American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-5, Issue-11, pp-187-194
www.ajer.org
Research Paper Open Access

Re-factory Load Balancing in a Cloud Computing Environment

Francisca O. Oladipo[1], Anigbogu G. Nkiru[2]
[1]*Computer Science Department Federal University, Lokoja Nigeria*
[2]*Computer Science Department Nwafor Orizu College of Education, Nsugbe Nigeria*

ABSTRACT: *The aim of this research is to re-factory the re-engineering the architecture of the DNS Round Robin Load Balancer in order to resolve the problems of server failovers which occurs in cloud-based systems during peak periods due to several technical and environmental factors. The Structured Systems Analysis and Design Methodology (SSADM) and the Rapid Prototyping were interface to form a hybrid creative methodology and deployed in the design of the new system. Java Server Pages were deployed in implementing the re-engineered load balancer because it allows for multiple plug-ins from several application layers and different vendors. MYSQL Database Management System provided the backend of the implementation due to its ability to handle large heterogeneous databases. The re-engineered Round Robin load balancing technique was implemented at the application layer of the cloud architecture to load balance by preventing server failovers through early pre-assignment detection such that traffic is diverted from over-loaded or failed servers. The system splits each website and databases into individual unit of work as well as monitors the clusters closely for load values; the loads generated by each unit is closely monitored and automatically transferred between cluster nodes such that no single node is ever overloaded. This is in contrast to the existing trial and error approach to load distribution as it supports sever affinity and availability.*

I. INTRODUCTION

Cloud computing in its simplest form is making computing and software services available to clients from a decentralized location. The computing model is based on virtualization, decentralization and provision of scalable services over the internet (Voorslays *et al*, 2011). This service covers both server as well as internet telephony point-to-point-like technology as was used in traditional network services (Srinivasulu *et al.*, 2012).

Historically, cloud computing emerged through an evolution process successfully passing through the phases of grid computing, utility computing and application service provision. The technology had come a long way after the original idea of an "intergalactic computer network" conceived by J.C.R. Licklider in the 60s and other modern day vendors like Amazon Web Services (AWS) and google had keyed into his idea of making delivery of computing resources over a network a reality and had gone ahead to make commercial success of it (Mohammed, 2009).

Cloud computing has several benefits. Noopur *et al*., (2012) believes that providing infrastructure outsourcing services saves times and leads to reduced glitches for businesses that utilize software programs for their management needs. The author also added that the fancy technology gives more virtual powers to users and due to the drastic reduction in use time of individual PCs, energy use is consolidated making the technology environment-friendly.

Cloud computing provides the flexibility required by organizations and business outfits whose network infrastructure demands are inconsistent as systems can scale up or down as the need arises. Other benefits of adopting cloud-based infrastructures according to Salesforce, UK include: provision of disaster recovery, automatic software updates, reduced cost of high-performance hardware, increase collaboration as team members can work on documents and share files from anywhere and anytime. In addition, data stored in the cloud is secure and recovery is easy as against storage on on-site based devices.

One major concern of cloud computing is efficiently allocating user requests as loads to the various nodes in order to prevent failovers. Load balancing techniques are used to evenly distribute requests within the network (Panek, 2010) by spreading tasks among available processors (Howe, 2010). The technique involves the automatic provisioning and deprovisioning of instances of applications without having to change the configuration of the network (Iannucci *et al.*, 2013).

In Nigeria, the level of acceptance of cloud computing services is still very low mainly because indigenous businesses are paranoid to about outsourcing their businesses and technology assets to third party organizations (Ogunjobi, 2015). While cloud computing has received good reviews and huge acceptance in developed countries, backed up by the respective governments, the uptake in developing countries like Nigeria remains comparatively low as indigenous businesses find it difficult to overcome their reluctance and resistance towards having their technology assets hosted and managed by third parties and this course is also not being helped by other environmental factors either.

This work is expected to serve as a medium of proving to pundits in Nigeria and other emerging economies who are reluctant to adopt the technology of cloud computing as a result of the fear of failovers that there are several technologies to deal with data traffic congestion in servers within the cloud technology through the development of a prototype load balancer. The work will contribute in no small way to demonstrating that high availability of outsourced network and storage resources is possible through the detection of host failover and automatically re-distributing web sites to available servers, such that congestions at peak periods are not noticeable.

This research is concerned with the development of a prototype cloud computing load balancer based on a re-factory of the DNS Round Robin architecture. The work will not address the privacy concerns in the design of the architecture nor will it provide over-site functions in the areas of design security for the cloud –based architecture. A brief discussion of previous related research will be presented in the next section. This will be followed by a detailed description of the methodologies and design elements. A detailed description of the result will be presented next and the paper concludes with some recommendations.

II. REVIEW OF RELATED RESEARCH

A study by (Ogunjobi, 2015) revealed that an estimated over 90% of Nigeria's Financial Services Institutions (FSI) would rather invest in their technological infrastructure than adopt cloud computing services.

