Original Article

A new technique to optimize dynamic load balancing problem with distributed algorithm

Alireza Beyramizadeh[1]*, Mohammadreza Mollahoseini Ardakani[2]

[1] phD Student, Faculty of Engineering, Islamic Azad University of Maybod Branch, Maybod, Iran. [2]Assistant professor, computer Engineering Department. Meybod Branch, Islamic Azad University, Meybod, Iran.

Correspondence: Alireza Beyramizadeh, phD Student, Faculty of Engineering, Islamic Azad University of Maybod Branch, Maybod, Iran. Email: alibeyrami@gmail.com

ABSTRACT

With the increasing growth of computing technology, the computers have become more efficient and compact. With the development of powerful microcomputers, the distributed computing has been introduced as an optimal solution to gain faster computing using existing capacities. One of the most important issues in distributed computing systems is allocating a set of tasks to a set of processors to create the load balance and to minimize the total costs. Due to the large scale and complexities of distributed systems, centralized allocation of tasks to specific servers is impossible. To properly manage the service providing resources, we need to acquire the load balance to be proposed to the service provider. Components are regularly monitored and when one component is not responding, the load balancer pops-up and prevents the traffic from being driven to that component. With proper in-situ load level analysis, problems of resource consumption can be usually be minimized, which not only leads to lower costs and green computation, but also lowers the pressure on particular circuits and potentially lengthens their service life. In fact, the aim of load balancing is to find a suitable mapping of the tasks on the processors in the system.

Keywords: Optimization and processing, load balancing, distributed

Introduction

The growth and development of applications and new customers have created new requirements such as data security, faster processing, dynamic data accessibility, and most importantly, cost saving.

To meet these requirements, organizations should make any effort to find new methods to increase memory capacity and meet the demands without the need for new hardware or hiring new staff [1].

Cloud computing is a relatively new service which is defined as using system resources in a completely transparent manner without engaging the user with the infrastructure details. It is one of the most recent developments in information technology and is becoming more prevalent faster over time. In terms of marketing, understanding the load balancing effects in the cloud is very significant. Cloud computing platform is a fully automated service platform allowing the users perform tasks such as purchasing, e-generation, dynamic scaling, and system management. Load balancing is currently a challenge in cloud computing systems.

Obviously, due to the large scale and complexity of these systems, centralized allocation of tasks to specific servers is impossible. To properly manage the service providing resources, we need to acquire the load balance to be proposed to the service provider. Components are regularly monitored and when one component is not responding, the load balancer pops-up and prevents the traffic from being driven to that component. With proper in-situ load level analysis, problems of resource consumption can be usually be minimized, which not only leads to lower costs and green computation, but also lowers the pressure on particular circuits and potentially lengthens their

Access this article online	
Website: www.japer.in	**E-ISSN:** 2249-3379

How to cite this article: Alireza Beyramizadeh, Mohammadreza Mollahoseini Ardakani. A new technique to optimize dynamic load balancing problem with distributed algorithm. J Adv Pharm Edu Res 2020(S1);149-156.
Source of Support: Nil, Conflict of Interest: None declared.

service life. In fact, the aim of load balancing is to find a suitable mapping of the tasks on the processors in the system, in such a way that the same amount of tasks are implemented on all processors, so that the overall runtime is minimized [1]. In distributed systems, where the number of requests and the scale of the workload are high, load balancing is a critical issue.

Methodology

Literature review

The traditional virtual machines scheduling in the cloud computing environment usually takes into account only the current state of the system and rarely pays attention to the previous states. This creates load imbalances in the system. The proposed Evolutionary algorithms such as the genetic algorithm and round-robin scheduling algorithm, used as the basic algorithm in this environment, have not been successful in addressing the load balancing problem.

