IJCSN - International Journal of Computer Science and Network, Volume 6, Issue 6, December 2017
ISSN (Online) : 2277-5420
www.IJCSN.org
Impact Factor: 1.5

# Global Server Load Balancing with Networked Load Balancers for Geographically Distributed Cloud Data-Centres

**[1] Amit Gajbhiye; [2] Dr. Shailendra Singh**

[1] Department of Computer Engineering and Applications
NITTTR, Bhopal, India

[2] Department of Computer Engineering and Applications
NITTTR, Bhopal, India

**Abstract -** The characteristic of high scalability, elasticity and availability of infinite computing resources have obliged many organizations to deploy mission critical and user centric applications in the cloud computing environment. To avoid any single point of failure in the cloud system the cloud service provider provide redundancy at all levels in the datacenter but the recent natural disasters have made the datacenter itself a single point of failure. Thus, to deploy applications in the datacenters dispersed across geographic regions has become the need of the hour. Efficient load balancing among these datacenters is critical to increase user responsiveness and seamless failover in case of natural disasters. In this research paper we critically analysed the state-of-the art techniques used for Global Server Load Balancing and have proposed a novel model of networked load balancers for load balancing across the datacenters in cloud computing environment. The proposed model overcomes the shortcomings of existing DNS based load balancing by considering current load status of datacenter, by making time to live of DNS server cache redundant and by finding the exact location by real IP of end user.

***Keywords -*** *Global Server Load Balancing, Domain Name System, Load Balancing, Disaster Recovery, Cloud Computing*

## 1. Introduction

Cloud computing (or cloud for short) is a disruptive technology. It is a model for delivering the hardware and software services on demand in a scalable manner. The compelling need which has given birth to this technology is the demand of instant infrastructure provisioning with no or less upfront cost and rapid scalability of the applications. The main thrust for this technology has been virtualization, parallel programming model with distributed file system and multiple speed gigabit networks. The datacenter hardware including various servers like storage, application and web and the software services deployed on them is called cloud [1]. The efficient utilization of datacenter resources and customer satisfaction for the responsiveness of their application or service is a crucial factor for the success of cloud computing. This fact strive us to think how to evenly distribute service requests(load) from the end users to the servers in order to maximize user responsiveness and minimise exploitation of a single server being over loaded with service request other being under loaded. In cloud computing parlance this is called load balancing. Load balancing [2] in cloud computing is essential for the following reasons as it:

- Increases resource utilization.
- Improves resource availability.
- Improves the responsiveness of services to users.
- Tracking and controlling traffic.
- Prevents over provisioning of resources.
- Avoid overloading and under loading of resources.

Load balancing can be done at the datacenter (in the datacenter) level or at the global level (across the datacenter). As mentioned at the datacenter level the load balancers are concerned with distributing dynamic load among the various severs evenly. On the other hand, the global server load balancing (GSLB) is concerned with distributing the load geographically across multiple datacenters. In the literature of cloud computing specially for load balancing much emphasis is given on the load balancing strategies within the datacenter. However, the load balancing among the datacenters dispersed across the geography is also crucial for two important reasons.

First, looking at the present demanding business needs for example in e-commerce and banking where critical business processes are deployed and run in the cloud computing environment, the organizations cannot afford the service outages even for short period of time. This scenario has increased the significance of factors like continuous availability and fast responsiveness of services many folds. The organizations are in a great need of efficiently utilizing their resources today than ever before.

Second the recent spate of natural disaster, power failures or DNS DDOS attack and many other unforeseen circumstances datacenters itself has become a single point of failure. Global server load balancing is used when an organization wants multi site load balancing and disaster recovery. They work at the datacenter level to provide resiliency to cloud hosted application.

In this research paper we review the state-of-the-art techniques for global server load balancing and propose a novel framework for global sever load balancing by networking load balancers across geographically distributes load balancers. Our theoretical analysis suggests that the proposed model not only overcome the shortcomings of existing DNS based load balancing by considering current load status of the datacenter, by making time to live of DNS server cache redundant and by finding the exact location by real IP of end user but also provide seamless failover and maximum uptime in case of natural disasters or a complete datacenter failure.