Cisco (2015) conducted a reality check on the state of cloud adoption in Africa and discovered that South Africa is leading with approximately 50% of businesses in the country haven adopted cloud services. This is in contrast to Kenya and Nigeria's 48%and 36% respectively. The study also found that about 24% of organizations in Kenya were considering cloud services adoption as at the time of conducting the study.

The benefits of adopting and utilizing cloud-based services and infrastructures were presented by (Mohammed *et al.*, 2015). The researchers identified the energy challenge, poor internet infrastructure in addition to lack of trust as some of the major barriers to adopting cloud computing services in Nigeria. The authors recommended enlightening both public and private organizations on the benefits of cloud adoption to eliminate their fears.

Onamade & Adedayo (2014) defined a framework for cloud services adoption in Nigerian universities. The research identified huge cost as a major barrier to entry and recommended a group-based approach where institutions can team up to jointly finance the subscription to a community cloud architecture. This is feasible as most of the universities share common goals and mission statements.

A survey of cloud computing load balancing algorithms was conducted by (Nuaimi *et al.*, 2012). The study identified and briefly discussed the various challenges against developing optimal solutions to load balancing problems in addition to presenting a detailed discussion of several algorithms with special emphasis on their characteristics. The researchers went further to compare the named algorithms based on network load, time series, and number of utilized attributes.

A Taxonomy of Load Balancers

Private Beta Load Balancer developed by Rackspace and used a technique referred to as ***transparent sharding*** to ensure linear scalability of cloud databases. It provides a high availability similar to those of clusters' and users can create in a matter of seconds, robust, scalable and high availability cloud storage (Odom, 2010). While Azure Traffic Manager is designed to provide excellent user experience through a global routing of traffic and based on four policies: latency, failover, nested and round robin. Azure Load Balancer is programmed for regional traffic direction. These are combined to achieve a combination of global traffic management regionalized failover (Lasnoski, 2015).

The pros and cons of different load balancing algorithms was discussed in a paper by (Nuaimi, *et al.*,2012). The analysis of the algorithms was done using metrics such as nodes' capabilities, network bandwidth, run-time properties, communications between clients and nodes, etc.

A dual-stage architecture for increasing scalability, reliability and failover prevention in Session Internet Protocol (SIP) networks using the clustering of SIP proxy server was developed by (Karimi e*t al.*, 2010). In addition to response time of SIP proxy server, the approach used also included the number of connections to each SIP proxy server as there were different call durations for all calls.

Other similar researches include a work on Linux Virtual Server IP load balancing by (Zhang, 2012), a Divisible Load Scheduling (DLS) based load balancing algorithm by Padhy and Rao (2011) and a cloud partitioning based load balancing model for public cloud developed by Xu *et al*., (2013).

III. METHODOLOGY

The following are the materials used in this research:

1. A creative methodology obtained by interfacing the Structured Systems Analysis and Design Methodology (SSADM) and the Rapid Prototyping Methodology (Figure 1). The SSADM was used for the analysis of the sample information system. Specifically, the early part of the SSADM involved the analysis of the re-factory candidate, the DNS RRB and the definition of the requirements for the re-engineered system. Prototyping methodology was deployed in building the prototype re-engineered DNS RRB. It is expected that this sample will continue to be refined until the desired level of quality or result is achieved.
2. Java Server Pages(JSP) was chosen to implement the prototype RRB load balancer (front end) while MYSQL Database Management System was used to build the backend.
3. Color codes were used as indicators of the current state of each server.