One of the most important issues in distributed computing systems is allocating a set of tasks to a set of processors to create the load balance and to minimize the total costs. The task allocation problem in the distributed computing environments to achieve higher efficiency when using system resources as well as the high level of load balance, refers to allocating a computer application, including a set of interrelated and collaborating tasks, to a set of computers or processors in a distributed system, taking into account a set of constraints on resources (processors, communicating channels, and so on). The marginal goal of this allocation process is the optimization of the overall costs of the system, including operational and communication costs. For this purpose, an appropriate cost function is defined for the allocation problem in distributed computing environments, and the objective is to optimize this cost function taking into account the constraints of the resources in the system (processors, communicating channels).

In the present article, we are to investigate the different load balancing methods to reduce response time in cloud computing environments, which is one of the most important factors in service agreement and increasing customer satisfaction. Thereafter, the proposed algorithm is discussed. In fact, with improving the load balance, the efficiency of cloud computing, and thus the customer satisfaction of the service provided by the cloud network, could be ascertained. According to the studies, there is currently no algorithm that addresses all of aspects of on load balancing with acceptable efficiency. The proposed algorithm tries to enclose the parameters along with their required effects in a highly flexible equation. Taking all aspects intro account, this algorithm can make a significant improvement in cloud computing.

Load balance

This reallocation of the total load to each and every node of the collective system is performed for the efficient utilization of resources and for improving the response time of a task, and, simultaneously, for getting rid of a condition where some of the machines are overloaded and some are under-loaded.

A naturally dynamic occurring load balancing algorithm cannot take into account the previous state or behavior of the system. So, this algorithm depends on the current state of the system. Significant issues to be considered include load estimation, load ratio, stability of various systems, system performance, inter-node interactions, nature of the communicated task, node selection, and many others [2].

This load could include the processor load, amount of storage memory used, delay, or network load. When a given workload is allocated to each machine in a cluster, if the available resources are utilized in an efficient manner, this load could run also in an efficient way. So there should be a mechanism to select the machines that have these resources available. Scheduling is a component or mechanism responsible for selecting a cluster machine [3, 4]. In such cases, scheduling requires algorithms to solve these problems.

In the real-world conditions, load balancing is mainly influenced by three factors [5]:

1. The environment in which load balancing is one of the requirements
2. The nature of the load itself
3. Tools available to balance the load

Goals of load balancing

- Significant improvement of efficiency
- Maintaining a backup plan even when a system component crashes
- Maintaining the stability of the system
- Preparation for the future changes in the system.

Various types of load balancing algorithms

Depending on who initiates the load balancing process, there are three classes of load balancing algorithms [6]:

- Sender initiated

When load balancing algorithms are initiated by the sender

- Receiver initiated

When load balancing algorithms are initiated by the receiver

- Symmetric

A combination of the above states

According to the current state of the system, load balancing algorithms can be classified into two categories [6]:

Static: It does not depend on the current state of the system and requires previous system data.

Dynamic: Decision for load balancing is made based on the current state of the system and no previous data is required. So, this is a more highly preferred technique than the static method.

- **Dynamic load balancing algorithms**

In dynamic load balancing algorithms, the workload is distributed between processors at runtime. The Master processor allocates the new process to the Slave processor on the basis of recently collected data [7].

In a distributed system, dynamic load balancing can be performed by two methods: distributed and undistributed.

In the distributed mode, the load balancing algorithms are deployed by all the machines throughout the system, and the load balancing responsibility is shared among the machines. Machine communication for achieving the load balance can be performed in cooperative and non-cooperative forms.

Parameters of cloud load balancing

Cloud load balancing techniques take into account various parameters such as efficiency, response time, scalability, throughput, resource utilization, fault tolerance, migration time and associated overhead. However, for a load balance with energy efficiency, the parameters should also include energy consumption and carbon emissions.

Relevant overhead: When a load balancing algorithm is implemented, it would determine the rate of overhead, which involves overheads settled for task improvement, process communication, and processing. This overhead rate should be minimized, such that the load balancing technique works effectively.

Throughput: It is used to calculate the tasks that have been completed.

Efficiency: It is used to control system efficiency and should improve reasonable costs; e.x. it should reduce response time of the tasks and keep the delays at an acceptable level.

Resource utilization: It is used to control the resource utilization, and should be optimized for efficient load balancing.

