

EVALUATING AND EXPERIMENTAL ANALYSIS OF LOAD BALANCER AND SELF ORGANIZED WEBSERVERS USING HA-PROXY

Muhammad Shakeeb¹ and Bilal Muhammad Khan^{2*}

¹ Department of Electrical Engineering, National University of Sciences and Technology, PNEC Karachi, Pakistan (muhammad.shakeeb2015@pnec.nust.edu.pk)

² Department of Electrical Engineering, National University of Sciences and Technology, PNEC Karachi, Pakistan (bmkhan@pnec.nust.edu.pk) * Corresponding author

ABSTRACT: Self-organize and load balancing of computing system is one of the emerging research area. Despite of having robust infrastructure; several organizations still face system downtime due to hardware failure, high load, memory exhaustion and network choking. The best way to avoid downtime is to have the ability to predict and analyze the behavior of link and infrastructure. This paper presents an experimental implementation and evaluation of load balancing using HA-Proxy as a software-based load balancer and a proactive approach for self-organizing of network systems using certain algorithms in case of system failure. We are using Network function virtualization using XenServer that provides the ability to take network and traffic management. However, making sure that achieving such centralized system is not a trivial task [15] as it raises various issues relating to network control, system management, data security, package requirements, routing and data migration.

Keywords - Self Organizing, HA-Proxy, Load Balancing, Network function Virtualization and XenServer.

I. INTRODUCTION

To support service availability a load balancer is needed which requires two or more servers. A load balancer receives traffic from external sources and distributes these requests to multiple defined load balancer servers. This load balancing of web traffic can be implemented with special hardware, dedicated software's or some time uses a combination of both software and hardware [8]. Our design consists of two virtual web servers and on top of it a load balancer.

The ability to run such infrastructure elements as virtual machines or software appliances has been around for a long time. Virtual Machine Monitor (VMM) software supports to view, monitor and manage multiple virtual machines with a single host. Typical virtual machine monitor includes User Mode line / ESX Server, VMware Server and Xen Virtual Server [2]. Virtual Machine may act as a logical server and can be hosted in physical machines which have complete isolation from the main server as it runs its own instances. Each virtual machine act as a real machine having its own tools, software's, users and network configuration. This paper uses a Xen Server an open source tool used to host Linux OS [14]. Network performance, measurement and quality of a service can be achieved by the hardware probes of the system. However special configuration of network monitoring system is needed which run on all gateway services on single physical machine. Usually hardware probe work around system clock

and reports it to monitoring system with these probe results in real time in order to fully maintain quality of service and system production [4]. The reason of introducing virtualization in system and networks has attracted the attention of both industries and academic bodies. Several works have been done on System and Network Virtualization [10].

In any communication environment availability of the network is highly required. This performance factor of network availability can be determined by analyzing the connectivity of end-to-end process flow. In order to test the network connectivity ARP and ICMP packets are sent to test and diagnose the network connectivity [3]. The aim of the proposed algorithm is to gather information from multiple sources using network utilities and later on using algorithm including from the point of incident and make effective decisions including shifting of the network traffic from one VM to other VM in case of any disaster or system degradation. Moreover, the proposed algorithm provides a unique solution and work on all such physical machine [12].

The paper is organized as; Section3 presents system design. Section4 describes load balancing using HA-Proxy, Section5 describes syncing of data before service degradation using designed algorithm. Section6 practical implementation of load balancer and its results. Section7 concludes the paper.

II. System Design

Server configuration may vary according to the hosting of virtual machine. In this paper three virtual machines are used which are hosted on single physical machine that can be managed by Citrix Xen management and monitoring tool. VMMs includes VMware server/ESX server [17], User Mode Line [16] and Citrix Xen Server [5, 14]. VMware provides virtualization for both Linux and windows based architecture. VMware is a well-known commercial product used in production environment. Para virtualization present a modified interface having a new architecture for the interface of guest OS. Xen is a para virtualized virtual machine monitor that supports Berkeley Software Distribution (BSD) and a Linux based OS. UML also supports virtualization but have low speed than Xen VMM; therefore, in this paper XenServer is used which is configured on Physical Machine and to monitor it we will use Xen virtual machine monitoring app.