The primary contributions of this paper are:

- A distributed and dynamic load balancing model for global server load balancing with networking among load balancers in datacenters distributed across geography.
- A novel load calculation strategy to calculate the load of entire datacenters based on the summation of load of virtual machines on individual server in datacenter.
- A request migration strategy from one datacenter to another to reduce the response time for requests.

## 2. Related Work

In this section we will review the existing research literature in the field of global server load balancing. Most of the academic research work in the field of load balancing in cloud computing is concentrated on balancing the loads on the servers of the same datacenters. But there is some pioneering research in industry for global server load balancing. Prominent among the organization involved in GSLB research are Akamai, F5 Networks and Citrix.

The authors in [3] have proposed a site selection method for selecting the best server among the set of application servers located across geographically distributed datacenters. The proposed DR3 method combines the DNS reply race with DNS reflection for selecting a site with faster response time and lower cost. The DR3 method selects site with higher accuracy and lower cost as compared to DNS reply race and ping method of site selection. The research work in[4] discusses the various technical and business goals of GSLB. The author also discusses the crucial challenges and their solution in this paper.

The [5] have proposed a server load balancing protocol for efficient server selection for the client. In this protocol the server registers itself and coordinates its services with a Server Load Balancer (SLB) manager. For registration to SLB manager the server uses the Transmission Control Protocol. The important jobs of SLB manager in this protocol is to register virtual IP address of server with a DNS based global server load balancer, support VIP address route injection and provide health status of services to applications which helps in selection of optimal site. A DNS based HTTP redirection and route health injection method for global server load balancing is outlined in [6].

The authors of [7], proposed a model in which the servers handles the task of balancing the loads by themselves. When the server receives a request it either processes the request or passes the request to its peer. The decision to process the request depends on the current load of the system and/or on the request contents. As there in no dedicated hardware resources for load balancing the disadvantages associated with them such as single point of failure are not found in this model of global server load balancing.

The peers communicate with each other by passing messages either by following request/response semantics or by circulating unsolicited multicast messages about their current load status. The messages may be passed periodically or in response to some events. In [8] the authors have proposed a global server load balancing switch which acts as a proxy to authoritative name servers. The GSLB switch gathers load information of the servers from switches attached to servers directly and returns the IP address of the least loaded server to the hosts.

In [9] the authors have presented a system and a method for load balancing multiple geographically separated servers. The system consists of load balancing domain name servers (DNS-LBS) which are located close to the servers of Internet Service Provider and hence are better able to judge the performance of servers. The DNS-LBS returns the IP address of the server that provides better performance according to the physical location of the server.

The paper discusses two GSLB methods called Round Trip Time (RTT) and Static Proximity. RTT method is dynamic probing method in which the round trip time is used to find the IP address of closest site and the IP address is returned to the user. In contrast to RTT method static proximity method uses pre-configured IP address tables to find the closest site.

## 3.Analysis of State-of-the-art GSLB Techniques

In this section, we critically review the state-of-the-art techniques in the field of global server load balancing and discusses their suitability in the field of cloud computing.

### 3.1 DNS based Global Server Load Balancing

The most common method of global load balancing followed nowadays is with the help of Domain Name System (DNS) [10]. DNS works in client/server architecture to map host names in the application layer to an IP address in the network layer of TCP/IP protocol suit. Whenever a client requests for a service over the internet the request is first sent to the DNS for translating domain names to IP addresses. The DNS client sends a GetHostByName query to request the nearest DNS server to map a name to an IP address. The local DNS server searches its cache to check if it already contains an entry for the host name if it finds an entry it replies with the corresponding IP address and if it does not have an entry it refers to the other DNS server to provide the information. DNS can be configured to respond with different IPs based on the location of the client or in a "round-robin" manner. Thus the DNS based Global Server Load Balancing effectively divides load among the servers across the geography by responding with a different IP address of the server for the client request. Fig 1. shows the DNS based GSLB mechanism.