Resource utilization involves the automated load balancing of a distributed system that may have an unexpected number of processes which demand higher processing power. In case the algorithm is able to use resources, they may be switched to lower-loaded processors to be more efficient.

Static load balancing algorithms utilize the resources less frequently, in such as way that the load balancing methods try to allocate tasks to the processors to achieve the lowest response time, and ignore the fact that this task allocation may lead to a condition where a number of processors finish their tasks immediately and remain idle due to lack of tasks.

Dynamic load balancing algorithms utilize the resources in a much more optimized way, such that they follow the principle that the load has to be distributed evenly among the processors, in a way that no processor remains idle.

Scalability: It is the load balancing capabilities of an algorithm in a system with a finite number of nodes. This parameter needs to be improved.

Response time: The response time of a specific load balancing algorithm in a distributed system. This parameter should be minimized.

Fault tolerance: The ability of an algorithm to perform load balancing uniformly in connection failure cases. Load balancing should be a good fault tolerance technique.

Migration time: Time to migrate tasks or resources from one machine to another. To increase system performance, this time should be minimized.

Carbon emission: It calculates all resources in the system. Energy consumption goes hand in hand with carbon emission, so, higher energy consumption leads to higher carbon footprint. Therefore, to obtain a solution with optimal load balancing energy, this value should be reduced.

Proposed algorithm

Since the load balancing is one of the major challenges in cloud computing, so it requires a dynamic local workload distribution among all machines in an evenly manner in order to attain user satisfaction and high resource utilization rates and to ensure a fair and efficient allocation of computing resources. So, we compared various load balancing algorithms in cloud computing, and found that we could use a particular algorithm tailored to our needs. As everyone knows, cloud computing involves a wide range of areas. This is applicable to both large-scale and small domains, but none of the existing algorithms meet the required criteria. Therefore, an adaptive algorithm needs to be developed in accordance with heterogeneous environments that could also reduce the costs.

For this purpose, taking into account factors that affect a cloud computing system, we design an algorithm that reduces the costs and system errors and also increases the satisfaction rate of the users through load balancing.

When a load is to be transmitted on a machine in the cloud network, some major parameters should be taken into account. In case the only objective is to appropriately distribute the original load on the cloud, this should be taken as a special case of load distribution on the cloud. In general, the algorithm can be used to distribute any load on the network, particularly, the migration of applications, codes and even the operating system. The idea behind the algorithm is very straightforward. It considers what parameter is the most important and allocates it a higher decision coefficient. Cloud computing machines can be simply informed of the conditions of some parameters on other machines. By a simple signaling, (e.x.) the delay may be calculated. This algorithm requires only a small amount of data on the neighboring machines. This data is of slight volume compared to the mass of data exchanged in this environment.

In the proposed algorithm, we mostly emphasize the data that are locally available on the neighboring machines. There is no need for a general picture of the entire cloud network. Cloud network is a huge network and in case we are to gain complete information about the network, we require lots of memory and a large amount of data to be sent and received. Therefore, this algorithm uses local machine data, especially neighboring machines, and each machine sends codes or data to other machines according to their conditions.

If a machine gets into trouble, neighboring machine will soon find out and refer to it in their decision making. So none of the machines require lots of scrutiny to make the decision and simply find what it calls for.

In the proposed algorithm, we use the following equations and, using factors effective in load balancing and appropriate standard coefficients we would achieve the desired results. In cloud computing, one of the factors that affect load balancing is the number of neighboring machines. According to the cloud model, we know that the larger the number of neighbors, the better the load balance in the system. So, in the following equation, we add the coefficient *a* to the numerator of the equation. One of the other factors in cloud computing is the energy of each machine. The higher the energy, the better we may use that machine for load balancing. So, this factor took the coefficient *b* in the numerator of the equation. This factor was summed up with the above factor, i.e. the number of neighbors. Another factor to be used is the original load of each machine, which should be as lower as possible for load balancing. For this reason, in the following equation, we put this factor in the denominator with the coefficient *c*.