A. Xen Server

To implement network virtualization at least two separate physical machines required one having Xen Server installed, and the other to run the Xen monitoring tool i.e. Xen App. This paper uses a system with multiple core enabled with 64bit server-class machine to allow virtualization [13] [14]. Xen machine has xen-enabled kernel with an optimized Linux partition which is responsible to controls the interaction and its communication between virtual devices and the physical hardware [6]. Figure 1 shows XEN architecture.

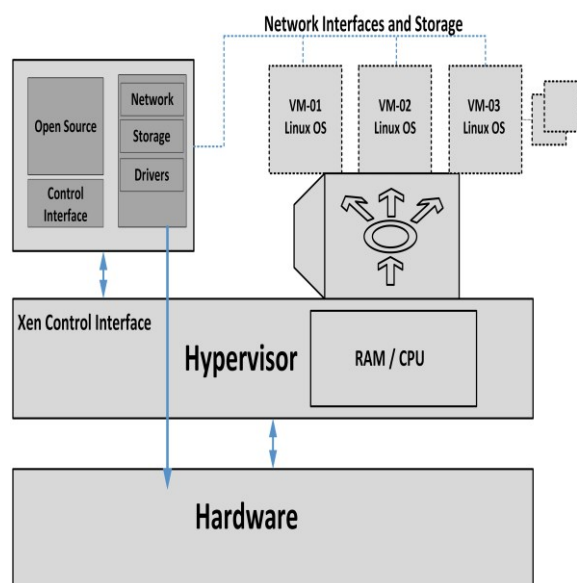


Fig. 1 XEN Architecture

B. Xen Remote Management

Xen App is installed on any supported operating system. It enables secure, remote access of Xen Hosted server. Through Citrix Xen app a remote management tool is used to install the virtual machine in Xen Hosted Server [2].

C. Xen Virtual Network

Xen enabled kernel creates virtual network, virtual interfaces like network interface cards for VMs and these Virtual network interfaces are bridge to real interface using Linux system utilities. Virtual machine has its own identity usually defined by the user which allows access from gateway machine [2] to access it remotely. We can setup a public IP as well to directly access it remotely. Figure 2 presents the internal architecture of Xen server hosted physical machine with its network interfaces.

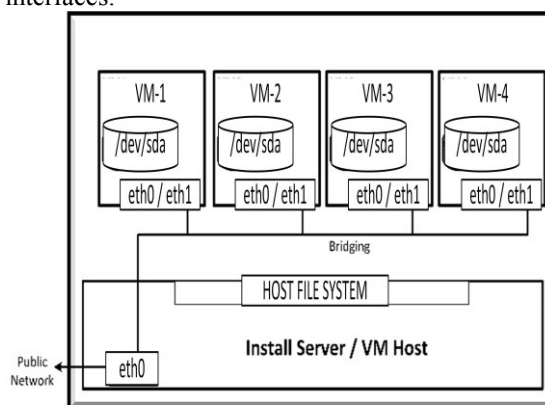


Fig. 2 Physical Machine

Table.1 shows system requirement to configured virtual machine. These may be varying as per the hardware size and system traffic requirement. In this paper three VMs are used two of these VMs has 1 network interface and the main load balancer have 2 network interface cards. Ubuntu as operating system is used for all the virtual machines.

Table.1 Virtual Machine Configuration Table.

System Req.	Virtual Machine		
	HOST-1	HOST-2	HA-Proxy
Processor	2700 MHz	2700 MHz	2700 MHz
RAM	512 MB	512 MB	1024MB
NIC	1	1	2
Disk	6 GB	6 GB	10GB

III. Load Balancing Using HA-Proxy

Load balancing services are offered by many of the cloud hosting providers and its demand increases so rapidly [19] [20] due to high internet usage. Social media popular websites like Facebook, twitter etc. have a huge number of clients which need to be served real time which causes high load on server due to server limitations response time increases so instantly some time causes server downtime. To maintain a tolerable service up response, several servers must be provided to prevent bottlenecks and allowed to forward the client requests to the backend servers to handle them [22]. User requests are dispatched using load balancing policies defined in load balancer to end servers. Load balancer using HA-Proxy uses a number of algorithms whose aim is to minimize the imbalance between different servers and distribute the load among in an optimal way.