The disadvantages of DNS based GSLB Model method is as follows [11]:

- It does not consider the current load on the server but simply directs the requests to the servers in a round robin fashion which may create load imbalance in the system.

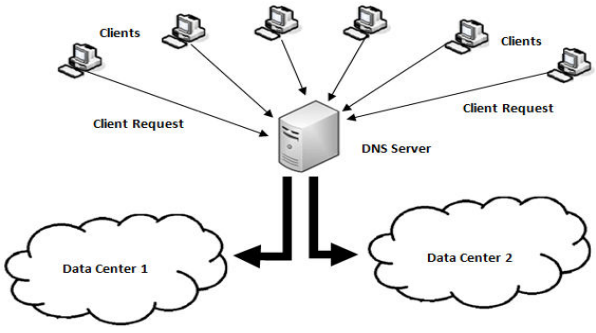


Fig. 1 GSLB Model with DNS

- To increase the efficiency by reducing the time spent in searching the DNS database, DNS servers uses the cache to store the most frequent name-address resolutions. The entries are cached in cache for a time (called TTL short for time to live) specified by authoritative servers. The TTL signifies the validity of cache entry. For a period specified by TTL the DNS server will send the cached address of services delivering server without querying the authoritative servers for the updated address if in case the old IP address is changed. For typical DNS implementation the value of TTL is 86400 seconds, which is 24 hours. In case of disaster or complete failure of datacenter the DNS will keep sending the client request for a period of 24 hrs to the defunct datacenter which is highly undesirable in dynamic cloud environment.
- The DNS distributes the load based on geographic location of the client. DNS follow either recursive or iterative resolution mechanism in case the resolution cannot be done at local DNS. Due to this resolution mechanisms the DNS may not know the correct geographic location of the end client and may direct the end clients to avail a service from distant server, which is not only time consuming but also incurs cost in cloud environment.

### 3.2 A Variation of the DNS based Global Server Load Balancing

To overcome the disadvantages of not considering current load of servers a new model is employed by many organizations like Yahoo by deploying dedicated global server load balancer. Fig. 2 shows the steps followed by this model to resolve a Fully Qualified Domain Name. Datacenter (Site) A for example, has a virtual IP address (VIP) of 11.11.11.11 and Datacenter (Site) B has a VIP of 22.22.22.22. In this model a Global Server Load Balancer acts as an authoritative name server which responds to the client request for appropriate server for establishing connection for service or communication. Figure 2, explains the whole communication steps for an example site www.ExampleModel2.com.

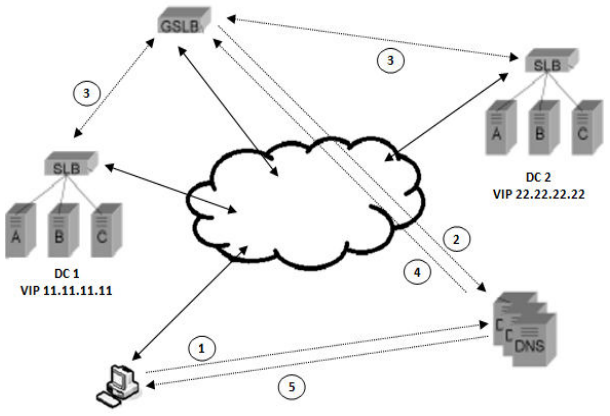


Fig. 2 GSLB model with dedicated servers

a) The DNS client (resolver) requests the local DNS server, which in our example is the name server of Internet Service Provider (ISP).

b) The client's DNS server queries root name servers iteratively and finally ends up at the authoritative name server for www.ExampleModel2.com. In this model GSLB is the authoritative name server.

c) The GSLB is connected to the load balancer of each of the datacenter and it has a record for the current load of the datacenter i.e. the health of datacenter.

d) Based on the load at each datacenter the GSLB now makes a decision as to which datacenter to connect to and returns the answer to client's DNS server. The value is stored at the client's DNS server for the TTL time period as specified by GSLB.

e) The choice of datacenter is returned to the client resolver, which now makes a TCP connection to the preferred datacenter.