Another factor affecting the load balance is the time of response of each machine to the request. We can calculate the elapsed time through sending data to the respective machine and receiving the corresponding data. This could be done for all machines in the cloud system and their neighboring machines. The machine with the minimum time could be used for load balancing. Therefore, we put this factor in the denominator with the coefficient *d* to help us achieve a better load balance in the cloud computing.

The proposed equation is as follows:

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

$$a+b+c+d=1 \qquad (2)$$

where, nn is the number of neighbors, en in the energy level, ld is the load of each machine and rt is the response time of each machine in the network.

Also, a, b, c and d are the impact factors of each parameter, which is a number above than zero and below 1, and, as obvious, their total sum must equal 1.

$x1$, $x2$ and $x3$ are parameters to be used to balance the ratio between the parameters in the equation. Here, the number of neighbors is taken as the base parameter and other parameters are compared to it. That is, these parameters are used to assimilate various parameters in the equation, and their values are determined according to the expected values of the parameters.

This algorithm has high flexibility and more criteria could also be added to it. In different environments and under various conditions, expectations from the cloud computing may vary. Therefore, these parameters can be applied with slight changes or new parameters could be added. So, the impact factors of various parameters should firstly be determined. However, the parameters should be approximated to determine the exact value of the assimilation parameters.

The proposed equation is as follows:

$$\frac{nn*a+en*b*x1}{ld*c*x2+rt*d*x3} \qquad (3)$$

where, nn is the number of neighbors of each server, en in the energy level of the server, ld is the load of each server and rt is the response time of each server. a, b, c and d are the impact factors of the parameters and $x1$, $x2$ and $x3$ are the load balancing parameters.

For a numerical example, let the impact factors of the parameters are as follows:

$$a = 0.15 \;\; b = 0.30 \;\; c = 0.35 \;\; d = 0.20$$

Also to consider the balancing parameters, the range of parameters should be determined in the equation. For example, by default, a cloud server has about 10 to 50 neighboring servers, or the daily amount of energy it may consume in the cloud environment is 20 to 100 joules; also, the server load can be based, for example, on the number of virtual machines running on the server or the occupied memory level of these virtual machines on the server. Each of these factors gives different values. In addition, the multiplicative combination of these two concepts can be used. If we include the number of virtual machines in the equation, it would range from 1 to 1000 (for large servers). Although the number of virtual machines could also be 0, at least 1 virtual machine per server is assumed to take into account the minimum impact.

The servers are assumed to be similar in terms of memory size and capacity to accept virtual machines; otherwise, this difference should also be included. The estimated response time of each server can be, for example, between $1ms$ to $10s$. These are hypotheses of the problem and there may be differences in the real world.

At first the parameters are rewritten with small conversions:

$$nn = 10 - 50$$
$$en = 20 - 100\,J$$
$$ld = 1 - 1000\,vm$$
$$rt = 1 - 10000\,ms$$

Now, the suitable values for load balancing parameters could be as follows:

$$x1 = 2$$
$$x2 = 1/10 - 20$$
$$x3 = 1/10 - 200$$

if we assume that the values could be uniformly distributed in the desired intervals, the average of each interval can replace that load balancing parameter: therefore $x2$ takes the value 10 and $x3$ takes the value 100.

Now the equation is as follows:

$$\frac{nn*0.15+en*0.30*2}{ld*0.35*10+rt*0.2*100} \qquad (4)$$

Now, we can analyze which two servers are more suitable for migration of a virtual machine. (For example, for one server, nn, en, ld and rt are assumed to be 20, 25, 900, and 2200, respectively, and for the other, they are 35, 15, 30 and 350, respectively).

The proposed method actually works when a decision is made to make a virtual machine migrate to a physical machine. So, it is somehow different with the mechanisms discussed above. However, most previous methods discuss the virtual machine and their requests. For example, the Active Monitoring Load Balancer or Throttled Load Balancer techniques take into account a mechanism to decide which requests should be assigned to which virtual machines. These methods can also be used along with the proposed method to further improve load balancing in cloud environments.

To better understand the proposed algorithm, it I defined in some general steps:

First, it is assumed we have several physical machines that can vary in terms of hardware resources. There are no virtual machines in the system and no user processes are running on the systems.