Load balancer supports weighted and non-weighted algorithms: first alive, hash, round robin, least connections. Weighted least connection and weighted round robin are different policies used. Weighted algorithm checks a server with higher weights and sent the request to that server who has a higher weight. It checks the preference and determine to which server receives next request. The percentage of traffic divided to each server is approximately equal to its weight which is divided by the cumulative weight of all servers in the presented member group. Similarly A non-weighted algorithm assumes that the capacity of all servers in the group to be equivalent. Although non-weighted algorithms are typically faster than weighted algorithms.

A. First alive

The first alive algorithm uses the concept of a primary and secondary (backup) servers. The primary server is the first server in the members list define in ha-proxy configuration similarly the secondary servers are any subsequent server in the members list. This algorithm checks the health state of primary server if it is up it will forward all the traffic to primary server and if the health state of the primary server is down, the data power service forwards connections to the next server in the list.

B. Hash

The hash algorithm uses HTTP header and the IP address of the client as the basis for server selection. This property is available for only the Multi-Protocol Gateway and Web Service Proxy. Hashing algorithms cannot ensure evenly distributed connections. [9]

C. Least connections

Least connections algorithm stored a record of all the active server and its connections and is responsible to forward new connection request to the server which has less active connections. [9]

D. Weighted least connections

Weighted least connections algorithm generates a weighted list and maintains a list of application servers with active number of connections. The service forwards a new connection to a server on the basis of its weight or number of active connections. It uses more computation times so increasing response time therefore known as slow algorithm.

E. Weighted round robin

Weighted round robin algorithm originates a weighted list. On the basis of connection proportion to the weight it forwards new connections to high weighted server. The algorithm uses additional computation time but have the ability to distribute the traffic more efficiently to the server that is most capable of handling the request.

F. Round robin

Round robin algorithm uses a list of servers defined in HA-Proxy configuration and forwards a new client connection request to the next server in members list [9].

Among commonly available load balancing polices, round robin is the most common one supported by major cloud providers [21]. In this paper two apache webservers are installed over VM having IP address 192.168.1.5 (host-1) and 192.168.1.6 (host-2) both contains same data in its web root directory with same server configuration. Another VM with HA-proxy is used as the main server which handles all client request and forward it to these webservers. HA-Proxy Server IP is set to 192.168.1.7 which is public interface and all the requests are sent to this IP.

Round robin algorithm is used to test the load balancer. In this paper a test.php code file is created which shows load balancer result containing web server IP, Load Balancer IP (HA proxy server) and X-Forwarded IP (A system which request for this page) .

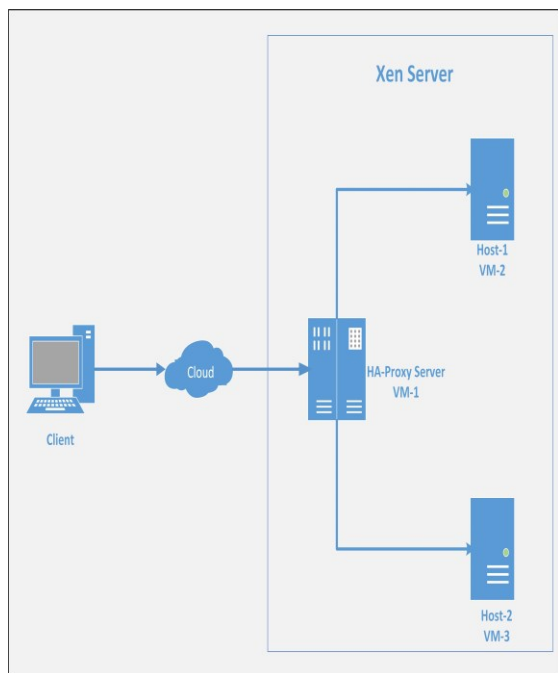


Fig. 3 HA-Proxy Architecture

To analyze round robin algorithm curl is used with a test script which shows below result. Curl is a tool to transfer data from or to a server which gives output as a command line.

```
shakeeb@Shakeeb-Laptop:~$ curl
http://192.168.1.7/test.php Server IP: 192.168.1.5
Load Balancer IP: 192.168.1.7
X-Forwarded-for: 192.168.1.4
shakeeb@Shakeeb-Laptop:~$ curl
http://192.168.1.7/test.php Server IP: 192.168.1.6
Load Balancer IP: 192.168.1.7
X-Forwarded-for: 192.168.1.4
shakeeb@Shakeeb-Laptop:~$ curl
http://192.168.1.7/test.php Server IP: 192.168.1.5
Load Balancer IP: 192.168.1.7
X-Forwarded-for: 192.168.1.4
shakeeb@Shakeeb-Laptop:~$ curl
http://192.168.1.7/test.php Server IP: 192.168.1.6
Load Balancer IP: 192.168.1.7
X-Forwarded-for: 192.168.1.4
```

