Round-robin Algorithm in HAProxy and Nginx Load Balancing Performance Evaluation: a Review

Luthfan Hadi Pramono
dept. of Computer Engineering
STMIK AKAKOM
Yogyakarta, Indonesia
luthfanhp@akakom.ac.id

Robby Cokro Buwono
dept. of Informatic Engineering
STMIK AKAKOM
Yogyakarta, Indonesia
robbycokro@akakom.ac.id

Yanuar Galih Waskito
dept. of Informatic Engineering
STMIK AKAKOM
Yogyakarta, Indonesia
yg.waskito@gmail.com

***Abstract*—The web application is expected to be continuously available, to meet these objectives the web application must be supported by a reliable, dynamic and flexible web server. This is possible if the system utilizes more than one web server to serve requests from users. Load Balancing makes web application performance better when more than one server is used. This can improve the efficiency of the results created by the server, and reduce the traffic load from a single server and distribute the traffic load to all servers connected to load balancer. In this study a load balancing performance review will be carried out on HAProxy and Nginx by implementing the Round-robin algorithm on both load balancing servers. The results of the final conclusions of this study are which load balancing server has better performance which can later be used as a descriptive and measurable reference for future studies.**

***Keywords*—load balancing, Round-robin, HAProxy, Nginx, web server**

I. INTRODUCTION

Today's web servers are very vital, with the current technological trend where many native applications are starting to migrate into web-based applications based on cloud computing services. Web applications must be supported by a reliable and robust web server. The use of a web server on a traditional system has become very limited, where usually a web application system only runs on one web server and the web server runs on the hardware directly. So that it can be concluded, one web application is in one web server and runs on one hardware. As we know, a single web server will have limited access load, so if the access load has exceeded the limit then the web server will not be able to access and a notification will appear 503 Service Unavailable http error code. For a single web server like this there are usually several techniques that are used as a solution to solve the problem of request overload, including by adding hardware resources such as RAM, disk, nic, or others. The second is tuning OS parameters such as hardware capabilities and usage. The third, using a more reliable web server application such as nginx. Because the user growth of a web application is greatly increased at this time, the single web server is very unreliable and irrelevant to implement, while the web application is currently demanded to be always available and has zero downtime operational. The reliability of web applications is still a major problem in web applications and has become a lot of attention from most users. Fault tolerance computing means that a job can continue and improve to carry out all work when the hardware and or software fail [1].

In order for web applications to continue to be available, web applications must be supported by a reliable, dynamic and flexible web server. This is possible if the system utilizes more than one web server to serve requests from users. Some of these web servers will serve the load of requests together which are managed by a load balancer that divides the load on each web server worker. Load Balancing makes web application performance better when more than one server is used. This can improve the efficiency of the results created by the server, reduce the traffic load from a single server and distribute the traffic load to all servers connected to load balancers. Load balancer can also provide solutions if a system crashes, because there are many servers that work to service requests. If there is one server that has failed, the system can still run using a server that still exists [2].

In this study will be discussed about performance evaluation review of load balancing servers from HAProxy and Nginx, by implementing the Round Robin algorithm for each load balancing server. Finally, as the conclusion of this study is the load balancing server that has the best performance from the results of testing and evaluation.

The paper is organized as follows. Section II presents related work. In Section III we describes the round-robin algorithm. Performance evaluation are reported in Section IV. Finally, the closing remarks describe in Section V.

II. RELATED WORK

Research [3] shows how load balancing works and distributes loads using the Round Robin approach. From the study, it was proved that load balancers that employ multiple servers provide more efficient results in terms of performance and accuracy compared to single servers, the traffic load normally received by a single server is also reduced because it has been distributed to all servers connected to load balancers.

Research that has been done [4], uses HAProxy load balancer with heterogeneous web cluster. The algorithms used to evaluate are Round Robin, Static Round Robin and Least Connection. This research was made to evaluate load balancing performance. On the one hand, the study discusses several load balancing techniques in heterogeneous clusters. On the other hand, analyzing load balancing systems in homogeneous and heterogeneous web clusters, it is possible to understand the problems that occur.

