INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING & TECHNOLOGY (IJCET)

ISSN 0976 – 6367(Print)
ISSN 0976 – 6375(Online)
Volume 5, Issue 8, August (2014), pp. 71-79
© IAEME: www.iaeme.com/IJCET.asp
Journal Impact Factor (2014): 8.5328 (Calculated by GISI)
www.jifactor.com

IJCET

©IAEME

LOAD BALANCING – SERVER AVAILABILITY ISSUE

T.N.Anitha[1], Dr. Balakrishna.R[2]

[1]Research Scholar, Dept of CSE, Rajarajeswari College of Engineering, Chickballapur-562101
[2]Professor & HOD, Dept of ISE, Rajarajeswari college of engineering, Mysore road, Bangalore

ABSTRACT

Recent research works done on various load balancing techniques on web servers has disclosed that the load balancing mechanisms used by most of the popular websites and web applications which are always overloaded with huge volumes of user requests, fail to provide fast and reliable services to its users. There could be many reasons behind it but the most common of all is server unavailability. Load balancing mechanisms provide guarantee of distributing load on server so that multiple server always get equal amount of load but they can't guarantee ever availability of the server it is redirecting request on. There could be a possibility where the server which is having minimum load goes down due to routing maintenance or any technical issues and load balancing mechanism may forward user requests constantly to that server due to minimal load without checking its availability. This type of problem may lead to downtime of entire application. In this paper, we evaluate the performance of proposed method for mitigating availability issue and compare the results with existing system.

Key Words: Load Balancing, Availability, Virtualization.

I. INTRODUCTION

The massive rise in internet traffic is the result of ever-growing user requests on famous Websites and web applications. Due to heavy volume of user requests, these websites sometimes fail to provide fast service to its consumers, especially in certain occasions, such as online results of universities or cricket scores. Overload on particular server may deteriorate the condition.[1] Administrators of these websites always look out for better mechanisms which can improve service rate of their websites.

One approach to manage well-liked Web sites is on the basis of duplication of data across an emulated-server architecture that offers a list of self-determining URL sites which have to be manually chosen by the users. This mechanism consists of many drawbacks, comprising the not

User-translucent structural design and the shortage of authority on the request sharing by the Web server method.[2] An additional hopeful resolution is to dependent on a dispersed architecture which is able to direct inward requests between numerous server nodes in a user-translucent method. This method could possibly perk up throughput function and offer Web-server applications with elevated flexibility and accessibility.[5] on the other hand, numerous confronts have to be highlighted to create a dispersed server group operation competently because a single server inside the framework of the HTTP protocol and Web browsers.[7]

In this paper we explain and talk about the diverse methods on routing requests between the dispersed Web-server nodes and importance of availability checking while redirecting user request to the server. We examine the competency and the confines of the various methods and the transaction among the alternatives. The scope of this article is not to explain the comprehensive technical characteristics of each one method, for which we refer the reader to the suitable piece of works. Its objective is somewhat to examine the features of each method and its efficiency over Web-server flexibility.

The rest of the paper is organized as follows: Section II describes the literature survey, section III describes the benefits of load balancing, section IV describes the design methodology, SectionV experimental results and VI conclusion.

II. LITERATURE REVIEW

Group of computer systems are interconnected by multisystem hardware or software to give services which was usually provided by only one system. [8]Without disturbing other servers functioning Servers can be added to the cluster. As the server is added, work will begins to transfer to that server by decreasing the pressure on the present server.[4]Distribution of users around several independent systems may lead to waste of capacity on several systems whereas some of them may get congested. Utilizing intelligent load balancing among cluster of systems, users are distributed to available systems depend on the load of each system. [6]

Any particular instance belonging to end user can be put off for maintenance or for any other reason without terminating services to the user. Without intimating to the user, service can be continued by connecting to the available server. It is a prime characteristic of load balancing mechanism that user request can be automatically redirected to the most suitable web server without the knowledge of end user.[9]

Redundancy and failover are techniques used to achieve high availability. Apache JMeter was used to load test the functional behaviour and measure performance based on throughput for dynamic querying of Project Monitoring data using Java servlets on Tomcat Web server. [11] Reliability issues associated with highly available applications – applications that need to remain operational and rapidly responsive even when failures disrupt some of the nodes in a system. But Web Services systems are also expected to automatically find servers and configure themselves, and then to operate securely and reliably in a completely automated manner.