Each client request has been verified by seeing that round robin algorithm works without any data loss and request has been handled by each server consecutively. First request is catered by the host-1 and second request by host-2 after that 3rd request is surfed by host-1 again and fourth by host-2 and so on which clearly shows round robin technique. We are using only two webservers as a load balancer so only two IPs appears as a server IP.

IV. Self-Organizing of Data before Service Degradation.

In this paper two webservers are created to cater client request and if there is any server failure to respond client request than another subsequent hosts is required to handle the request. To predict high load we have developed a script which checks system load and on certain defined threshold need to enable (Failover webserver). To develop this approach we have build replica server with same content, system packages. We can achieve this using XenApp tool which helps to create a complete snapshot of the VM. Snapshot takes some time to build complete image of the webserver only depends on the server size. Now a day's people create scripts to define program for all UNIX environment to break all complex projects into simple smaller task also known as bash scripting (Bourne again Shell) in Linux environment. In this paper an algorithm is developed using a shell scripting which uses a way to explore the capabilities in Linux environment [11] using same algorithm we have designed customized algorithm which checks all the updates from primary server host-1 and host-2. Sync the data to the backup server automatically. In our designed algorithm if the website content is change or update than this algorithm won't backup complete site but it takes incremental backup and ignore the old content which is already in backup server which helps to reduce time in sync of data from main server.

To examine system performance, load and memory test we are using a Perl script and on the basis of it if there is high load another bash script runs which sync the data. Syncing data is possible to any remote host without human intervention which generates exact copy of data to failover server and shifts the traffic if required using the designed algorithm. All these scripts are placed inside main HA-Proxy server which is responsible for load balancing test. In this paper, complete data backup size is approx. 1.2MB which takes 1.1MB/sec to transfer data. Using our designed algorithm if the primary webserver host-1 or host-2 gets down than an auto update runs which edit main configuration file of HA-Proxy and add new failover server which out any human intervention and down time.

V. Experimental Analysis of Load Balancer

Below are few experiments presented which measures performance analysis of using load balancer and using virtual machines having same system configuration with the help of apache bench test.

1. Web stress test without using multiple webservers (Host-1).
2. Web stress test by increasing host-1 memory to 1024MB.
3. Web stress test over load balancer using two 512MB webservers.

All experiments have been performed to check the performance on HA-Proxy server by increasing number of concurrent connection from luser to 100users and then analyze response time for each request. These results based on number of connections, response time, connection rate, and timeouts. We are using a demo webpage with a size of 1.2MB so that to analyze same page over all presented scenarios.

A. **First Scenario** has been tested by switching off all the web server except (Host-1) here host-1 has a memory of 512MB and to analyze its behavior a curl request is generated. Curl request shows test script result where X-Forwarded is responsible to take client request and forward it to Server. Here Server IP is static as we have enable single server so it shows server IP of host-1.

```
shakeeb@Shakeeb-Laptop:~$ curl
http://192.168.1.7/test.php Server IP: 192.168.1.5
Load Balancer IP: 192.168.1.7
X-Forwarded-for: 192.168.1.4
shakeeb@Shakeeb-Laptop:~$ curl
http://192.168.1.7/test.php Server IP: 192.168.1.5
Load Balancer IP: 192.168.1.7
X-Forwarded-for: 192.168.1.4
shakeeb@Shakeeb-Laptop:~$ curl
http://192.168.1.7/test.php Server IP: 192.168.1.5
Load Balancer IP: 192.168.1.7
X-Forwarded-for: 192.168.1.4
shakeeb@Shakeeb-Laptop:~$ curl
http://192.168.1.7/test.php Server IP: 192.168.1.5
Load Balancer IP: 192.168.1.7
X-Forwarded-for: 192.168.1.4
shakeeb@Shakeeb-Laptop:~$ curl
http://192.168.1.7/test.php Server IP: 192.168.1.5
Load Balancer IP: 192.168.1.7
X-Forwarded-for: 192.168.1.4
```

By generating curl request we analyze that single webserver is active. Table.2 shows web stress test of 1st, 10th, 50th, 51st, 99th and 100th number of request. After generating web stress test results shows that response time increases so instantly on increasing number of concurrent request. Initial request response time is approx. same but if we check the result of 100 concurrent users it reaches from 206.4ms to 718.572ms after 100th request which is 3.5 times greater than its first request.