This model works exactly like normal DNS but it applies additional intelligence to decide the preferred datacenter to connect to as opposed to normal DNS. However this model also has the following shortcomings.

- First, although this model have the advantage of considering load at the time of deciding preferred datacenter it still is infected by the delays caused by high TTL and even if the authoritative server (GSLB) reduces the TTL the resolving name servers (ISP DNS server) may override the TTL value. Therefore, even if TTL specified by GSLB is 60 seconds the resolving name servers may cache the response for 3600 second to avoid overloading its resolving name server [12].
- Second, most organization [13] [14] claims to provide load balancing based on end client location but they considers the IP address of resolving name server for calculating the proximity of request serving datacenter and end client, which may not be true for all the cases because the end client may use any DNS Server for resolving names to IP address and even if the client uses the DNS server of ISP, the ISP may host its DNS services far from end client.

Motivated by the need to provide an effective and efficient global server load balancing and to fill the gap of current practices, we propose novel global server load balancing model for centralised load balancer in the datacenter. In our dynamic load balancing model load balancers across the geography communicate and coordinate with each other to provide high responsive and cost effective delivery to end users and provide seamless failover in case of complete datacenter failure. We balance the load at two levels first on the basis of current load status and second on the basis of real IP address of end user. In essence we are distributing load among central load balancers by decentralising the authority load balancing.

## 4. Proposed Approach

### 4.1 Networked Load Balancers System: Architecture, Design and Working

In this section, we present the system model of the networked load balancers for distributed load balancing. This model focuses on organizations that have many datacenter distributed across the globe with their respective centralised load balancer. We propose the idea of connecting the load balancers of these organizations datacenter so that they can communicate and coordinate with each other. The load balancers in the datacenters can communicate with each other to share their current load status and coordinate accordingly distribute service request (load) among them. Fig. 3 shows the proposed model.

In the proposed model the centralised load balancers of the datacenters maintain a load table depicted in Fig. 4(a). The load table contains the current load information against the IP address of datacenter and their health status which essentially is the further load accepting capacity of the datacenter. The three value of health status is under loaded, normally loaded and over loaded. Each load balancer is connected to every other load balancer like a mesh topology with the shortest link, calculated with Dijkstra's algorithm. These shortest links are calculated once as the location of datacenter are fixed and do not change frequently. The connected load balancers share their load status with each other following these shortest links. Each load balancer sends its load status to every other load balancer it is connected to via the shortest path available.

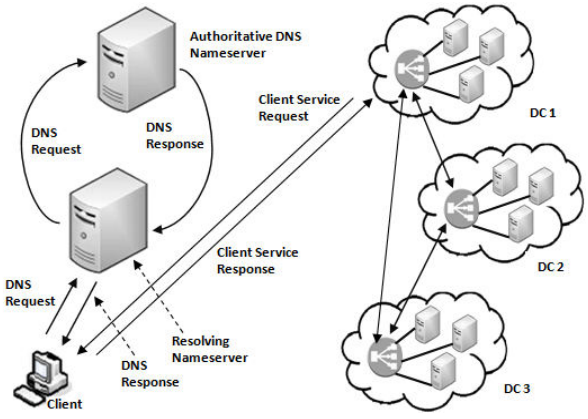


Fig. 3 Proposed GSLB Model

The load balancer consolidates the load information it has received from other load balancers and prepares a complete load table shown in Fig. 4(b) sorted on ascending load value. Thus each load balancer has the current health status of all the load balancers in the form of complete load table.