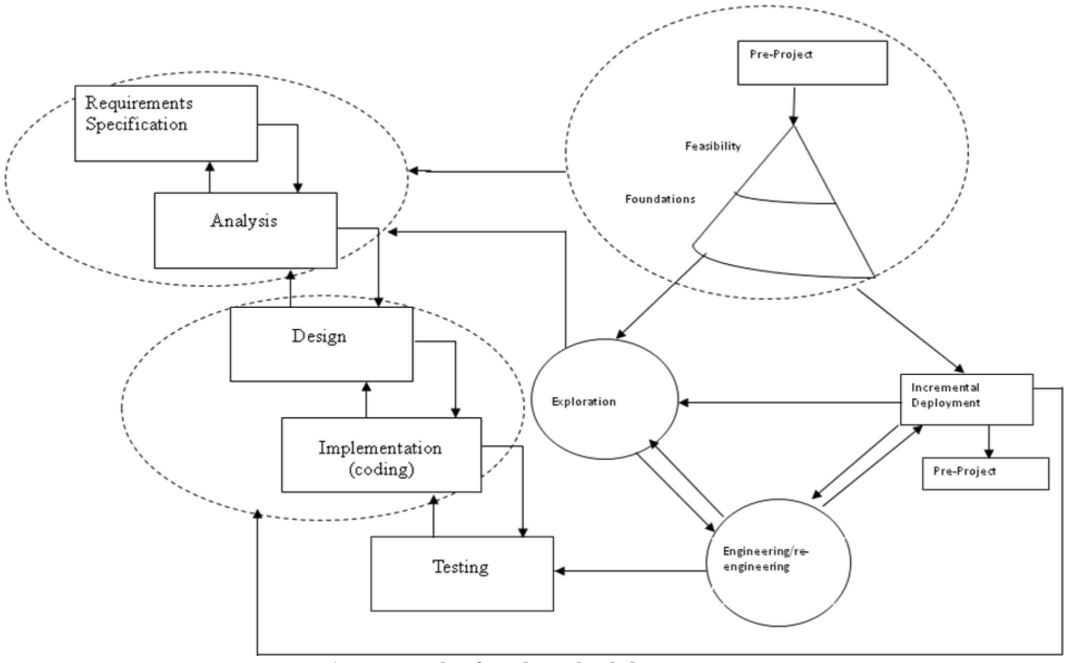


Figure 1. The fused methodology

IV. RESULTS DISCUSSIONS AND CONCLUSION

The re-factory process is in two-phases: A definition of a re-factory architecture at the application layer, and the implemetation of a prototype RRLB based on the defined architecure. The high level model resulting from the re-factory of the DNS RRB is built on the Tomcat clusters system (Figure 2). Each application is accessed through the load balancing path and the prototype system consists of three servers where loads and network traffics are distributed.

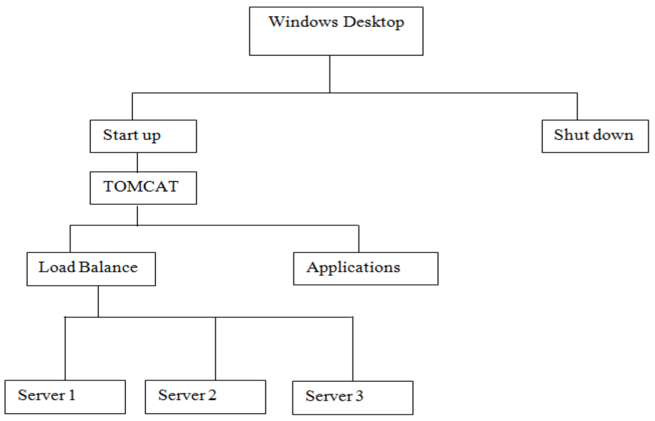


Figure 2. The high level model resulting from the re-factored DNS RRB

The RRB was re-factored to load balance at the application layer to load balance by preventing server failovers through early pre-assignment detection such that traffic is diverted from over-loaded or failed servers. The system splits each website and databases into individual unit of work as well as monitors the clusters closely for load values; the loads generated by each unit is closely monitored and automatically transferred between cluster nodes such that no single node is ever overloaded.
Fugure 3 illustrated the architecture of the load balancing prototype.

- User login
- User makes request to use server
- Load balancer checkes if any available and active server
  - If server is available and active, then the request is shifted to active server
  - If server is not available and active then control is returned back to load balancer checks for active and available server
- The load balancer checks also whether an available is not busy otherwise control moves back to searching any available and active server

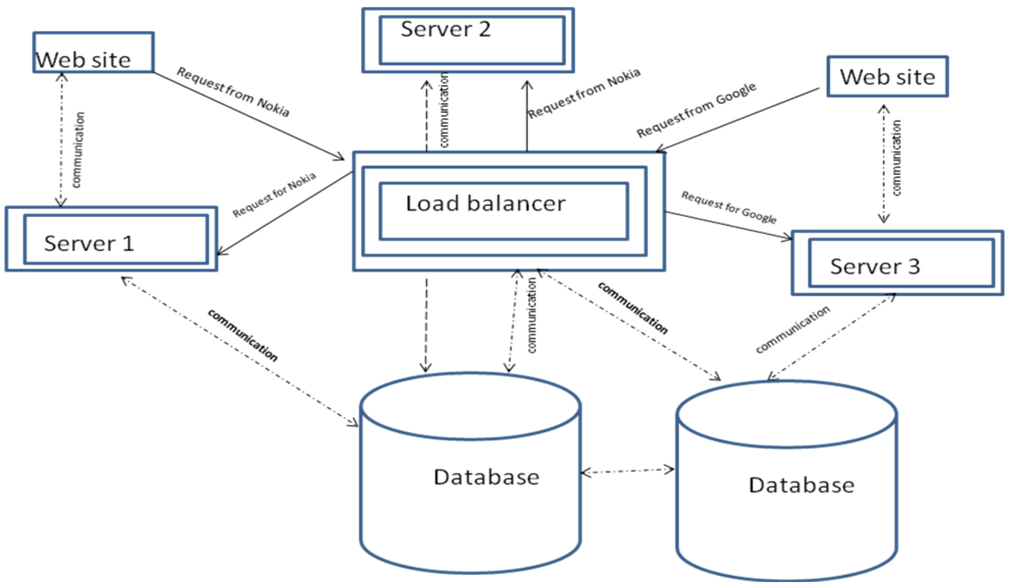


Figure 3: Prototype Architecture of the re-factored RRLB

The design idea surrounding this is to illustrate the design thinking in this system. Three web sites are making request to use their respective servers and the requests are channeled to a load balancer who in turn finds out from the available servers, which one to serve the request. The load balancer communicates back by assigning a server to a user request, a server also communicate with all the existing database for load balancing update and to satisfying the user. The process runs across all the three platforms of; PaaS, IaaS and SaaS respectively. However, the load balancer is located at the application layer since most of the request come from users through different applications (Figure 4)