Table.2 Response time without Load balancing
512MB server

Concurrent Traffic	Response time (ms)					
Single user	4.745	4.978	6.559	6.561	9.627	11.783
10 users	9.916	10.179	50.083	51.99	135.968	158.474
20 users	41.525	46.036	108.467	108.717	189.918	211.071
40 users	95.199	124.306	198.26	198.336	373.609	376.724
60 users	101.575	101.732	316.606	316.774	470.743	486.241
80 users	161.734	165.228	422.644	434.881	645.731	671.081
100 users	206.449	206.483	454.381	454.464	717.569	718.572

Figure4. Shows graphical representation when single 512MB webserver serves all client requests and the response time shows in milliseconds. By analysis graphical result it shows as much as the number of concurrent request increases response time also increases while after increasing number of test the load on the server also increases causes high load time which is much higher than load balancer and presented in our next scenario.

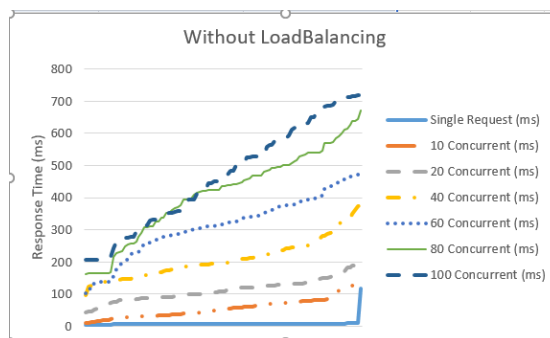


Fig. 4 Response time without Load Balancing Server with 512MB

B. **Second experiment** has been performed over 1024MB webserver which is equivalent memory of two webservers used in third scenario. In this experiment single VM has been created with same data and memory is set to 1024MB. To analyze the behavior using load balancer a curl request is generated which shows single server on load balancer is activated.

```
shakeeb@Shakeeb-Laptop:~$ curl
http://192.168.1.7/test.php Server IP: 192.168.1.5
Load Balancer IP: 192.168.1.7
X-Forwarded-for: 192.168.1.4
shakeeb@Shakeeb-Laptop:~$ curl
http://192.168.1.7/test.php Server IP: 192.168.1.5
Load Balancer IP: 192.168.1.7
X-Forwarded-for: 192.168.1.4
shakeeb@Shakeeb-Laptop:~$ curl
http://192.168.1.7/test.php Server IP: 192.168.1.5
Load Balancer IP: 192.168.1.7
X-Forwarded-for: 192.168.1.4
```

By analyzing above it only shows single server is activated. After generating apache stress test over load balancer server with single activated 1024MB memory Virtual machine. Table.3 shows response time of 1st, 10th, 50th, 51st, 99th and 100th request. Here Initial request response time is far better as compare to previous scenarios but if we check the result for 100 concurrent request it reaches from 195.412ms to 666.255ms which is 3.5 times of its first request this ratio is approximately equal to the 512MB webserver. Only the response time is less as compare to previous scenario.

Table.3 Response time without Load balancing
1024MB server

Concurrent Traffic	Response time (ms)					
	1st	10th	50th	51st	99th	100th
Single user	4.075	4.173	6.149	6.178	9.489	10.354
10 users	12.356	12.787	61.116	61.159	106.44	110.572
20 users	68.459	68.728	114.331	116.619	181.13	193.713
40 users	97.137	107.622	229.705	232.305	329.071	358.619
60 users	124.46	132.802	318.959	319.688	453.023	462.495
80 users	168.386	170.413	391.529	395.226	570.056	570.491
100 users	195.412	195.495	445.232	447.765	665.426	666.255

Below graphical representation shows all hundred requests response time. In Figure 5 we clearly analyses the response time using 1024MB of webserver is less as compare to 512MB webserver. After 100th request in previous scenario the response time is higher than 700ms and as per the below graphical representation if we increases system memory its response time is far better.

