Middle-East Journal of Scientific Research 23 (7): 1545-1553, 2015
ISSN 1990-9233

DOI: 10.5829/idosi.mejsr.2015.23.07.22578

Prediction of Cluster's Capacity in Openflow Enabled Networks

[1]Mohd A. Saifullah and [2]M.A. Maluk Mohamed

[1]ANNA Universities M.A.M College of Engineering,
Computer Science and Information Technology, Siruganur, Tiruchirapalli, India
[2]ANNA Universities, M.A.M. College of Engineering, Information Technology, Tiruchirappalli, India

Abstract: The explosion of web users and data center services has led to a need to cut down considerable energy footprint of data center. Software-Defined Networking (SDN) has emerged as a promising solution to solve the challenge as it provides the flexibility for each user by giving the programmatic control and traffic statistics. Server load balancing policy plays a critical role to achieve scalability and better quality of service offered by cluster of web servers. In this paper, we present Server load balancing strategy which predicts the capacity of the cluster and reduces the energy footprint (PCCSDN) using SDN. PCCSDN policy is capable of switching off the extra servers which are not required during night time, weekend and holidays thereby reduces the energy footprint of the data center and continuously predicts the service usage to take care of unexpected traffic. We show that using PCCSDN strategy, the energy footprint of the data center is reduced by 37.9%. This is achieved by taking the near-real perspective of the service requests using the measurement of flow counters at that moment and predict the future service requests up to ~ 10 minutes. Open Flow provides statistics: packet count and byte counts are used for predicting the size of the cluster.

Key words: Software Defined Networks · Cluster Computing · Server Load Balancing · Linear Prediction

INTRODUCTION

As the demand on web based services is increasing, the load that web servers need to support is also growing but the customers expect high availability of service and better response times. Hence, service providers need to offer the services with very high performance to keep the existing clients happy and attract new clients [1-23]. Server load balancing solves these challenges as it makes several web servers take part in the same web service and share the load as the service capacity of a single web server is limited. Thus, Server load balancing gives critical benefits like availability, scalability, security and manageability of Web services. One of the most popular types of web server load balancing is cluster based web servers [4, 11, 21, 18]. In this option, the content is replicated on multiple servers of the cluster to achieve important benefits. But this requires robust load balancing strategy to achieve the benefits. A server load balancing strategy consists of distributing or assigning the tasks of a parallel application across the available servers in a cluster. The best load balancing strategy avoids the situation where some servers are idle while others are busy and have multiple jobs queued up [9, 1, 7, 12].

Currently, Software Defined Networking (SDN) is attracting the significance of both industry and research. SDN offers separation of data plane and control plane, OpenFlow protocol is efficient of collecting per-flow statistics, this is becoming a key element to network optimization algorithms and for predicting the size of network. SDN is getting lot of consideration from research perspective as performing experiments is becoming easier that were difficult earlier due to vendor-dependant information. Present SDN-based web cluster solutions have to solve the challenge of energy usage by the data center. At the same time the proposed solution needs to consider the over loads conditions and flash crowds or sudden high demand that is common in the context of current service requests [24-30]. In this proposal we are addressing the reduction of energy footprint by shutting

Corresponding Author: Mohd A. Saifullah, ANNA Universities M.A.M College of Engineering, Computer Science & Information Technology, Siruganur, Tiruchirapalli, India.

down the unused servers during the night time and weekend as the service request load is low in normal conditions and also predicting the service usage to take care of over load condition or flash crowds as this is bigger challenge compared normal condition of service requests [5, 2, 13, 19] by restarting the required number of servers.

The rest of the paper is organized as follows: in Section 2 gives the motivation behind this work. In Section 3, we introduce the proposed architecture of SDN-based web clusters. Section 4 gives the functional description of OpenFlow based controller. In Section 5, we present the proposed load balancing strategy we designed and implemented. In Section 6, we describe the experimental setup used evaluate the performance of our proposal. Section 7 discusses the related work and the need for our proposal. Section 8 gives conclusions and future work.