The complete load table is now used to direct the requests across the datacenter according to the geographic location of end client. As all the datacenter now have a list of least to most loaded datacenters with them i.e. a list with ascending load values of datacenter with their IPs. The updated values of IPs of these least loaded datacenters are then updated to the authoritative name server for the domain of the service and to the DNS servers of the ISP of the end client so that they always resolve to the IP address of least loaded datacenter.

| Datacenter IP | Load value | Health Status |
|---|---|---|
| 1.1.1.1 | 25 | Under loaded |

(a)

| Datacenter IP | Load value | Health Status |
|---|---|---|
| 1.1.1.1 | 25 | Under Loaded |
| 2.2.2.2 | 48 | Normal |
| . | . | . |
| . | . | . |
| . | . | . |
| 3.3.3.3 | 96 | Over Loaded |

(b)

Fig. 4 Data Structures of the proposed model.
(a) Load Table (b) Complete Load Table

## 4.2 Working of Networked Load Balancers System

The working of the proposed model is implemented with the help of two sub module named Load Assessor Module (LAM) and IP updater Module (IPM) as shown in Fig. 5. The two modules communicate with each other to pass the information. The following two sections describe their responsibility in detail.

### *A. Load Assessor Module (LAM)*

LAM manages the load calculation and load table preparation. It maintains a periodic interaction with IP updater Module (IPM) and updates load table entries. Thus it has the following responsibility.

- Load Assessment and preparation of complete load table: LAM calculates the load of datacenter and builds it load table in every 1800 seconds (30 minutes). It also consolidates the load information received from all other load balancers to prepare complete load table with ascending load value.
- Communication with IPM: LAM also communicates with IPM to exchange both types load tables.

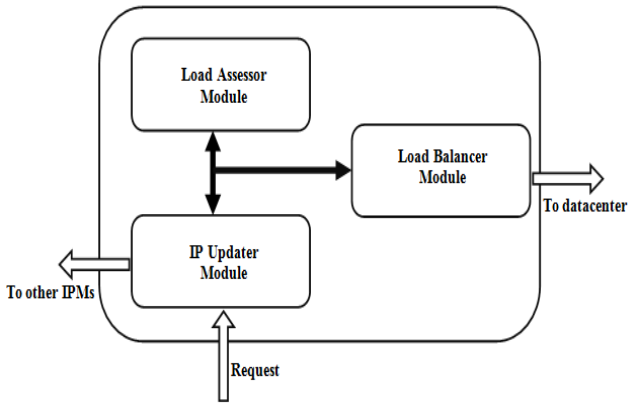


Fig. 5 Modules of proposed model

### *B. IP updater Module (IPM)*

Every load balancer in the system has an IPM module. This module takes care of inter and intra communication of load balancers. It communicates with the end client to accept or migrate the request to other datacenter and with the load balance module if the decision is taken to accept the request. Apart from this the following are its other important responsibilities:

- Accepting Load Tables: The module communicates with LAM for the transaction of load tables. It interacts with IPM of other load balancers to exchange the load tables.
- Sending Periodic IP Updates: This is the most crucial task of the whole system. It sends the authoritative name severs and ISP DNSs the least loaded server IPs. It may send multiple IPs to DNS or whole complete load table.
- Seamless transfer of request: When a service request comes to a datacenter the IPM examines the real IP address of end client and finally routes it to best datacenter based on the factors like health status and geo location of end client.
- Identifying the defunct datacenter: In case of natural disaster or power outage the whole datacenter may be down. IPM identifies such datacenter and does not route the request to such datacenters.

### *C. Working of LAM*

The most important job of LAM module is to calculate the load of datacenter. For this we propose a novel dynamic load calculation algorithm. The algorithm proposed is depicted in Algorithm 1. Based on the calculated load we categorise the health status of datacenter. A physical server hosts many virtual machines (VMs) in the virtualised cloud environment. The load of a server in principle can be calculated as the sum of loads of virtual machines it hosts [5].

The set of all physical machines (servers) in the datacenter is $S = \{S_1, S_2, S_3, \dots, S_N\}$ where N is the total number of servers in the datacenter. The set of all the hosted VMs on the server $S_i$, $V_i$ is given by $\{V_{i1}, V_{i2}, \dots, V_{im}\}$ where m is the number of virtual machines on server $S_i$.

The LAM module completes the load table shown Fig. 4(a) by executing the Algorithm 1 and delivers it to IPM. It also consolidates the complete load table depicted in Fig. 4(b) from the load tables of other load balancers routed to it by other IPMs in the network.