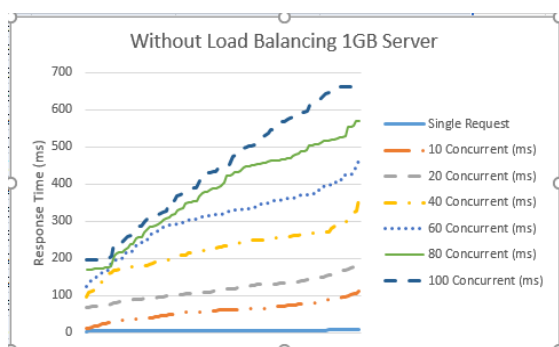


Figure5. Response time without Load Balancing Server with 1024MB

C. **Third experiment** has been performed on two 512MB webserver which is used as a load balancer also presented in Figure.3 .Minimum request time for single user is approximate same as previous scenarios but as soon as the number of concurrent request increases its response time is far better than previous scenarios. When a first requests generated which shows a response time little high with the previous one as some response time utilizes by the

HA-Proxy server to cater the request and forward it to next defined server in HA-proxy configuration. After 100 concurrent connection first request takes 241.282ms and reaches to 481.57ms which is approx. 2 times of initial request.

Below Table.4, shows the result of 1st, 10th, 50th, 51st, 99th and 100th test with defined number of concurrent users using load balancer. As per experimental analysis if the number of specific request increases the response time to surf single request also increases but have far better response time as compare to all previous scenarios.

Table.4 Response time with Load Balancing Server

Concurrent Traffic	Response time (ms)					
	1st	10th	50th	51st	99th	100th
Single user	4.921	5.33	6.932	6.958	9.049	11.116
10 users	21.007	24.35	39.713	39.916	65.686	78.748
20 users	31.954	52.461	90.535	90.753	148.195	148.37
40 users	97.309	102.25	167.51	167.57	239.44	239.49
60 users	137.762	137.86	229.13	230.19	358.28	359.45
80 users	172.273	172.32	262.18	262.19	424.62	439.24
100 users	214.282	214.38	351.55	353.76	475.97	481.57

For all apache bench test over HA-Proxy load balancer server presented in Figure 6. Shows that after 100 concurrent request response time is lesser than 500ms which is previously greater than 700ms. From all presented experiments response time using load balancer server is far better as compare to using single webserver with same system configuration.

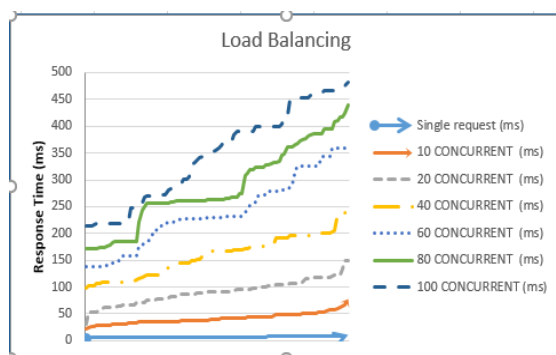


Figure6. Response time with Load Balancing Server

VI. CONCLUSION:

This paper presents an experimental analysis to investigate best possible method to introduce self-organizing within the network without incurring additional complexity and better response time using a described algorithm. This paper presents the difference between load balancer using HA-proxy webserver or virtual webserver. Using round

robin as load balancer algorithm performs the best in handling high traffic and with efficient implementation of shell script in order to organize servers is fast as compare to another load balancer algorithms. Our designed algorithm script helps to organize webserver and add another host on load balancer server configuration. Implementation of such centralized physical machine reduces hardware cost, human intervention, power consumptions and space per site. Designed algorithm helps in improving the traffic flow and smooth evacuation during shifting and switching of traffic from one host to another. As per our analysis round robin handle the request efficiently as the first client request is surf by the 1st host another client request is surf by 2nd host causes less load on the server which helps in fast response time to handle all client requests. It can also be concluded from the experiments that software switching is more reliable and efficient as compare to hardware switching in real time. This virtualization scheme improves the utilization of hardware which reduces the cost and its maintenance.