Motivation: Web service hosting needs server resources according to the service demand at the expected service request loads. Since the data center for web service hosting is an essential requirement for the business, data centers should manage the server resources efficiently. It is better to do capacity planning of required resources and efficient admission control to limit the risk of failing to meet the worst case demand instead of over provisioning the resources [15, 16]. An efficient scheme for server resource management will assign minimum server resources to each service request and releases extra resources to be used by other service requests. Provisioning options of the resource management scheme has to accommodate the deviations in the service request load as they take place. Due to these causes managing resources in a automated way becomes a hard challenge.

The algorithms of computer science are used for resource management but actual dimensions of these resource requirements are more random than predictable [24]. The same is upheld by the perception that servers do many computations and executions within a small fraction of time and serve lot of interrupt replies so predictability turns hard. Efforts are made to study the patterns of server resource usage first and then those perceptions are used to predict the size of the forthcoming requests of server resource usage [25, 27]. Even though these attempts are successful to some extent, forecasting is not factual for longer durations. The prediction of longer timescales uses training first and then predicts the human interaction patterns [28, 29]. Patterns of utilization are observed in numerous contexts like utilization of gasoline during summer, national telephone calls on mother's day and airline passenger movement in airports during the year end timeframe (from Thanksgiving to New Year's Day). Although these studies [28, 29] deals with water usage or traffic patterns, the human patterns occur in computing too, which may be forecasted by employing identical techniques?.

World Cup 1998 web traffic analysis is a good instance from literature of human patterns in server computations [30]. Figure 1 depicts that soon after game begins, web service request traffic increases sharply and falls down as the game ends. The same pattern is seen for the full World Cup series, summarized in Section 5.2 of Martin Arlitt and Tai Jin's investigation report [30]. The lines which are vertical show the beginning of a World Cup game. Table 1 lists the teams participated in each game with its abbreviation and expansion. For instance, around 2pm on Monday June 15, the game was played between England (ENG) and Tunisia (TUN) at that time web requests were received at a rate of 6.1 million per hour.

Vimal Mathew *et al.* [31] proposed energy-aware load balancing system in content delivery networks with the experimental tradeoffs between reduction of energy, wear-and-tear of hardware because of server transitions and availability of service which is important for customer SLAs. They developed techniques for saving of energy both for global and local load balancing components of CDN. They developed two algorithms: one was offline algorithm and other was online algorithm. The offline algorithms that reduces the utilization of energy are derived by changing the number of active servers needed to handle incoming requests. Their offline algorithm achieved a good amount of reduction of energy (64.2%) at system level. In the online mode Hibernate, an load balancing algorithm was proposed which takes decisions depending upon past and present work load but not the load of future. Their results depict that good amount of reduction of energy is possible in content delivery networks if those systems are re-designed with the concern of energy [32-40].

The Proposed Architecture of SDN-Based Web Cluster: The proposed SDN-based web cluster architecture is shown in Fig. 2. The proposed cluster consists of mainly a OpenFlow based controller, a OpenFlow based switch

Table 1: Team Name Abbreviations

Abbreviation	Team	Abbreviation	Team	Abbreviation	Team	Abbreviation	Team
ARG	Argentina	DEN	Denmark	ITA	Italy	NOR	Norway
AUT	Austria	ENG	England	JAM	Jamaica	PAR	Paraguay
BEL	Belgium	ESP	Spain	JPN	Japan	ROM	Romania
BGR	Bulgaria	FRA	France	KOR	South Korea	RSA	South Africa
BRA	Brazil	GER	Germany	KSA	Saudi Arabia	SCO	Scotland
CHI	Chile	HOL	Netherlands	MEX	Mexico	TUN	Tunisia
CMR	Cameroon	HRV	Croatia	MOR	Morocco	USA	United States
COL	Columbia	IRN	Iran	NGA	Nigeria	YUG	Yugoslavia