Then, the first request arrives and a virtual machine should be created on a physical machine or server. This virtual machine is created taking into account the power and resources of the physical machine as well as the processes it needs to implement and the SLA required by the process. That which physical machine takes the responsibility for the process is determined based on the above equations. This could be done in a different way, because usually there is little data on the cloud environment in a distributed manner in the physical machines and there is a need for a coordinator in which physical machine data are based.

In terms of the subsequent requests, there will be three modes: They may be running on a pre-existing virtual machine; a new virtual machine may be created on an idle server; or a new virtual machine may be created on an occupied server. The third case usually doesn't happen, but in order to balance the load in the cloud environment, if the physical machine is still idling, a new virtual machine is better to be created on it to implement the process. In this decision, the SLA of the requests should also be taken into account. At this stage, the previously mentioned methods such as Active Monitoring Load Balancer can be used.

If a physical machine is in trouble or if it is forced to make the processes migrate for any reason, such as the high load and low resources required running the processes, the physical machine should specify the destination for each VM. At this point, the proposed equation is used and the destination physical machine is selected for each VM. The proposed design will be running until there is any process on the network for processing.

Discussion and Conclusion

Load balancing on the network leads to a loss of quality and, finally, the loss of user requests. When the load balancing is invoked, some phases may be observed. Depending on the different parts of the network, a special case may be related to the virtual machine migration on the server to gain the data.

In the present paper, the system in question is an IaaS environment represented as a data center containing N heterogeneous physical nodes. Each node I contains three features: processor use known as MIPS; memory use and bandwidth. Servers do not include local disks and the memory is provided through a NAS to allow live migration. Several independent users make their requests for using M heterogeneous virtual machines that have three characteristics: processing power (MIPS), memory capacity, and network bandwidth. Every user signs an SLA contract with the service provider; and in case of violation of SLA, the service provider is required to pay a penalty.

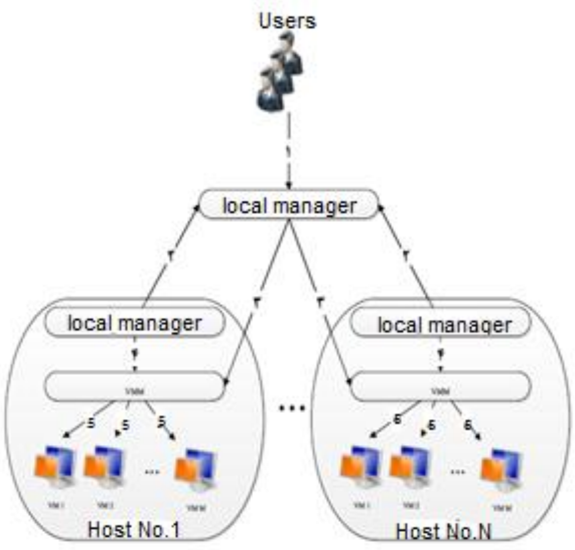

Figure 1. The model system used in this algorithm (users-general manager, local manager, local manager, host No.1, host No.N)

The software layer is divided into two sections of local manager and general manager. The general manager is located on the VM in each host. The tasks of these managers include constant monitoring the efficiency of the host CPU, resizing the virtual machines according to the requirements, and deciding on which virtual machines to migrate. The general manager is located on the main node and collects the required information from all local managers. The general manager orders the required instructions to optimize the location of virtual machines. VM is assigned to processes of resizing, migrating virtual machines and power variation.