REFERENCES

- [1] Lizhe Wang, Gregor von Laszewski, Jie Tao, and Marcel Kunze, "Grid Virtualization Engine: Design, Implementation, and Evaluation IEEE systems journal, vol. 3, no. 4, December 2009
- [2] Stephen Childs, Brian Coughlan, David O'Callaghan, Geoff Quigley, John Walsh, "A single-computer Grid gateway using virtual machines," 19th International Conference on Advanced Information Networking and Applications, 2005 IEEE.
- [3] Zongjian He, Guanqing Liang, "Research and Evaluation of Network Virtualization in Cloud Computing Environment,"
- [4] Xiaoliang Li, Feng Liu, Zhenming Lei and Kun Zhang, self-organizing load balancing for network measurement server cluster," work is supported by National Natural Science Foundation of China. 2011 IEEE
- [5] Jiabin Wang, Lianhe Yang; Miao Yu; Wang "Application of Server Virtualization Technology Based on Citrix XenServer in the Information Center of the Public Security Bureau and Fire Service Department." IEEE July 2011
- [6] LinOxide [URL] 2016, online <https://linoxide.com/tools/configure-citrix-xenserver-6-5-vmware-workstation/>
- [7] M. Tatzono, N. Maruyama, and S. Matsuoka, "Making wide-area, multisite MPI feasible using Xen VM," in Proc. Workshop on XEN in HPC Cluster and Grid Computing Environments (XHPC), LNCS, 2006, vol. 4331, pp. 387–396.
- [8] Round-trip time (RTT) [URL] April 2007, Online <http://searchnetworking.techtarget.com/definition/round-trip-time>.
- [9] Load balancing technique and algorithms https://www.ibm.com/support/knowledgecenter/en/SS9H2Y_7.5.0/com.ibm.dp.doc/lbg_algorithms.html
- [10] J. Lee, J. Tourrilhes, P. Sharma, and S. Banerjee, "No more middlebox: integrate processing into network," SIGCOMM Comput. Commun. Rev., vol. 41, no. 4, pp. –, Aug. 2010. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2043164.1851262>
- [11] Advanced Bash-Scripting Guide [URL] 2014 online Available: <http://www.tldp.org/LDP/abs/html/why-shell.html>
- [12] Virtualization [URL] May, 2016 <https://en.wikipedia.org/wiki/Virtualization>
- [13] Hardwar [URL] Jun, 2012 <http://www.hardwaresecrets.com/everything-you-need-to-know-about-the-intel-virtualization-technology/>
- [14] CITRIX [URL] 1999-2016 Citrix Systems <http://docs.citrix.com/fr-fr/xencenter/6-5/xs-xc-intro-welcome/xs-xc-system-requirements.html>
- [15] Bilal Zafar; Soheyl Gherekhloo; Aidin Asgharzadeh; Mehdi Tavakoli Garrosi; Martin Haardt "Self-Organizing Network with Intelligent Relaying (SONIR)" "The 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2010)
- [16] Unified Modeling Language [URL] Nov. 2007 [Online]. Available: <http://www.omg.org/technology/documents/formal/uml.htm>
- [17] VMware Virtualization Technology [URL] Nov. 2007 [Online]. Available: <http://www.vmware.com>
- [18] Bash Script Algorithm [URL] 2016, Available : <https://linuxconfig.org/bash-scripts-to-scan-and-monitor-network>.
- [19] M. A. Adnan, R. Sugihara, and R. K. Gupta, "Energy efficient geographical load balancing via dynamic deferral of workload," in Proc. of IEEE Cloud. 188-195, 2012.
- [20] H. Goudarzi and M. Pedram, "Geographical load balancing for online service applications in distributed datacenters," in Proc. of IEEE Cloud. 351-358, 2013.
- [21] K. Bloor, R. Chirkova, T. Salo, and Y. Viniotis, "Heuristic-based request scheduling subject to a percentile response time sla in a distributed cloud," in Proc. of IEEE GLOBECOM. 1–6, 2010.
- [22] Agung B. Prasetijo, Eko D. Widiyanto and Ersya T. Hidayatullah "Performance Comparisons of Web Server Load Balancing Algorithms on HAProxy and Heartbeat" in Proc. of 2016 3rd (ICITACEE), Oct 19-21st, 2016.