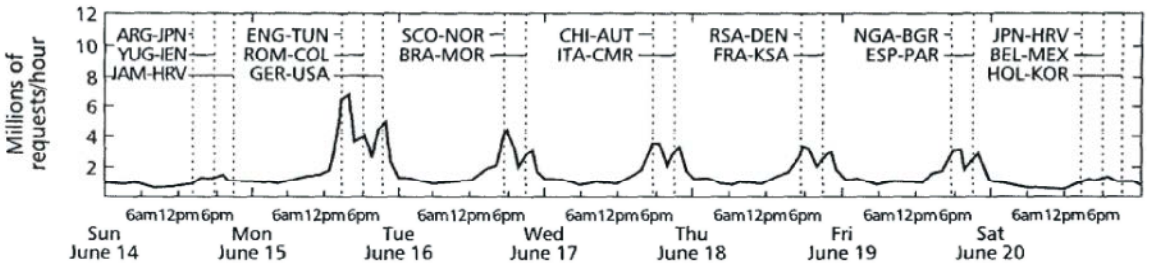

Fig. 1: Workload Characteristics of 1998 World Cup Web Site

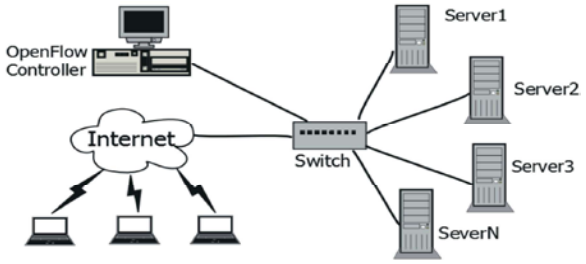

Fig. 2: Proposed Architecture of SDN-based web clusters

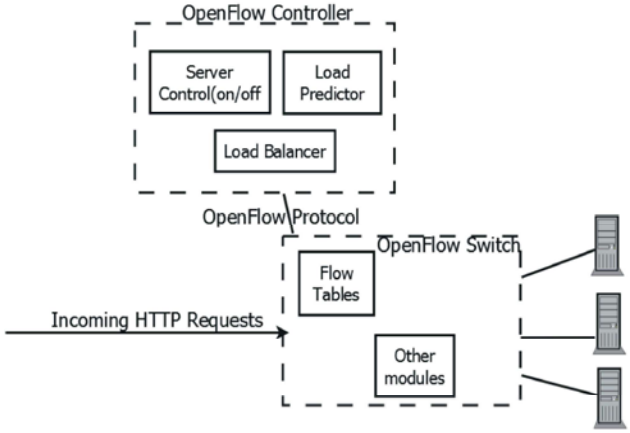

Fig. 3: High levels Diagram of OpenFlow Controller and Switch

and a set of web servers. An OpenFlow switch is a packet forwarding node which forwards network packets confirming to the rules specified in its flow table. Each row of flow table is called a flow entry or flow rule, each flow entry contains match fields, counters and instructions. These flow entries are added, modified and removed by the OpenFlow controller.

The OpenFlow based switch is the front end node of the SDN based cluster of web servers which receives requests and forwards them to the right web servers according the flow table populated by a OpenFlow based controller and if the incoming request does not match any flow entry in the flow table of the OpenFlow switch it sends the packet to the OpenFlow controller.

In this proposed cluster architecture, the requests are routed by routers to the cluster through forwarding the requests to the centralized web-switch. On receiving of a new web service request by the OpenFlow controller from the OpenFlow switch, controller consults the Load Predictor and eventually sends the flow entry to the OpenFlow switch to forward that request to a chosen web server according to its load balancing policy [32-40].

The Functional Description of Openflow Controller: Figure 3. Shows the proposed high level diagram of Open Flow based controller with its main components from the context of load balancing and also the OpenFlow switch with its main components. The main components of Open Flow controller in our context are load predictor, server controller and load balancer. Load predictor predicts the load of the cluster using the flow counters provided by OpenFlow switch. Server Controller controls the servers based on the input from the Load predictor if the expected future load is too low it will shut down few servers and if the future load is expected high, it will restart the servers. Load balancer according to the load balancing strategy finds out the right web server to be used for the given request.