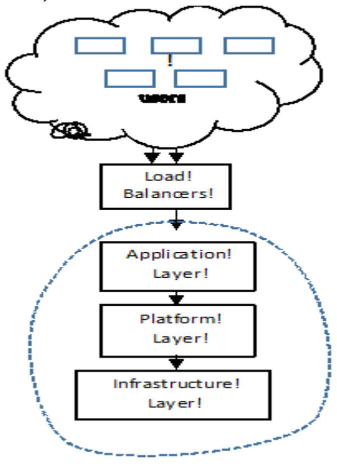


Figure 4 A Prototype architecture of load balancer at the application layer

The overall DFD across the entire prototype is shown in (Figure 5). Following our design, the system begins with a user starting the program from the desktop by double-clicking on the start up icon. This stimulates the runing of the Tomcat window which is minimized after which the browsers can be launched and the load balancer is run from the first browser window before other urls of desired websites are entered in a second window. This is to enable the visualization of the load balancing process in color codes where white indicates free servers, yellow indicates a working but not overloaded server, red indicates an overloaded server, grey indicates crashed server while green indicates a load-balanced/recovered server. Figure 6 shows the prototype load balancing screen monitor at zero request and Figure 7 shows a sample source code. More sample GUI from the re-factory are shown in the appendix.