III. SERVER LOAD BALANCING AND ITS BENEFITS

Functionality of load balancing is utilized to load balance through various web server farms, by means of scattering congestion to the crucial web services most efficiently. Server load balancing is the process of distribute service requests across a group of servers. Server load balancing deals with some of the requirements for Server which are becoming highly vital for critical applications. [10]

a. Increased scalability – Most of the content-intensive applications had extended above the level in which an individual server can offer sufficient processing power and throughput. Flexibility is required for both the ventures and service providers to install extra servers immediately and transparently to users. Because of Server load balancing number of server works as a single server.

b. High performance – High performance is accomplished at the time when the processing power of servers is utilized intelligently. Highly developed load-balancing techniques may transfer end-user request to the server which is less occupied and so it will be able to provide quick response time. Load-balancing device must be able to handle the collective traffic of multiple servers.

c. High availability and disaster recovery – Another one advantage of server load balancing is that its capability to improve application availability. In case of application or server failure on one server, user request can be redirected automatically to other server in the cluster which is available. Also it is helpful at the instances where any hardware or software is.
Server clustering is in high demand, as it is capable of handling expected number of simultaneous user requests and following traffic, and it also lessens single point of failure for a robust system uptime. So as most of the important applications are made available to the end-user in an association by making the application crucial. Redundancy and failure are the methods used to gain high availability.[13] This is done by gaining numerous copies of server. So there will not be only one web server, two web servers will be there, first one is in active state and in case if it fails then second server will get active. Rather than this one more option is to use a cluster of active servers where all the servers are in active state. Load-balancing method scatters the load between the clients of the cluster. In case if one to two members of the cluster fail other users are not affected.

IV. METHODOLOGY FOR AVAILABILITY

The steps used to develop our proposed techniques

1. Receive User Request and keep in a Queue
2. Get list of servers
3. Find individual server load status
4. Find server with minimum load
5. Check availability of the least load server using INetAddress
6. If available then forward user request to that server
7. Else except this down server go to step 5

Initialize down server, upper bound of down servers as 0 and minimum load as 1000. We keep minimum load as 1000 for calculation purpose as we assume that maximum load also on any server would not exceed 1000. Q is assumed to be set of User Requests in a Queue and LS is assumed to be the set of all servers present. For every User Request UR which is present in Q, we have to find minimum load server and its availability and redirect UR to minimum load server. So, list of all servers is fetched and each server load is measured. Load of particular server can be measured by summing up all user requests belonging to it. If server load is found to be the minimum among all the servers in LS then minimum load server is checked for availability. To check availability of server, INetAddress class is used where its method is Available() finds out if server is available or not. It uses general ARP pings to find availability, if server acknowledges the ping and respond then it is assumed to be alive and it returns true else it returns false to denote its unavailable. If minimum load server is found to be up then UR is forwarded to the minimum load server else if server found to be down DS then it is added in the list of Down Servers LDS.[11] After removing

LDS from LS, again the same steps are followed to find second least load server till available server is found and UR is finally forwarded to minimum load server which is available.