The load tables which are exchanged between the load balancers are very small in size as they just contain the record for itself and every load balancer prepares its own complete load table. The process of load table delivery from one IPM to another IPM is done with the help of user datagram protocol (UDP). UDP ensures the best effort delivery of load tables.

**ALGORITHM 1**
**DCHEALTHCHECK**

**Input:**
1) $C_{im}$: Processing power of virtual machine *m* of server *i*.
2) $M_{im}$: Memory resource utilization of virtual machine *m* of server *i.*
3) $B_{im}$: Bandwidth utilization of virtual machine *m* of server *i.*
4) $load_{max}, load_{norm}, load_{min}$

**Output:**
1) Health status of datacenter.

**Comments:**
1) $p, q, r$ are the weights assigned to $C_{jm}, M_{jm}$ $B_{jm}$ and $0 < p + q + r < 1$.

1: ***for*** *each* $S_j$ ***do***
2: ***for*** *each* $V_{jm}$ ***do***
3: $L(V_{jm}) \leftarrow p * C_{jm} + q * M_{jm} + r * B_{jm}$
4: $TL(S_j) += L(V_{jm})$
5: ***end for***
6: ***end for***
7: $TL(DC) \leftarrow \left(\frac{1}{N} \times \sum_{j=1}^{N} TL(S_j)\right)$
8: ***if*** $TL(DC) >= load_{max}$ ***then***
9: ***return*** *OVERLOADED*
10: ***else if*** $TL(DC) < load_{min}$ ***then***
11: ***return*** *UNDERLOADED*
12: ***else***
13: ***return*** *NORMAL*

*D. Working of IP Updater Module*

Our model does load balancing at two levels for the first level it directs the requests to the least loaded datacenters and at the second level it further consider the real IP address for geo sensitive load balancing if the first level of load balancing is not cost effective.

As soon as the IPM receives a request it finds out the IP address of resolving nameserver of end client by running an applet at the client side. Now to know the authoritative nameservers for the requested domain it sends NS (name server) type query with the fully qualified domain name in query name field of query message to the DNS of end client. Once it receives the response for this query it knows the authoritative nameservers and updates them with IP address of least loaded datacenter.

At the first level every IPM learns and caches the list of frequently used domain names and their corresponding authoritative name servers and sends them the unsolicited DNS responses consisting of IP address of least loaded datacenters. For every first request for a particular domain which the datacenter receives the IPM module sends a NS type (name server) query with the fully qualified domain name in query name field of query message to its resolving DNS server to know the list of all authoritative nameservers for the domain. It then updates its cache with all the authoritative servers and then sends them an unsolicited A (address) resource record containing the IP address of least loaded datacenter. As a result of this the authoritative name servers always contains IP address least loaded datacenter making TTL practically ineffective as they are forced to update their cache with unsolicited DNS response.

But we merely do not update the authoritative nameserver we also update the resolving nameservers (DNS of ISP) of end client. To update the DNS of ISP the model employs the secure and portable client-server architecture of java applets. The model will deploy an applet at client side (end user) to find out its default DNS server and send them A resource record for the least loaded datacenter. This ensures that all the clients who are using this nameserver as their default nameserver will always get the address of least loaded datacenter.

In cloud environment the least loaded server may not always be the optimal datacenter to deliver the services as response time to mission critical application deployed in clouds is also a critical factor at the global level of load balancing. To cater to the need of less response time, we have employed a second level of load balancing in our model. This balancing is done on the basis of geographic location. Once the request comes to least loaded datacenter the IPM module judges the feasibility of serving the request on the basis of response time (round trip time) and distance of the end client. This is done with

the help of real IP address of end client and not on the IP address of authoritative or resolving name server which helps in finding the exact location of client and the nearest least response time datacenter for the request. The IPM can forward the requests from one datacenter to another datacenter on the basis of response time and health status of datacenters. The request migration algorithm is depicted in Algorithm 2 for IPM module.

