Mar. 2014.

[12] K. A. Nuaimi, N. Mohamed, M. A. Nuaimi, and J. Al-Jaroodi, "A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms," in *Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on*, 2012, pp. 137–142.

[13] A. Rahman, X. Liu, and F. Kong, "A Survey on Geographic Load Balancing Based Data Center Power Management in the Smart Grid Environment," *Commun. Surv. Tutorials, IEEE*, vol. 16, no. 1, pp. 214–233, 2014

[14] R. A. M. Razali, R. A. Rahman, N. Zaini, and M. Samad, "Virtual machine migration implementation in load balancing for Cloud computing," in *Intelligent and Advanced Systems (ICIAS), 2014 5th International Conference on*, 2014, pp. 1–4.

[15] C. Zou, Y. Lu, F. Zhang, and S. Sun, "Load-based controlling scheme of virtual machine migration," in *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*, 2012, vol. 01, pp. 209–213.

[16] J.-L. Chen, Y. T. Larosa, and P.-J. Yang, "Optimal QoS load balancing mechanism for virtual machines scheduling in eucalyptus cloud computing platform," in *Future Internet Communications (BCFIC), 2012 2nd Baltic Congress on*, 2012, pp. 214–221.

[17] L. De-wen, H. Wen-jun, L. Long, and L. Long, "Design of real-time database for industry control system based on cloud theory," in Control and Automation (ICCA), 2013 10th IEEE International Conference on, 2013, pp. 363–367.

[18] G. Fen, M. Hua-Qing, and Y. Jie, "Performance Weighted Deploying and Scheduling Strategy Research for Virtual Machine on Clouds," in *Emerging Intelligent Data and Web Technologies (EIDWT), 2013 Fourth International Conference on*, 2013, pp. 56–60.

[19] Y. Gan, S. Han, G. Chen, and Z. He, "A research of object mapping algorithm based on cloud storage," in *Pervasive Computing and Applications (ICPCA), 2010 5th International Conference on*, 2010, pp. 228–231.

[20] M. Wang, X. Wu, W. Zhang, F. Ding, J. Zhou, and G. Pei, "A Conceptual Platform of SLA in Cloud Computing," in *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, 2011, pp. 1131–1135.

[21] A. M. Hammadi and O. Hussain, "A Framework for SLA Assurance in Cloud Computing," in *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*, 2012, pp. 393–398.

[22] A. J. Gonzalez and B. E. Helvik, "System management to comply with SLA availability guarantees in cloud computing," in Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on, 2012, pp. 325–332

[23] D. Liu, U. Kanabar, and C.-H. Lung, "A light weight SLA management infrastructure for cloud computing," in *Electrical and Computer Engineering (CCECE), 2013 26th Annual IEEE Canadian Conference on*, 2013, pp. 1–4.

[24] C. Redl, I. Breskovic, I. Brandic, and S. Dustdar, "Automatic SLA Matching and Provider Selection in Grid and Cloud Computing Markets," in *Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on*, 2012, pp. 85–94.

[25] J.-S. Liao, C.-C. Chang, Y.-L. Hsu, X.-W. Zhang, K.-C. Lai, and C.-H. Hsu, "Energy-Efficient Resource Provisioning with SLA Consideration on Cloud Computing," in *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*, 2012, pp. 206–211.

[26] A.-F. Antonescu and T. Braun, "Improving management of distributed services using correlations and predictions in SLA-driven cloud computing systems," in Network Operations and Management Symposium (NOMS), 2014 IEEE, 2014, pp. 1–8.

[27] L. Eyraud-Dubois and H. Larcheveque, "Optimizing Resource allocation while handling SLA violations in Cloud Computing platforms," in *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, 2013, pp. 79–87.