Notation	Description
UR	User Request
Q	Queue
M	Upper Bound of User Request
UR_k	Particular User Request in Queue
LS ǀ S	List of Servers ǀ Server
N	Upper Bound of Server List
LDS ǀ DS	List of Down Servers ǀ Down Server
A	Availability of Server
A_{s_i}	Availability of ith Server, where i>=1 and i<=n
L ǀ L_{s_i}	Load ǀ Load on Particular Server
L_{min}	Minimum Load
$L_{min_{s_i}}$	Server with Minimum Load

```
1.  $DS \leftarrow \emptyset,\ o \leftarrow \emptyset,\ L_{min} \leftarrow 1000$
2.  $Q = \{UR_1,\ UR_2, .., UR_m\}$
3.  $LS = \{S_1,\ S_2, .., S_n\}$
4.  $\boldsymbol{for\ all}\ UR \in Q\ \boldsymbol{do}$
5.      $\boldsymbol{for\ all}\ S \in LS\ \boldsymbol{and}\ i \leq n\ \boldsymbol{do}$
6.          $L_{s_i} \leftarrow \sum UR\ \in S_i$
7.          $\boldsymbol{if}\ L_{s_i} \leq L_{min}\ \boldsymbol{then}$
8.              $L_{min} \leftarrow L_{s_i}$
9.              $L_{min_{s_i}} \leftarrow s_i$
10.         $\boldsymbol{end\ if}$
11.     $\boldsymbol{end\ for}$
12.     $A_{s_i} \leftarrow INetAddress.isAvailable\left(L_{min_{s_i}}\right)$
13.     $\boldsymbol{if}\ A_{s_i} \equiv true\ \boldsymbol{then}$
14.         $UR_k \rightarrow L_{min_{s_i}}$
```

```
15.     $\boldsymbol{end\ if}$
16.     $\boldsymbol{else}$
17.         $DS \leftarrow L_{mins_i}$
18.         $LDS \leftarrow LDS \cup \{DS\}$
19.         $LS \leftarrow LS - LDS$
20.         $\boldsymbol{Repeat}\ Step\ 5$
21.     $\boldsymbol{end\ else}$
22. $\boldsymbol{end\ for}$
```

Before checking availability, user request has to be kept in queue so that there should be proper scalability of server load status. Once user request are accepted in queue, dispatcher has to get the list of all servers. Availability of servers is not checked at the time of taking list, because pinging takes some time to check availability and checking availability of all the servers initially is more time consuming than to check availability of one server. As servers get down rarely for maintenance, so, occurrence of event is not so frequent and is more time efficient than to check each server availability at initial stage.

V. EXPERIMENTAL RESULTS

Experimental results are carried out using JMeter Software for testing the authenticity of the concept. JMeter is a specialized software which can forward simulate ***n*** number of user quests as load on server and check efficiency of server in varying conditions. To achieve the result we make two prototypes of application where first prototype does just load balancing and fails to handle user request where less load server gets down. This is considered as existing system. On the other hand one more part of application is designed where availability of server is checked before forwarding user request to the server. The results which are generated automatically through JMeter is shown below as diagrams and is described while explaining results produced by the graphs.