Table 1. Specifications of the servers used

Server	CPU Model	Cores	Frequency (MHz)	RAM (GB)
HP ProLiant G4	Intel Xeon 3040	2	1860	4

HP ProLiant G5	Intel Xeon 3075	2	2660	4
IBM Server x3250	Intel Xeon 3470	4	2933	8
IBM Server x3550	Intel Xeon 5675×2	2*6	3067	16

Table 2. Power consumption of servers in different modes

Server	Idle	%10	%20	%30	%40	%50	%60	%70	%80	%90	Full
HP ProLiant G4	86	4.89	6.92	96	5.99	102	106	108	112	114	117
HP ProLiant G5	7.93	97	101	105	110	116	121	125	129	133	135
IBM Server x3250	6.41	7.46	3.52	9.57	4.65	73	7.80	5.89	6.99	105	113
IBM Server x3550	4.58	98	109	118	128	140	153	170	189	205	222

Here we make use of four servers: HP Proliant ML 110 G4, HP Proliant ML 110G5, IBM SERVER and IBM SERVER x3550. The configuration and specifications of these servers are provided in Table 1 and their power status in Table 2.

Calculation of costs of live migration and volume of migrated data

Live migration allows virtual machines to be migrated from one host to another without interruption and with the shortest downtime. Anyway, live migration has a negative effect on the performance of running programs. Scientiscts have made an empirical study on this effect and provided a model. They concluded that performance degradation and downtime depend on the behavior of the application (like: how many memory pages are updates by the program at runtime). Anyway, for the applications with variable workloads, such as web applications, average performance degradation can be estimated as about 10% of the processor efficiency.

Live migration increases the violation of the SLA. Therefore, the number of migrations needs to be as low as possible. The duration of a live migration depends on the amount of memory occupied by the virtual machine and the available bandwidth. Eq. 5 is used to estimate the duration of migration and performance degradation of the virtual machine j:

$$T_{m_j} = \frac{M_j}{B_j} \qquad U_{d_j} = 0.1\int_{t_0}^{t_0+T_{m_j}} u_j(t)\,dt \tag{5}$$

Where, U_{d_j} denotes the performance degradation due to virtual machine j. t_0 indicates the migration start time and T_{m_j} the is the time of migration. $u_j(t)$ shows the processor productivity, M_j is the amount of memory used by the virtual machine J, and B_j is the available bandwidth.

Eq. 6 is used to calculate the total amount of data migrated:

$$D_m = \sum_{i=1}^{I} M_i \tag{6}$$

Where, D_m is the total amount of data transmitted during migration; I is the total number of migrations and M_i is the amount of memory migrated.

SLA violation metrics

Quality of service is of high importance in cloud computing environments.

Quality of service is often characterized in the framework of SLA, which is determined by criteria such as minimum efficiency or maximum response time of the system. Since these properties vary depending on the type of application, it is necessary to introduce a metric that is independent of the workload. So, in the present paper, the following method is used to estimate the SLA violation. Two metrics have been proposed for this purpose:

The time during which the active host has experienced a 100% efficiency of the processor.

$$SLATAH = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{s_i}}{T_{a_i}} \tag{7}$$

Where, N is the number of hosts, T_{s_i} is the time during which the ith host has experienced a 100% efficiency of the processor, and T_{a_i} is the time during which the ith host has been active.

It has been shown that when the host reaches 100% efficiency, the performance of the applications is limited to the host capacity. Therefore, the desired level of performance may not be provided for the virtual machines.

Total performance degradation due to the migrations (PDM)

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{d_j}}{C_{r_j}} \tag{8}$$

Where, C_{d_j} is the performance degradation due to the migrations of jth virtual machine, C_{r_j} is the total capacity of the processor that the virtual machine has requested during its lifetime, and M is the number of virtual machines.
Eq. 9 shows the SLA violation rate

$$SLAV = SLATAH.PDM \qquad (9)$$

Conclusion

Cloud service providers deal with a trade-off between power and efficiency (improving fair communication vs. $SLAV$ reduction). To maximize the profit, service providers require energy efficient methods to consolidate the virtual machines and putting the idle servers to the sleep mode in order to improve the fair communication. Anyway, this consolidation may increase the violation rate of SLA, which has been signed between the service provider and the user at the beginning.
In the present article, a new method was provided to find new under-loaded hosts for the virtual machines. The proposed algorithms were evaluated by simulating a large-scale data center, using with data from thousands of PlanetLab virtual machines.