One of Nginx performance examination is the result of comparing the performance of the Apache web server with the Nginx and Varnish reverse proxy caching obtained from tests performed by [5]. The examination in this study was carried out in 2 scenarios, namely resource management testing and then continued with quality service testing (QoS) from the use of Nginx and Varnish on the Apache web server. In both tests, the load assigned to each server is divided into 3 load categories, which are full traffic, half traffic, and quarter traffic. The number of requests for each load category in sequence are 2500, 1250, and 625 requests per minute. The request is made only for one web page so the server will repeat the requested web page.

Various load balancing techniques are analyzed in different cloud computing environments (homogeneours & heterogeneous) [6], then the performance of the system is evaluated using several parameters. The results of this study are the performance comparison between load balancing algorithms and the cloud computing environment used.

To deal with the problem of job allocation from ill-balanced, long response times, low output averages and poor performance when the system cluster handles work, [7] Introduces the concept of entropy in thermodynamics to the load balancing algorithm and propose a new load balancing algorithms for homogeneous cluster based on the maximum entropy method. By calculating the entropy of the system and using the entropy maximism principle to ensure that each scheduling and migration is carried out in accordance with the tendency of entropy, the system will be able to get the status of load balancing as soon as possible, shorten execution time and high performance.

The research carried out [8] introduced a new class of product-form queueing networks that allows dynamic load balancing algorithm modeling initiated by the recipient. The theorem put forward shares with the results of other literary problems which assumed the existence of non-linear level equation system solutions without providing an explicit mechanism for their calculations. The research directs this problem to networks with open topologies and provides algorithms to calculate pooling averages between network stations that ensure both networks become in product form and the set of stations specified has load balance.

Cluster-based multimedia web server has been done in research [9] which is used dynamically to produce video units to meet the varying bit rate and bandwidth requirements of the client. The media server divides the task into several jobs and schedules it on the backend computing node for processing. For stream-based applications, the main design criteria for scheduling are to reduce total processing time and maintain the order of the media units for outgoing streams. This research has been designed, implemented and evaluated three scheduling algorithms, First Fit (FF), Stream-based Mapping (SM), and Adaptive Load Sharing (ALS), for multimedia transcoding in a cluster environment. Predictions regarding CPU load are also considered for each multimedia job and schedule it according to the diversity of individual work. Therefore, this study also proposes two load scheduling algorithms namely Prediction-based Least Load First (P-LLF) and Prediction-based Adaptive Partitioning (P-AP), which can be used to predict and improve performance. The performance of the system is evaluated by the provisions of the system output, the average out-of-order output of media streams, and the load balancing overhead through tangible measurements using a cluster computer.

The use of resources and energy consumption can be less of a concern in research in Load Balancing or lack of discussion about it. To improve resource use and energy consumption, [10] in its research propose the DVFS improved scheme, as well as conducting comparative studies between static and dynamic load balancing algorithms.

In research done by [11], a new load balancing solution was made about resource use, ease of provisioning needs and helped to tolerate sudden rising loads. The proposed solution is to use a protocol for remote resource reservation. This protocol prevents overloading from the remote server by limiting the amount of load from each server that can lead to another server.

The survey conducted by [12] is about flat and hierarchical load balancing strategies proposed by researchers for computational grid environments. In this survey, comparisons were made based on certain feature and performance metrics such as response time, throughput, communication delay, resource utilization, communication overhead, fault tolerance, scalability, and the type of implementation of activities.

A survey of load balancing technique has been carried out by [13] in its use for grid computing. In the survey, several techniques were then grouped into a number of them namely, tree based approach, estimation based, optimization based, agent based, artificial life technique, hybrid base, neighbor based, and partitioning based. Algorithms, research focus, contributions, features, compared models, performance metrics, improvements, gaps and future work are presented in the survey.

Statistics based on article studies were made by [14] through a survey of load balancing algorithms in cloud computing. From the statistics that have been made, data obtained 22% study technique leads to response time metric, 24% leads to makespan, 19% leads to resource utilization metric, 9% leads to throughput,% 9 leads to energy saving, 9% leads to scalability , 8% leads to migration time metric.

III. LOAD BALANCING ALGORITHM

Based on the state of the system Load balancing algorithm is classified into two, namely [15]:

- Static load balancing algorithm: is a load balancing algorithm that does not depend on the current state of the system. At the beginning, before the request enter to the server, it will be determined on which server the request will be executed.
- Dynamic load balancing algorithm: in this algorithm load balancer will analyze the load statistics of each server in the current state and execute the request on the right server.