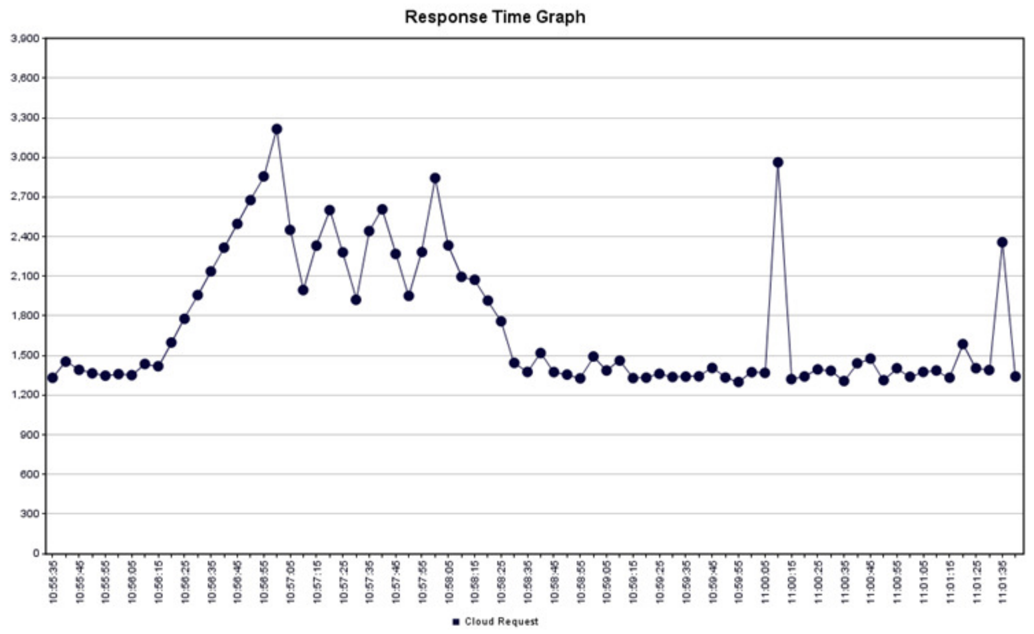


Fig 1: Existing System Response Time Graph

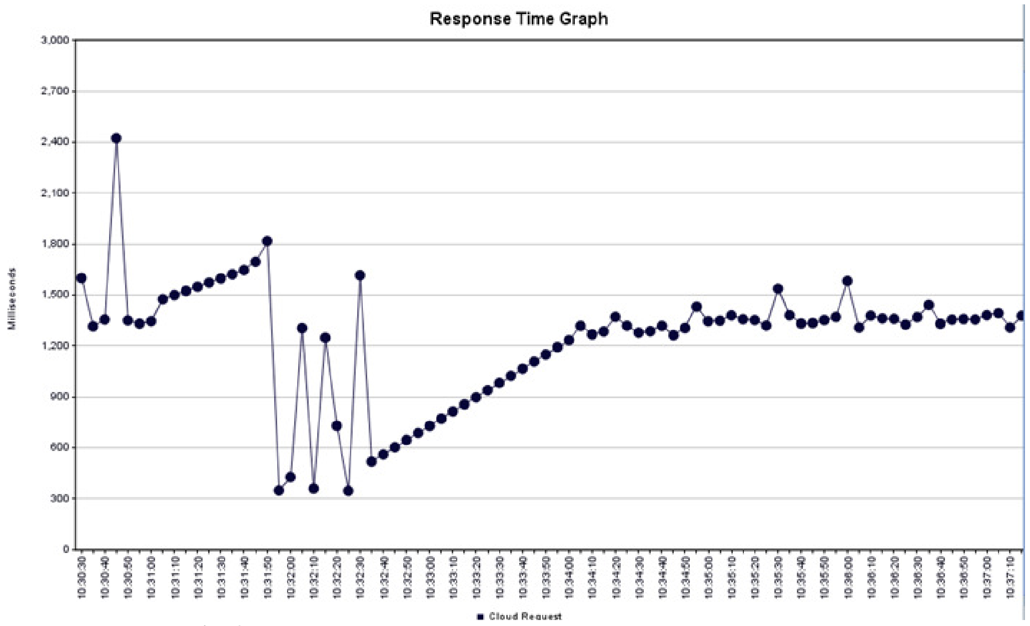