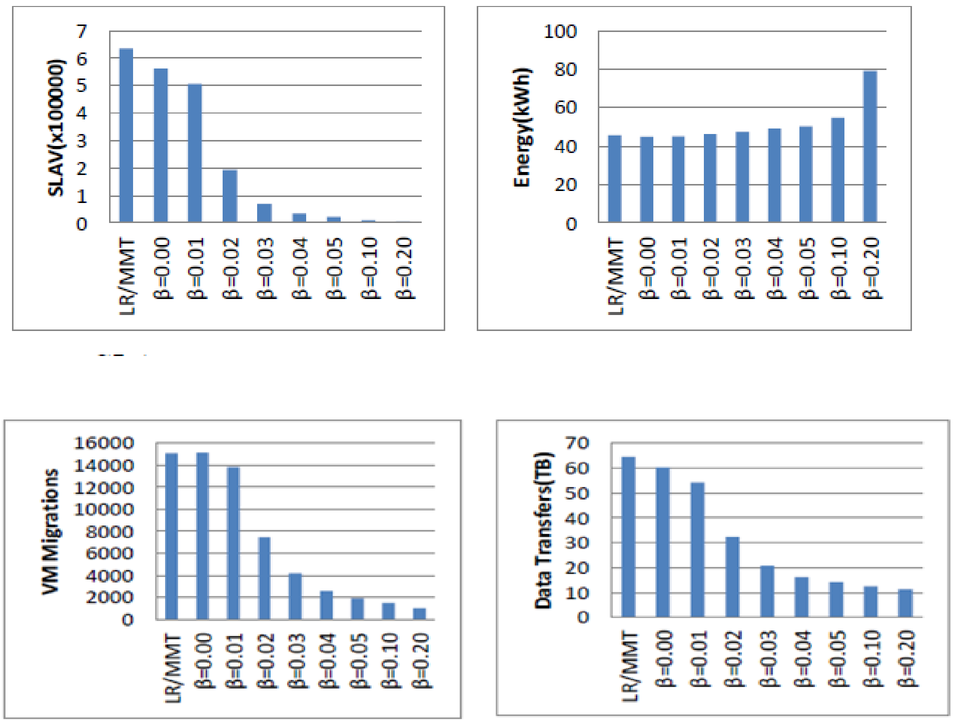

Figure 2. Simulation diagrams

The results show that the proposed method is better than other dynamic virtual machines consolidation algorithms in terms of average SLA violation as well as reduced amount of data migrated.
The results of the simulation are significant in two respects:
The number of virtual machines present in a host is an important factor to determine whether or not the host in under-loaded. That’s because, it is highly likely that the productivity of the host increases with a higher number of virtual machines due to its increased efficiency.
The behavioral history of the virtual machines is an important factor. Using this history will improve the fair communication and SLA violation.
The energy consumption is reduced because in this method, the peak energy consumption of the virtual machines will not occur simultaneously.
SLA violation is reduced because the number of conditions where hosts reach their peaks (100% efficiency) will be reduced.

References

1. Hamo A, Saeed A. Towards a Reference Model for Surveying a Load Balancing. IJCSNS International Journal of Computer Science and Network Security. 2013 Feb;13(2):42-7.
2. Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and experience. 2011 Jan;41(1):23-50.
3. Raghava NS, Singh D. Comparative study on load balancing techniques in cloud computing. Open journal of mobile computing and cloud computing. 2014 Aug;1(1).
4. Sethi S, Sahu A, Jena SK. Efficient load balancing in cloud computing using fuzzy logic. IOSR Journal of Engineering. 2012 Jul;2(7):65-71.

5. Walker BJ, Steel D. Implementing a Full Single System Image UnixWare Cluster: Middleware vs Underware. InPDPTA 1999 Jun (pp. 2767-2773).
6. Alakeel AM. A guide to dynamic load balancing in distributed computer systems. International Journal of Computer Science and Information Security. 2010 Jun;10(6):153-60.
7. Kokilavani T, Amalarethinam DG. Load balanced min-min algorithm for static meta-task scheduling in grid computing. International Journal of Computer Applications. 2011 Apr;20(2):43-9.