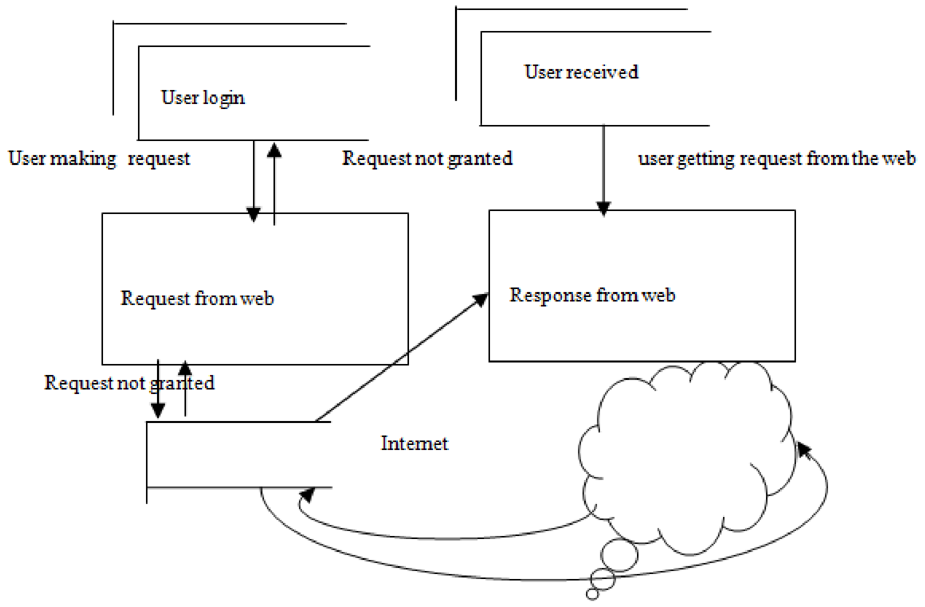


Figure 5: The Overall Dataflow Diagram

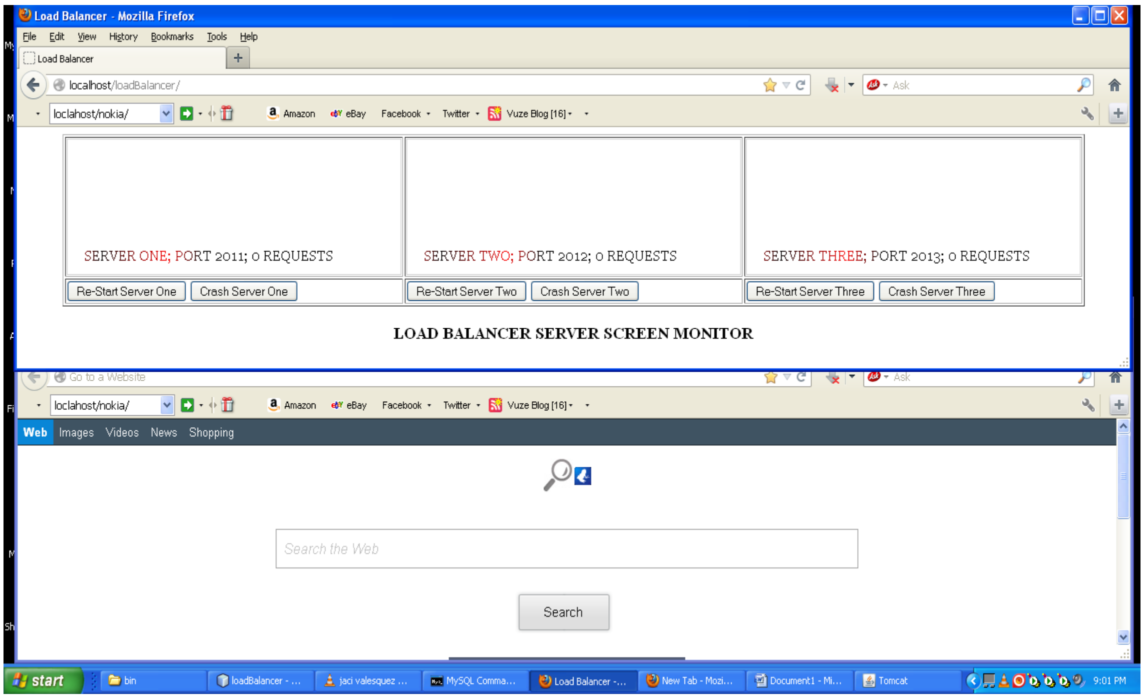
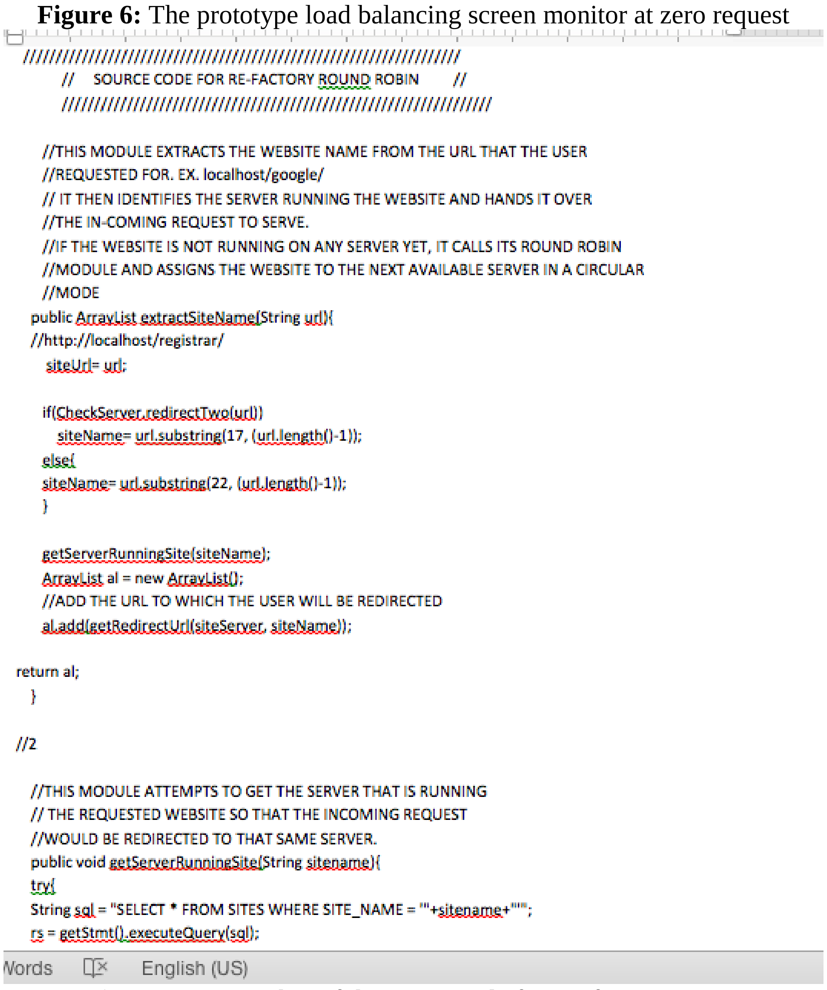

Load Balancer - Mozilla Firefox
localhost/loadBalancer/
SERVER ONE; PORT 2011; 0 REQUESTS
SERVER TWO; PORT 2012; 0 REQUESTS
SERVER THREE; PORT 2013; 0 REQUESTS
Re-Start Server One
Crash Server One
Re-Start Server Two
Crash Server Two
Re-Start Server Three
Crash Server Three
LOAD BALANCER SERVER SCREEN MONITOR
Search the Web
Search