Fig 2: Proposed System Response Time Graph

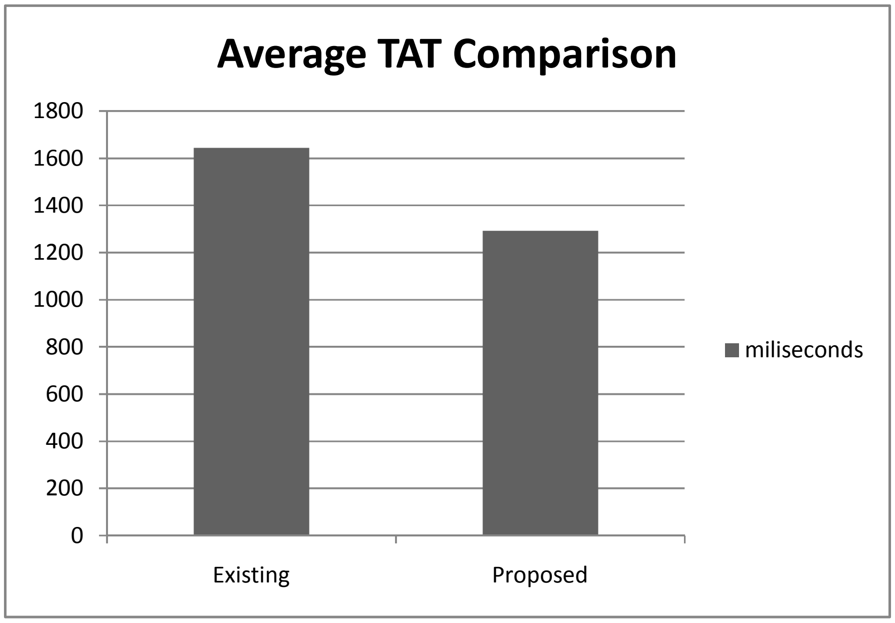
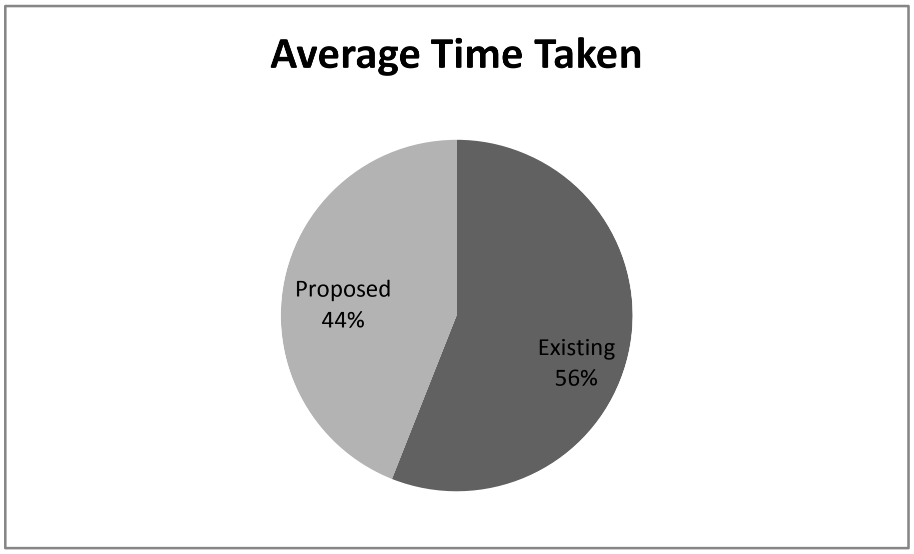
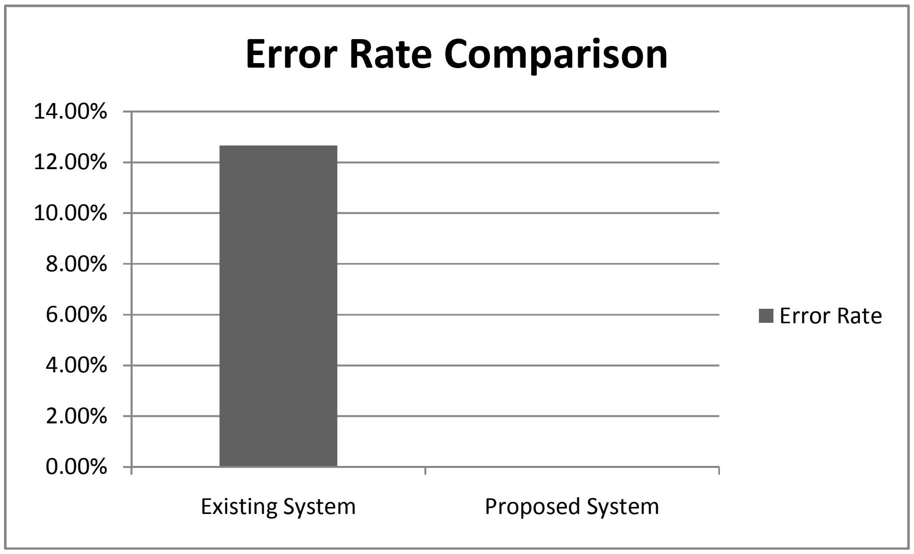