All the client requests to reach the web servers of cluster have to reach the OpenFlow switch using virtual IP address. The OpenFlow switch finds out the right web server to service the request by matching the incoming request parameters with the available flow entries of flow table in OpenFlow switch which are populated by the OpenFlow controller and forwards the request to the server of matching flow entry. If no flow entry matches the incoming request, then the OpenFlow switch sends that request to the OpenFlow controller. OpenFlow controller upon receiving the new request as a packet-in message, it consults the load predictor module and gives the inputs to the load balancer. Eventually load balancer module inside the OpenFlow controller finds out the right web server to service the request and this information is sent to the OpenFlow switch as a flow entry to add into its flow table. And OpenFlow switch receives this message and according this flow entry incoming requests are forwarded to the selected web server which is given in the flow entry.

PCCSDN: Server load balancing strategy with the prediction of cluster's capacity to reduce the energy footprint (PCCSDN) using SDN.

Theoretical background and implementation details of proposed PCCSDN, a server load balancing strategy with the prediction of cluster's capacity and reduction energy footprint using SDN networks is described in this section. The emergence of Software-Defined Networking gives efficient and affordable solution for network traffic management [20, 22]. The main feature of SDN is separation of the data plane from the control plane and it clearly enables addition of difficult software implementations of complicated networking applications on top of control plane. Though OpenFlow is a very much accepted protocol to monitor and configure OpenFlow switches in SDN, SDN is not just limited to OpenFlow only. There are other control plan separation techniques existed before OpenFlow. OpenFlow enabled switches connect to OpenFlow controllers. In addition to control and monitoring of the forwarding plane, the OpenFlow protocol is efficient of collecting per-flow statistics, a feature that is essential for predicting the capacity of cluster. Linear Predictive Coding (LPC) is well known to predict the speech signals [32]. As speech signal is a time-series, it may be possible to use LPC for other time-series with identical statistical properties. LPC is already used successfully to forecast stock market prices and lifetimes of process in networked workstations [33]. Attempts to apply LPC to predict network traffic [34, 40] are supportive. To predict the future service request load the following formula is used in LP Analysis:

$$\hat{y}(n)=\sum_{k=1}^{p} a(k)*y(n-k) \tag{2}$$

where the estimate of the current value of cluster capacity (service request load) is expressed by weighted sum of past values $\hat{y}(n)$. The coefficients a(k) are acquired by reducing the error in forecast e(n), where:

$$e(n)=y(n)-\hat{y}(n) \tag{1}$$

An OpenFlow software controller is currently used to install the rules in the forwarding network elements to act on the network traffic through the network and it enables the efficient traffic management instead of just using routing protocols and Access Control Lists (ACLs). Packets received at this switch are matched with the match field of flow entries. If the packet matches any flow entry, counters of flow entry are updated and the action specified in the flow entry is executed. Controller can

balance the traffic by sending the flow rules into the OpenFlow-enabled switches. The OpenFlow switch operates as an intelligent switch that forwards customer requests to a web server given in the matching flow rule. This intelligence or flow rules are provided by the OpenFlow Controller. The OpenFlow Controller receives the packet-in message from OpenFlow enable switch whenever a new request does not match any flow entry of flow table. As the new load balancing request is forwarded to OpenFlow controller and load balancer module decides the right web server to service the request. Eventually this decision is sent to OpenFlow enabled switch by adding a flow entry into its flow table. Load Predictor's inputs are used by Load Controller to shut down or restart the servers.

Performance Evaluation: Experimental test bed setup and experiments are described in this section. Experiments with varying workloads are run to find out the performance of the PCCSDN strategy. Initially we ran the tests in mininet [3] emulator as it provides environment to run the real code and also the development of the application is easier and faster. Our cluster of web servers consisted of 7 machines. Extreme Networks summit x440 24t is used as OpenFlow switch in our experiments. The hardware configuration of web servers is core 3 2.0 GHz CPUs with 4GB of DDR RAM. We used enough 2.8 GHz core 2 quad machines as the client emulators to ensure that they would not become bottlenecked in any of our experiments. As the web switch uses a unique virtual IP address for the clients reach ability, cluster's distributed architecture gets hidden from all the clients. Httperf [8] is a tool to find the performance measurement of web servers. It is used to generate the client workloads.