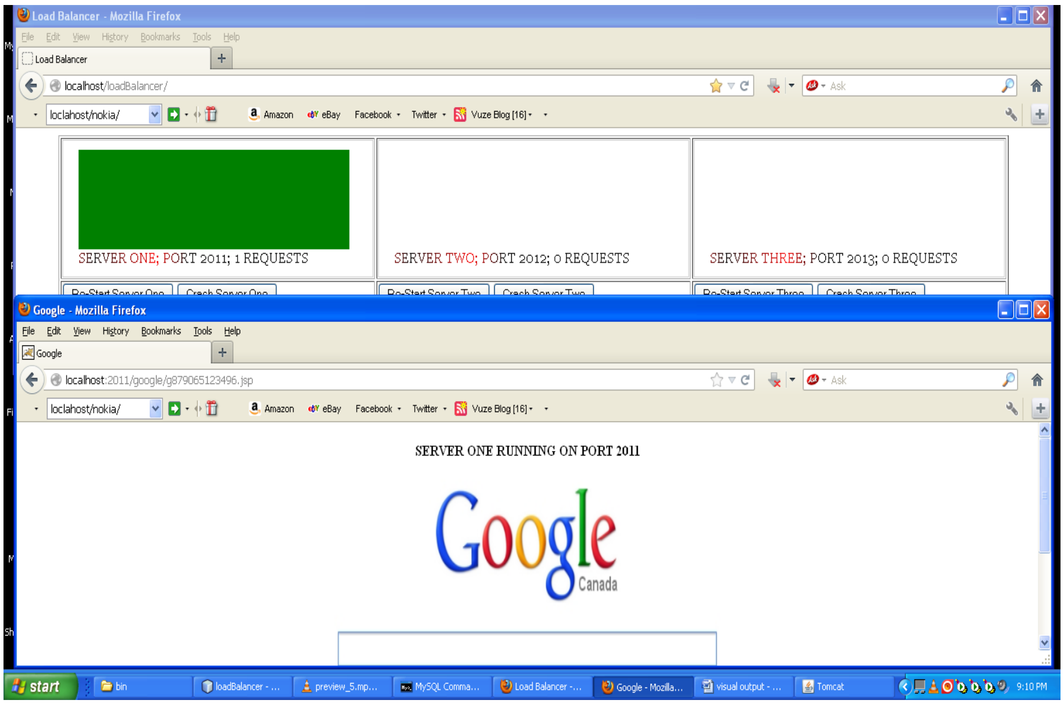
Figure 6: The prototype load balancing screen monitor at zero request



```
//////////////////////////////////////////////////////////////////////
      //    SOURCE CODE FOR RE-FACTORY ROUND ROBIN        //
      //////////////////////////////////////////////////////////////////////

   //THIS MODULE EXTRACTS THE WEBSITE NAME FROM THE URL THAT THE USER
   //REQUESTED FOR. EX. localhost/google/
   // IT THEN IDENTIFIES THE SERVER RUNNING THE WEBSITE AND HANDS IT OVER
   //THE IN-COMING REQUEST TO SERVE.
   //IF THE WEBSITE IS NOT RUNNING ON ANY SERVER YET, IT CALLS ITS ROUND ROBIN
   //MODULE AND ASSIGNS THE WEBSITE TO THE NEXT AVAILABLE SERVER IN A CIRCULAR
   //MODE
  public ArrayList extractSiteName(String url){
  //http://localhost/registrar/
     siteUrl= url;

   if(CheckServer.redirectTwo(url))
      siteName= url.substring(17, (url.length()-1));
   else{
   siteName= url.substring(22, (url.length()-1));
   }

   getServerRunningSite(siteName);
   ArrayList al = new ArrayList();
   //ADD THE URL TO WHICH THE USER WILL BE REDIRECTED
   al.add(getRedirectUrl(siteServer, siteName));

return al;
  }

//2

  //THIS MODULE ATTEMPTS TO GET THE SERVER THAT IS RUNNING
  // THE REQUESTED WEBSITE SO THAT THE INCOMING REQUEST
  //WOULD BE REDIRECTED TO THAT SAME SERVER.
  public void getServerRunningSite(String sitename){
  try{
  String sql = "SELECT * FROM SITES WHERE SITE_NAME = '"+sitename+"'";
  rs = getStmt().executeQuery(sql);
```