Fig 3: Average Turn Around Time Comparison

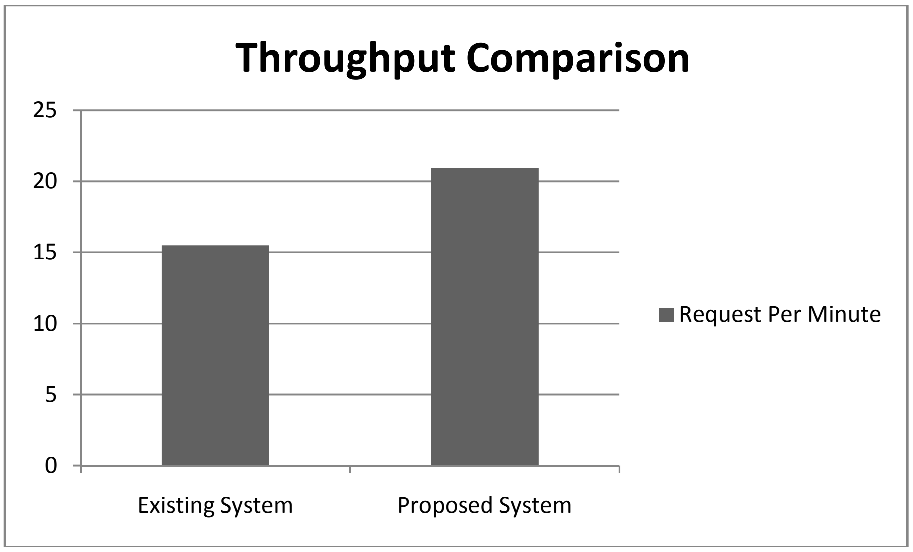
Figures 1, 2 & 3 mentioned in above displayed Turn-Around-Time graph depict that average time taken by servers to provide response to user in existing system is approx 1600ms-1700ms i.e., 1.6 to 1.7 seconds. After applying the proposed mechanism to the same server, it can be clearly justified that proposed system has more efficiency in providing fast response to users. There is an edge of more than 350ms, which matters a lot when website is highly popular and servers are bound to provide great service to its users.



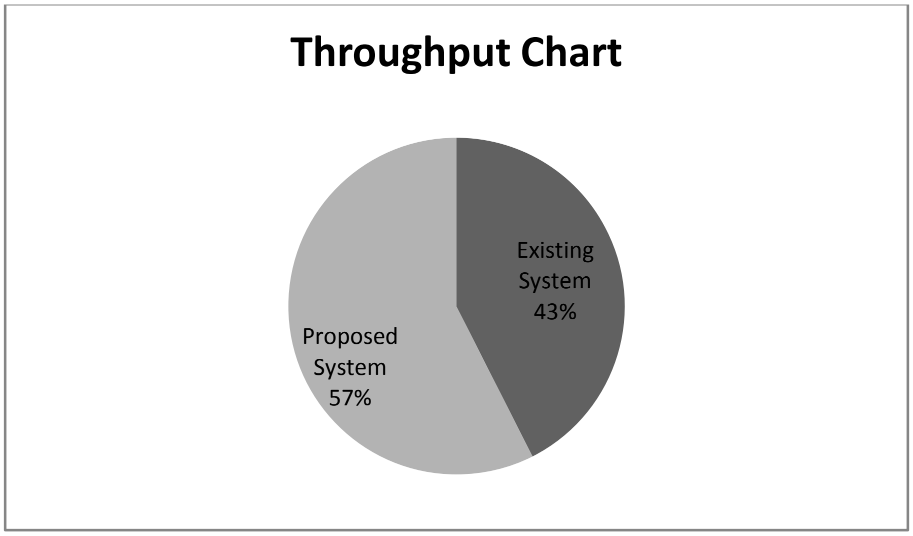
Above graph presents the same TAT comparison in a different fashion. Here figures are presented in percentage to provide more concise view. It is very much clear with this figure that proposed system has taken just 44% of the overall time compared to existing system which has taken huge 56% of TAT.