A number of issues still remain in using linear prediction coding for the analysis of time series. It is not really clear from the view point of signal processing what is the optimal choice of parameters for poling interval and look forward time. If the time series are non-stationary or quasi stationary, short frame size can be chosen. But if the time series are stationary, large frame sizes can be chosen. Too large a frame size will not indicate the varying characteristics of the system. While a too short a frame will not give right estimates of the predictor coefficients. Different experiments were performed to study the behavior of the system by changing the polling interval, frame size and look forward time. We tested the behavior of the system in each case in terms of estimation error. In server load balancing, our main goal is to be able to predict the requirement of the data center ahead of time so that suitable action can be taken, if the number of requests (capacity) of the system is likely to be coming down or going high. Look forward time of ~ 10 minutes is more than sufficient for an automated process to take suitable action in the context of server load balancing. Figure 4 shows the error percentage for different polling intervals. We can observe that error percentage is decreased as the polling interval is increased. For small polling intervals, the performance of the predictor is likely to be influenced by the clear structure of time series which indicates service request traffic, when the polling interval is bigger the performance of the predictor is influenced by the wrapper of time series which indicate service request traffic. As our aim is to predict the future service request size (capacity) for look ahead times of ~ 10 minutes in the context of load balancing of servers in data centers, hence a bigger polling interval can be taken. We observe that a polling interval of 30 seconds is sufficient from the experiments we conducted.

Figure 5 depicts the error percentage for different look ahead times. For this experiment polling interval of 30 seconds is used to find out the performance of the load balancing system for different look forward times. From Figure 5 we can find that the error percentage increases that is performance of load balancing system decreases as the look ahead time increases.

Related Work: Handigol *et al*. [35] presented a network load balancing proposal called LOBUS. This algorithm tries to solve two problems of load balancing: server load and network congestion. Plug-n-serve is the name given to the load balancing system of LOBUS algorithm, it is developed by Stanford. LOBUS is a OpenFlow based solution that uses NOX controller. Plug-n-serve uses a greedy selection solution to select the (server, path) pair which gives the minimum response time for every request. LOBUS lets the administrator enhance the size of web services by adding server computing resources. LOBUS used same IP alias for all the servers in the demo. As the request arrives with the destination IP address of servers at the switch it gets forwarded to the controller and LOBUS finds out the right server and route which minimizes the response time. To arrive at the right server LOBUS does the following: it monitors the status of servers and network topology and congestion.