Figure 7: Screenshot of the sourcecode for Re-factory RRLB

Appendix
Screenshots from the prototype system


SERVER ONE; PORT 2011; 1 REQUESTS

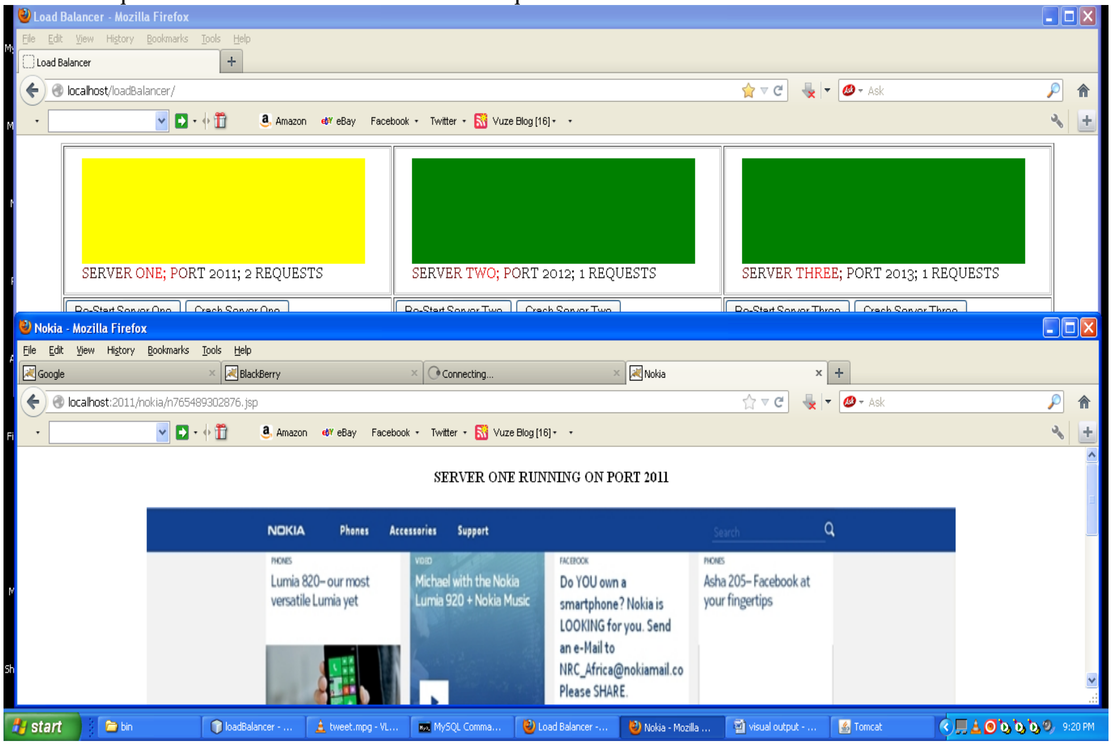
SERVER TWO; PORT 2012; 0 REQUESTS

SERVER THREE; PORT 2013; 0 REQUESTS

localhost:2011/google/g879065123496.jsp

SERVER ONE RUNNING ON PORT 2011

A website is requested and Server One serves the request


SERVER ONE; PORT 2011; 2 REQUESTS

SERVER TWO; PORT 2012; 1 REQUESTS

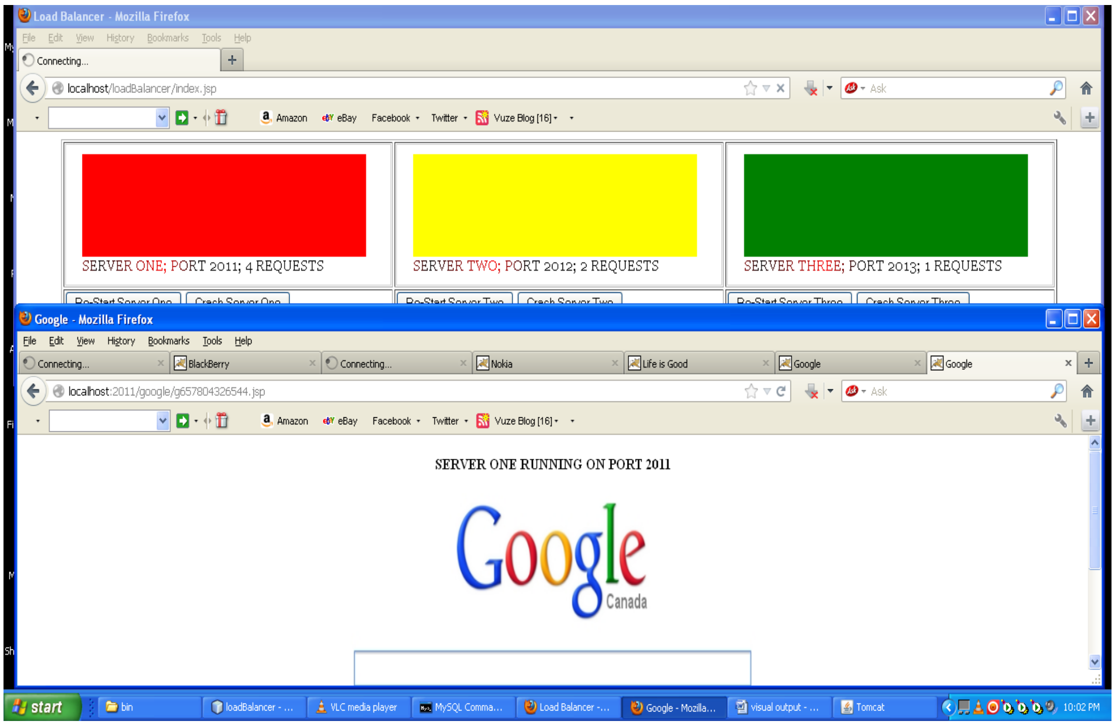
SERVER THREE; PORT 2013; 1 REQUESTS

localhost:2011/nokia/n765489302876.jsp

SERVER ONE RUNNING ON PORT 2011

Load Balancer - Mozilla Firefox
localhost/loadBalancer/
Nokia - Mozilla Firefox
NOKIA
Phones
Accessories
Support
Lumia 820– our most versatile Lumia yet
Michael with the Nokia Lumia 920 + Nokia Music
Do YOU own a smartphone? Nokia is LOOKING for you. Send an e-Mail to NRC_Africa@nokiamail.co Please SHARE.
Asha 205– Facebook at your fingertips
start