Examples of static algorithms are Min-min, Roud-robin, Opportunistic Load Balancing (OLB), and Max-min. Included in the category of dynamic algorithms are Honey Bee Foraging, Ant Colony Optimization and Throttled [10].

In this research, the algorithm used is Round-robin. For comparison to other algorithms, Leastconn algorithm will also be implemented. Considering that the Round-robin and Leastconn algorithm are used in both load balancing server of HAProxy and Nginx as well.

Round-robin (RR) planning calculations are arranged for time-sharing systems. This calculation is similar to FCFS, but the pre-emtion is further to switch between other procedures. The Round-robin load balancing algorithm is used as a content switching algorithm, it treats all servers equally, similar to rotary DNS but without propagation delay or caching problems. Each load is distributed equally on each server, regardless of the current number of connections or response time. Round-obin is very suitable to be used when server workers in load balancing systems have the same process specification capabilities, if not, then some servers will receive excess requests exceeding their process capabilities while others only use some of their resources.

If there are *n* processes in the queue that are ready and the time slice is *q*, then each process will ideally get *1/n* of the CPU time in the time unit *q*, and each process will wait no more than the unit of time *nq* until the next quantum. Then the time unit formula obtained as in (1).

$$TU = n(q+o) \qquad (1)$$

Where *o* is an overhead context switch. The performance of the Round-robin algorithm depends on the size of the quantum (time). Quantum size is the time needed to complete a job. If quantum time is very large, the Round-robin algorithm will be similar to the First-come First-served algorithm. If quantum time is very small, the Round Robin approach is called a Processor sharing [15].

IV. Performance Evaluation

The scenario of performance evaluation in this research is creating an identical load balancer network design between HAProxy and Nginx to measure the performance comparison of Load Balancer shown in Fig. 1.

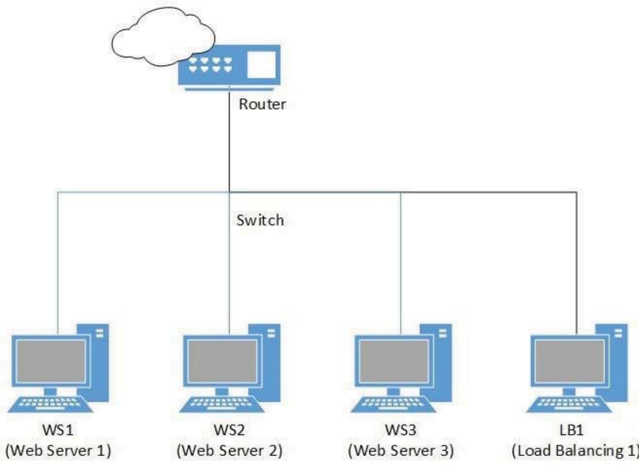

Fig. 1. Network design

Some test scenarios will be implemented in the evaluation of this reaserch. HAProxy already uses Keep Alive in the program code, while Nginx uses a separate configuration and optionally which Keep Alive (KA) feature is not activated by default. As of in this research will also compare the Nginx test using the Keep Alive configuration as a consideration of the test results. Each server load balancing will be tested with the same variable value. In this research another algorithm will be used as a consideration and evaluation comparison, namely Leastconn algorithm. The Leastconn algorithm is a non-round-robin algorithm that is used in load balancing server both HAProxy and Nginx.

Several examination are performed to evaluate load balancing performance of HAProxy and Nginx:

A. Load and stress test

By using Apache-Jmeter-4.0, each server load balancing will be tested with several variables, Number of Thread (users) with a value of 500, is the number of requests or users who access the website, Ramp-Up Period (in seconds) with value 10, is how long it takes to increase the overall number of selected threads, and the Loop Count with a value of 1, indicating how many repetitions occur for each user. The results obtained are shown in table I and table II.