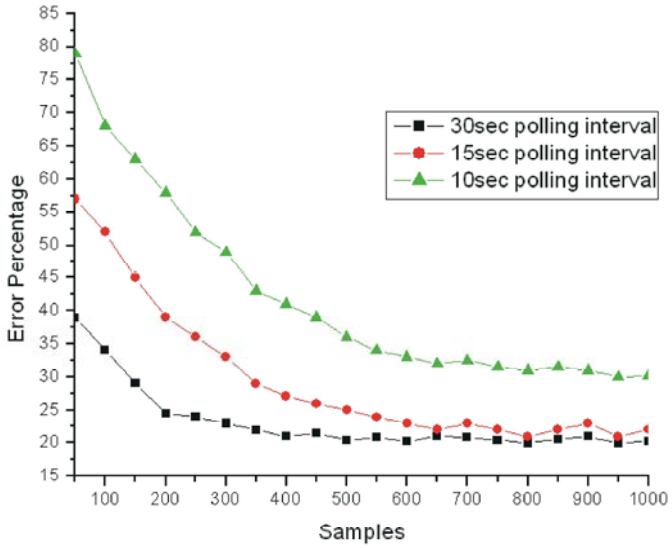

Fig. 4: Error Percentage for different polling intervals

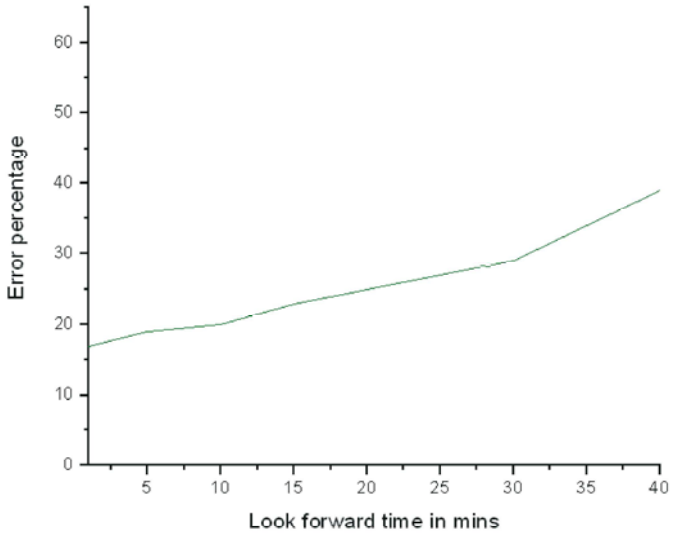

Fig. 5: Error percentages for different look forward times

Uppal and Brandon [36] developed an alternative load balancing strategy based on OpenFlow protocol. The basic architecture of this load balancing strategy contains an OpenFlow switch connected to an NOX OpenFlow controller and a group of servers connected to OpenFlow switch. It has the ability to be more resistant to failures with future versions of OpenFlow switches. NOX OpenFlow controller maintains the list of servers and each server uses a static IP address. NOX controller defines the load balancing strategy and adds a flow entry into the flow table of the OpenFlow switch. Three server load balancing policies namely: Round Robin, Random and Least Loaded were compared. Clients send the requests with destination as the IP address of VIP, as the request is received by OpenFlow switch it sends the request to NOX controller; NOX informs the switch target IP address and target physical address of the packet according to the load balancing policy. Upon receiving the flow entry from NOX controller, OpenFlow switch writes it into flow table and changes the destination MAC address and destination IP address and forwards the packet to the selected server. The server sends the reply message to OpenFlow switch, OpenFlow switch changes the source IP address and source Physical address to the destination IP address and destination Physical address of the client request.

Long *et al*. presented a new load balancing algorithm called LABERIO to distribute the network traffic dynamically [37]. By utilizing resources better LABERIO reduces the network latency and transmission time and enhances the throughput. The results of their experiments demonstrate that LABERIO can get better response time compared to previous load balancing solutions. These proposals do not consider the health state of the server in distributing the traffic among several servers. In this work, we describe the enhanced server health monitoring used in load balancing for OpenFlow based networks. L. Yu and D. Pan presented a load balancing algorithm which does dynamic routing to get better performance and lesser latency in the data center [38]. This proposal describes a load balancing solution for the fat-tree network architecture with support of multipath. This proposed load balancing system is extended with dynamic routing algorithm. This algorithm is flexible to scheduled flows and network traffic by checking the available bandwidth of the remaining trunks. R. Syrotiuk and A. Ghaffarinejad [39] presented a load balancing system to be used in their campus network which is SDN enabled. They present that their solution can reduce costs and increase network adaptability. In this work, we propose prediction of cluster capacity in OpenFlow enabled networks which is not available in the literature of data centers using software defined networks.

CONCLUSION

In this paper, a novel energy-efficient load balancing strategy PCCSDN which predicts the capacity of the cluster and reduces the energy footprint (PCCSDN) in Software Defined Networks is proposed. This proposed policy, PCCSDN policy is capable of switching off the extra servers which are not required during night time, weekend and holidays thereby reduces the energy footprint of the data center and continuously predicts the service usage to take care of unexpected traffic. We show that using PCCSDN strategy, the energy footprint of the data center is reduced by 37.9%. This is achieved by taking the near real perspective of the service requests using the measurement of flow counters at that moment and predicts the future service requests up to ~ 10 minutes. OpenFlow provided counters: packet count and byte counts are used for predicting the size of the cluster. The core selection & control process of PCCSDN policy can be enhanced with the help of real time status information of transactions and also using Business Activity Monitoring for the improvement of load balancing efficiency.