Server Three crashes and the load balancer automatically redirects its website to Server Two; the next less busy Server at that moment


SERVER ONE; PORT 2011; 4 REQUESTS

SERVER TWO; PORT 2012; 2 REQUESTS

SERVER THREE; PORT 2013; 1 REQUESTS

SERVER ONE RUNNING ON PORT 2011

Load Balancer - Mozilla Firefox
localhost/loadBalancer/index.jsp
Google - Mozilla Firefox
localhost:2011/google/g657804326544.jsp
Connecting...
BlackBerry
Nokia
Life is Good
Google
Canada
start

Google receives two more requests and Server One which serves both Google and Nokia turns red to show it is gradually **becoming** overloaded.

References

[1]. Voorslays, W., Broberg, J., and Buyya, R., (2011): Introduction to Cloud Computing Chapter 1 pages 1-41 Cloud computing principles and paradigms John willy & sons inc

[2]. Srinivasulu, B., Karthikeyan, K., Nirmala, M., and Chandramouli, M., (2012): Comparative Analysis of Cloud computing Utilities. Journal of Emerging Trends in Computing and information sciences ISSN 2079-8407 vol 3

[3]. Mohammed, A. (2009) A history of cloud computing, Computer Weekly Magazine, Tech Target USA

[4]. Noopur, M., Shweta, S., and Rupa, G., (2012): Cloud base CRM Application. Proceedings published by international journal of computer Applications (IJCA) ISSN: 0975-8887 pp 17-21

[5]. Panek, W.,(2011): MCLS Windows Server 2008 R2 complete study giude. Retrieved from http://www.intermec.com/support/downloads/search.aspx?productnodeid=CMPTR700COLOR

[6]. Howe, D. (2010): Load balancing in Technology. Retrieved April 2012 from http://foldoc.org

[7]. Iannucci, P., and Gupta, M., (2013): IBM Smart Cloud: building a cloud Enabled Data Center, IBM Red Books

[8]. Ogunjobi, M. (2015), Barriers to cloud computing adoption in Nigeria. Retrieved September 2016 from http://guardian.ng/technology/barriers-to-cloud-computing-adoption-in-nigeria/

[9]. Cisco, The Cloud in Africa: Reality Check, 2013. Retrieved December 15th, 2014 from: http://www.cisco.com/web/ZA/press/2013/112813.html

[10]. Muhammed, K., Zaharaddeen, I., Rumana, K, Turaki, A. M (2015), Cloud Computing Adoption in Nigeria: Challenges and Benefits. International Journal of Scientific and Research Publications, Volume 5, Issue 7, ISSN 2250-3153

[11]. Onamade, A.A & Adedayo, O.S (2014), A Framework for Cloud Computing Implementation for Nigerian Universities, Journal of Advances in Scientific Research & Applications(JASRA), Faculty of Science, Adeleke University, Ede, State of Osun, Nigeria

[12]. Nuaimi, K., Mohamed, N., Alnuaimi, M., Al-Jaroodi, J. (2012). A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms. in: Proceeding NCCA '12 Proceedings of the 2012 Second Symposium on Network Cloud Computing and Applications, IEEE Computer Society Washington, DC, USA, Pages 137-142.

[13]. Odom. J., (2010): Announcing cloud load balancing; Private Beta http://www.rackspace.com/blog/announcing-cloud-load-balancing-private-beta/

[14]. Lasnoski, N. (2015). Azure Traffic Manager vs. Azure Load Balancer. Downloaded 01 November 2016 from https://www.concurrency.com/blog/w/azure-traffic-manager-vs-azure-load-balancer

[15]. Karimi, A., Sarram, M., and Ghasemzadeh, M., (2010): Two stage Architecture for load balancing and failover in SIP network middle –East Journal of Scientific Research pp 6(1):88-92 ISSN 1990-9233

[16]. Zhang, W., (2010): Linux Virtual Server for Scalable Network services. National Laboratory for Parallel & distributed processing, Changsha, China

[17]. Padhy, R.P., and Rao, P. G., (2011): Load balancing in Cloud computing System. Deparment of computer Science & Engineering, National Institute of Technology, Rourkela Orissa India

[18]. Xu, G., Pang, J and Fu, X., (2013): A Load balancing Model Based on Cloud Partitioning for the Public cloud. Tsinghua Science and Technlolgy ISSDN 1007-0214 vol 18