Above graph shows overall error rate in running of the application on server. Here existing system accounts for almost 13% of the error rate in comparison to 0% in proposed system. This error rate comes while switching the server when one server goes down. Here in existing system user request is still forwarded to the server which is down regardless of its unavailability because of its less request load. Commonly load balancing algorithm just checks for server load but often forget to see the availability of the server before processing the request.



Above figure depicts throughput comparison in bars where proposed system shows clear edge over existing system. Here proposed system shows throughput of approx 21 compared to existing system which counts not more than 16.



This chart represents throughput comparison between existing and proposed system. As it is clearly seen in the diagram that existing system provides overall only 43% of the throughput compared to the proposed system which provides 57% throughput which in turn makes it 14% more efficient than existing system throughput.

VI. CONCLUSION

After analysing all the aspects it can be clearly defined that Load balancing alone does not guarantee uptime of any server in cluster. As it is inevitable for servers to go through routine maintenance and due to this there would always be some downtime for users accessing the application on that server. The role of ever availability of server is much vital in today's advance technology. Our testing on JMeter software proves the authenticity of our concept and better results in case of overload on server and at the downtime of any server in cluster.

Though this method works well for normal load balancing but still there is scope of work when it comes to load balancing on content aware servers. In future works this mechanism would be implemented and experimental results would be analysed to see its consistency.

REFERENCES

[1] M. Armbrust *et al.*, "Above the clouds: A berkeley view of cloud computing," University of California, Berkeley, Tech. Rep., Feb 2009.

[2] L. Siegele, "Let it rise: A special report on corporate IT," in *The Economist*, Oct. 2008.

[3] *J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in Proc. of the ACM Symposium on Operating System Principles (SOSP'01), Oct. 2001.*

[4] *C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A scalable application placement controller for enterprise data centers," in Proc. of the International World Wide Web Conference (WWW'07), May 2007.*

[5] G. Chen, H. Wenbo, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *Proc. of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'08)*, Apr. 2008.

[6] *M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments," in Proc. of the Symposium on Operating Systems Design and Implementation (OSDI'08), 2008.*

[7] *M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: Fair scheduling for distributed computing clusters," in Proc. of the ACM Symposium on Operating System Principles (SOSP'09), Oct. 2009.*

[8] *M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," in Proc. of the European conference on Computer systems (EuroSys'10), 2010.*

[9] *T. Sandholm and K. Lai, "Mapreduce optimization using regulated dynamic prioritization," in Proc. of the international joint conference on Measurement and modeling of computer systems (SIGMETRICS'09), 2009.*

[10] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proc. Of the Symposium on Networked Systems Design and Implementation (NSDI'07)*, Apr. 2007.

[11] Alok jain, High availability setup and Performance improvement for RRCAT Information Portal using Server Load Balancing published in International Conference on Cloud, Big Data and Trust 2013, Nov 13-15, RGPV

[12] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *Proc. of the IFIP/IEEE International Symposium on Integrated Network Management (IM'07)*, 2007.

[13] *A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: integration and load balancing in data centers," in Proc. of the ACM/IEEE conference on Supercomputing, 2008.*

[14] *Y. Toyoda, "A simplified algorithm for obtaining approximate solutions to zero-one programming problems," Management Science, vol. 21, pp.1417–1427, august 1975.*

[15] *D.Asir, Shamila Ebenezer and Daniel.D, "Adaptive Load Balancing Techniques in Global Scale Grid Environment", International Journal of Computer Engineering & Technology (IJCET), Volume 1, Issue 2, 2010, pp. 85 - 96, ISSN Print: 0976 – 6367, ISSN Online: 0976 – 6375.*