REFERENCES

1. Aron, M., P. Druschel and W. Zwaenepoel, 2000. Cluster reserves: a mechanism for resource management in cluster-based network servers. In: Proc. of ACM SIGMETRICS.
2. Boone, B., S. Van Hoecke, G. Van Seghbroeck, N. Joncheere, V. Jonckers, F. De Turck, C. Develder and B. Dhoedt, 2010. Salsa: QoS-aware load balancing for autonomous service brokering, Systems Software., 83(3): 446-456.
3. Lantz, B., B. Heller and N. McKeown, 2010. A Network in a Laptop: Rapid Prototyping for Software-Definded Networks, ACM SIGCOMM.
4. Cardellini, V., E. Casalicchio, M. Colajanni and P.S. Yu, 2002. The state of the art in locally distributed web-server systems. ACM Comput Surv (CSUR), 31: 263-311.
5. Casalicchio, E., V. Cardellini and M. Colajanni, 2002. Client-aware dispatching algorithms for cluster-based web servers. Clust Comput, 5(1): 65-74.
6. Cardellini, V., E. Casalicchio, M. Colajanni and M. Mambelli, 2001. Web Switch Support for Differentiated Services. ACM SIGMETRICS Perform Eval. Rev. (29).
7. Carter, R.L. and M. Crovella, 1995. Dynamic Server selection in the Internet, Tech. Rep. TR-95-014, Computer science Department, Boston University, Boston, MA.
8. Mosberger, D. and T. Jin, 1997. Httperf: A Tool to Measure Web Server Performance, Proc. USENIX Symp. Internet Technologies and Systems, pp: 59-76.
9. Andrews, G.R., D.P. Dobkin and P.J. Downey, 1982. Distributed allocation with pools of servers, in ACM SIGACT-SIGOPS Symp. Principles of Distributed Computing, Aug., pp: 73-83.
10. Long, H., Y. Shen, M. Guo and F. Tang, 2013. LABERIO: dynamic load-balanced routing in OpenFlow-enabled networks,in Proceedings of the 27th IEEE International Conference on Advanced Information Networking and Applications (AINA '13), IEEE, pp: 290-297.