TABLE I. Comparison Results of Load Balancing Server Tests With The Ramp-up Period Using Round-robin

LB Server	**Throughput (request/minutes)**			
	Test 1	***Test 2***	***Test 3***	***Average***
HAProxy	2.959	2.965	2.912	2.945
Nginx	2.959	2.973	2.973	2.968
Nginx KA	3.059	2.962	2.962	2.994

TABLE II. Comparison Results of Load Balancing Server Tests With The Ramp-up Period Using Round-robin compared to Leastconn Algorithm

LB Server	**Throughput (request/minutes)**	
	Round-robin	***Leastconn***
HAProxy	2.945	1.978
Nginx	2.968	34.561
Nginx KA	2.994	45.256

Comparison of load balancing performance using Round-robin and Leastconn algorithms looks very significant. The resulting output when using the Leastconn algorithm reaches tens values. From the test using Round-robin and Leastconn algorithms the results obtained have in common, where the highest load balancing server performance is on Nginx with KA, followed by Nginx and the lowest is HAProxy.

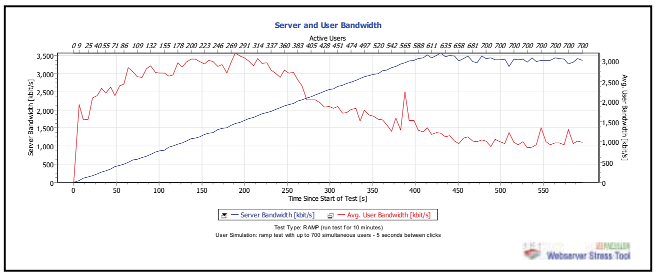

Fig. 2. Server and user bandwidth graph, load balancing server HAProxy

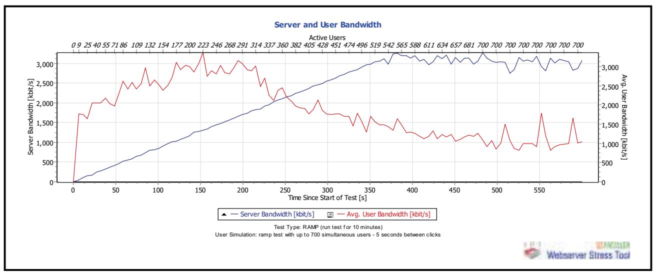

Fig. 3. Server and user bandwidth graph, load balancing server Nginx

Webstresstool also used as RAMP testing with an increased load scenario for the specified time. Followed by an increase in the load of 1 user to the number of users determined to reach 80% of the testing time. During the last 20%, the number of full users will be active. The value given to the Number of Users variable is 700, which is the number of users accessing the web server. The value given in the Run Test For variable is the duration of the time for which this test will run for 10 minutes. While the value given to the variable Click delay is 5 seconds.

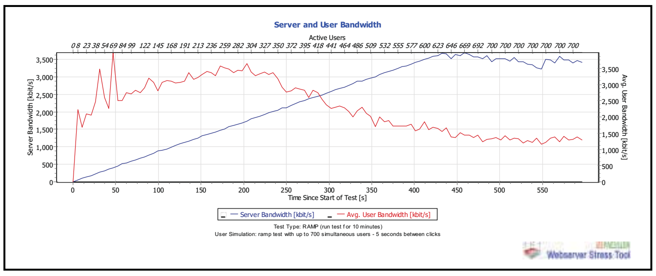

Fig. 4. Server and user bandwidth graph, load balancing server Nginx with Keep Alive

Using Round-robin algorithm, from the test results obtained server statistics and user bandwidth shown in Figure 2 to Figure 4. Graph server and user bandwidth, load balancing of Nginx servers with KA in fig. 4 shows quite low traffic compared to other load balancing servers where server bandwidth reaches peak in the number of user requests above 623. The second rank is load balancing server HAProxy on fig. 2, where server bandwidth reaches the peak of the number of user requests above 588. While load balancing server Nginx standard fig. 3, server bandwidth reaches the peak on the number of user requests above the number 542. This can be concluded that load balancing of the Nginx server with KA is able to handle user requests quickly than others.