In case of disaster the IPM will not receive the load tables from the defunct datacenter and it will not consolidate the load information of such datacenters in the complete load tables. Such datacenters will automatically be left out of request forwarding business as they do not have an entry in the complete load table.

**ALGORITHM 2**
**MIGRATEREQUEST**

**Input:**
1) $H_L$: Health Status of least loaded datacenter
2) $H_N$: Health Status of nearest datacenter
3) $RT_L$: Response time from least loaded datacenter
4) $RT_N$: Response time from nearest datacenter
5) $IP_L$: IP address of least loaded datacenter
6) $IP_N$: IP address of nearest datacenter

**Output:**
1) IP address of datacenter to fulfil request

1: **if** $HN =$ $NORMAL\ or\ UNDERLOADED$ **then**
2: **if** $RT_N <= RT_L$ **then**
3: $return\ IP_N$
4: **else if**
5: $return\ IP_L$
4: **end if**
5: **else if** $H_N = OVERLOADED$ **then**
6: **return** $IP_L$
7: **end if**

## 5. Conclusion and Future Work

In this paper, we have proposed a new model for global server load balancing which gives resilient and intelligently balanced datacenters. Balancing load at two levels this framework considers the current health status of datacenter to deliver services from the least loaded datacenter to reduce the response time. It calculates the geographic location of end user with the help of its IP address to deliver services from the least loaded datacenter which is nearest to the user. Our future research will be directed towards comparing the implementation of proposed model with the existing practices in order to evaluate its overall performance improvements. We will continue to refine our model to move towards achieving greater efficiency specially by efficiently calculating the load of the datacenter in the dynamic cloud environment. We will also deploy and evaluate our model in real time environment.

## References

[1] Fox, Armando, Rean Griffith, Anthony Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, and Ion Stoica. "Above the clouds: A Berkeley view of cloud computing." Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS 28 (2009): 13.

[2] Rahman, Mosaddequr, Sajid Iqbal, and Jerry Gao. "Load balancer as a service in cloud computing." In Service Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on, pp. 204-211. IEEE, 2014.

[3] Lin, Qunyang, Junqing Xie, Zhiyong Shen, and Xunteng Xu. "DR3: Optimizing Site Selection for Global Load Balance in Application Delivery Controller." In Open Cirrus Summit (OCS), 2012 Seventh, pp. 11-15. IEEE, 2012.

[4] MacVittie, Lori. "Cloud Balancing: The Evolution of Global Server Load Balancing." F5 Networks, Inc (2010).

[5] Wollman, William V., Harry Jegers, Maureen Loftus, and Caleb Wan. "Plug and play server load balancing and global server load balancing for tactical networks." In Military Communications Conference, 2003. MILCOM'03. 2003 IEEE, vol. 2, pp. 933-937. IEEE, 2003.

[6] http://support.citrix.com/article/CTX123976

[7] O'Neil, Kevin, Robert Nerz, and Robert Aubin. "System for balancing loads among network servers." U.S. Patent Application 10/162,419, filed June 4, 2002.

[8] Hsu, Ivy Pei-Shan, David Chun Ying Cheung, and Rajkumar Ramniranjan Jalan. "Global server load balancing." U.S. Patent 7,454,500, issued November 18, 2008.

[9] Bahl, Pradeep, Feng Sun, Bernard D. Aboda, and Arnold S. Miller. "System and method for performing client-centric load balancing of multiple globally-dispersed servers." U.S. Patent 7,653,700, issued January 26, 2010.

[10] Bourke, Tony. Server load balancing. " O'Reilly Media, Inc.", 2001.

[11] Tenereillo, Pete. "Why DNS Based Global Server Load Balancing (GSLB) Doesn't Work." Aquired at: http://tenereillo. com/GSLBPageOfShame. htm 17 (2004).

[12] Cardellini, Valeria, Michele Colajanni, and S. Yu Philip. "Dynamic load balancing on web-server systems." IEEE Internet computing 3 (1999): 28-39.

[13] http://www.communitydns.eu/global-load-balancing-service.html

[14] Krapf, E. "Alteon's Global Server Load Balancing." Business Communications Review (1999): 60.