11. Gilly Katja, Carlos Juiz and Ramon Puigjaner, 2011. An up-to-date survey in web load balancing, World Wide Web, 14(2): 105-131.
12. Saifullah, M.A. and M.A. Maluk Mohamed, 2014. Server load balancing through enhanced server health report, International Journal of Applied Engineering Research (IJAER), ISSN 0973-4562, 9(24): 28497-28516.
13. Saifullah, M.A. and M.A. Maluk Mohamed, 2014. Scalable load balancing using virtualization based on approximation, IEEE International Conference on Computer Communication Technology (ICCCT), Dec 11-13, Hyderabad.
14. Andreolini, M. and E. Casalicchio, 2004. A Cluster-Based Web System Providing Differentiated and Guaranteed Services, Cluster Computing, 7: 7-19.
15. Aron Mohit, 2000. Differentiated and Predictable Quality of Service in Web Server Systems. PhD thesis, Department of Computer Science, Rice University.
16. Abdelzaher Tarek F., Kang G. Shin and Nina Bhatti, 2001. Performance Guarantees for Web Server End-Systems: A Control-Theoretical Approach, IEEE Transactions on Parallel and Distributed Systems.
17. Saifullah, M.A. and M.A. Maluk Mohamed, 2015. Scalable load balancing using enhanced server health monitoring and admission control, IEEE International Conference on Engineering and Technology (ICETECH), March 20th, 2015 Coimbatore, TN, India.
18. Saifullah, M.A. and M.A. Maluk Mohamed, 2015. Efficient server load balancing through improved server health report, ARPN Journal of Engineering and Applied Sciences, ISSN 1819-6608, 10(10): 4706-4716.
19. Saifullah, M.A. and M.A. Maluk Mohamed, 2015. OpenFlow based load balancing using enhanced server health reports, Submitted to IEEE International Conference on Power, Control, Communication and Computational Technologies for Sustainable Growth (PCCCTSG), 2015 Kurnool andhra Pradesh, India.
20. McKeown, N., T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner, 2008. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2): 69-74.
21. Jayabal, R. and R. Mohan Raj, 2014. Design and Implementation of Locally Distributed Web Server Systems using Load Balancer, International Journal of Engineering Sciences & Research Technology, ISSN: pp: 2277-9655.
22. Wang, R., D. Butnariu and J. Rexford, 2011. Open Flow-Based Server Load Balancing Gone Wild, in Workshop on Hot-ICE, Mar. 2011.
23. Sharifian Saeed, Seyed A. Motamedi and Mohammad K. Akbari, 2010. An approximation-based load-balancing algorithm with admission control for cluster web servers with dynamic workloads, The Journal of Supercomputing, 53(3): 440-463.
24. Andreolini, M. and S. Casolari, 2006. Load prediction models in web-based systems, in Proceedings of the 1st international conference on Performance evaluation methodolgies and tools, ACM: Pisa, Italy, pp: 27.
25. Dinda, P.A., 2006. Design, Implementation and Performance of an Extensible Toolkit for Resource Prediction in Distributed Systems, IEEE Transactions on Parallel and Distributed Systems, 17: 160-173.
26. Wenbo Chen, Zhihao Shang, Xinning Tian and Hui Li, 2014. Dynamic Server Cluster Load Balancing in Virtualization Environment with OpenFlow, International Journal of Distributed Sensor Networks, Article ID 531538.
27. Rood, B. and M.J. Lewis, 2010. Availability Prediction Based Replication Strategies for Grid Environments, In 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. Melbourne, VIC, Australia.
28. Huang, J., 2008. Short-Term Traffic Flow Forecasting Based on Wavelet Network Model Combined with PSO. In International Conference on Intelligent Computation Technology and Automation.
29. Liu, J., R. Zhang and L. Wang, 2010. Prediction of Urban Short-Term Water Consumption in Zhengzhou City. In International Conference on Intelligent Computation Technology and Automation, Changsha, Hunan, China.
30. Arlitt, M. and H. Jin, 1999. Workload Characterization of the 1998 World Cup Web Site, in HPL-1999-35 (R.1), HP Laboratories Palo Alto, 90.
31. Vimal Mathew, Ramesh K. Sitaraman and Prashant Shenoy, 2012. Energy-Aware Load Balancing in Content Delivery Networks 31st Annual IEEE International Conference on Computer Communications (IEEE INFOCOM). March 2012.
32. Makhoul, J., Linear prediction: A tutorial review Proceedings of the IEEE, 63(4): 561-580.
33. Mahamuni Atul B., Timothy A. Gonsalves and Bhaskar Ramamurthi, 1993.Efficient Load Information Management for Load Sharing in Distributed Systems, Computer Networks Architecture and Applications, (C-13).

34. Thottan, M. and C. Ji, 1998. Proactive anomaly detection using distributed intelligent agents, IEEE Network, 12: 21-27.
35. Handigol, N., S. Seetharaman, M. Flajslik, N. McKeown and R. Johari, 2009. Plug-n-serve: load-balancing web traffic using OpenFlow, in Proceedings of the ACM SIGCOMM Conference on Data Communication (SIGCOMM'09), Barcelona, Spain.
36. Uppal, H. and D. Brandon, 2010. OpenFlow based load balancing, Project Report CSE561: Networking, University of Washington.
37. Long, H., Y. Shen, M. Guo and F. Tang, 2013. LABERIO: dynamic load-balanced routing in OpenFlow-enabled networks, in Proceedings of the 27th IEEE International Conference on Advanced Information Networking and Applications (AINA '13), IEEE, March 2013, 290-297.
38. Yu, L. and D. Pan, 2013. OpenFlow based load balancing for fat-tree networks with multipath support, in Proceedings of the 12th IEEE International Conference on Communications (ICC '13), Budapest, Hungary.
39. Ghaffarinejad, A. and V.R. Syrotiuk, 2014. Load balancing in a campus network using software defined networking,in Proceedings of the 3rd GENI Research and Educational Experiment Workshop (GREE '14), IEEE.
40. Ramasamy, A., Hema A. Murthy and T.A. Gonsalves, 2000. Linear Prediction for Traffic Management and Fault Detection, Proc. Int'l. Conf. Info. Tech., Bhuvaneshwar, 4.