B. *Time test*

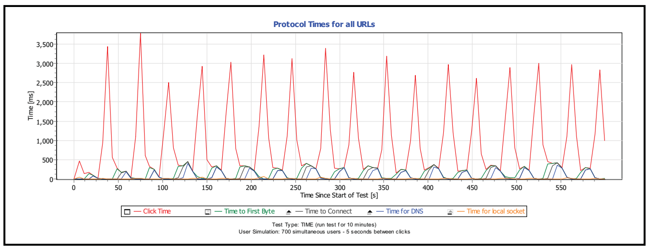

Fig. 5. Protocol Times for all URLs graph, load balancing server HAProxy

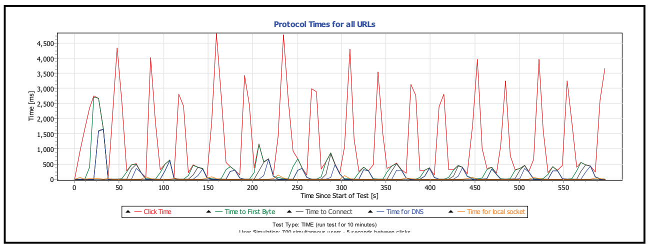

Fig. 6. Protocol Times for all URLs graph, load balancing server Nginx

The test will run for a certain number of minutes. The algorithm used is Round-robin. Timed testing is often used for "burn in tests". The value given to the Number of Users variable is 700, which is the number of users accessing the web server. The value given in the Run Test For variable is the duration of time used for how long this test will run for 10 minutes. While the value given to the variable Click delay is 5 seconds.

Fig. 5 through 7 show a graph of Protocol Times for all URLs. From the server load balancing that is compared, we get data that the smallest value from Time to First Byte, Time to Connect, Time for DNS and Time for local socket is owned by the Nginx server load balancing with KA which is fig. 7, this proves that Nginx with KA has response time is very small so it can more quickly handle requests from users. The rating below is load balancing server HAProxy shown in fig. 5 and the last is load balancing server Nginx Standard shown in fig. 6.

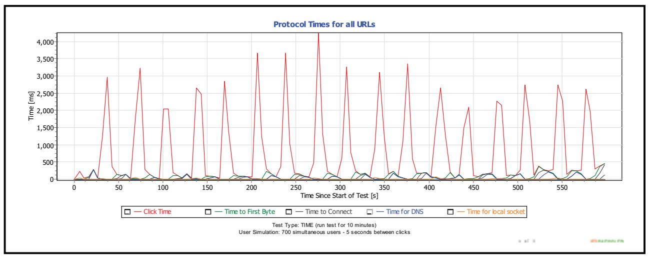

Fig. 7. Protocol Times for all URLs graph, load balancing server Nginx with KA configuration

C. *Click test*

Constant load testing until each user produces the specified number of clicks. The test will be completed when each user has started the number of clicks given. The value for the test given to the Run Until variable is 1, so that each user will click once. The value given to the Number of Users variable is 500, which is the number of users accessing the web server. And the value given to the Click delay variable is 5 seconds. For the test results with the click scenario shown in table III. This test will show information about the request time from the web page. If the request time value is small, it can be concluded that the web has a good performance.

TABLE III. LOAD BALANCING SERVER TEST RESULTS USING A CLICK TEST SCENARIO USING ROUND-ROBIN ALGORITHM AND COMPARED TO LEASTCONN ALGORITHM

LB Server	**Clicks (Time required [ms])**	
	Round-robin	***Leastconn***
HAProxy	19.454	808,939
Nginx	101.893	382,101
Nginx KA	30.739	401,184

From the test results using the CLICK scenario shown in table III shows that load balancing with round-robin algorithm in HAProxy server has a very fast time in responding to requests from a number of users as many as 500, with the maximum time needed to complete the load that is 19.454 ms. From this scenario we can conclude that the load balancing of the HAProxy server is very fast giving response requests. While the slowest in handling requests is standard Nginx with a value of 101.893. Whilst Nginx with KA has a value of 30,736. When compared using the Leastconn algorithm, the results obtained are the opposite of load balancing using the round-robin algorithm, where the smallest value is obtained from Nginx, then followed by

Nginx with KA and HAProxy being the last. The time value needed when using the leastconn algorithm is large enough if we use the CLICK test scenario.

D. Benchmark test

By performing benchmark test, we will know the capabilities of the web server, how many request per second the web server is capable of serving. This test uses apache benchmark (ab), which utilizes several attributes as shown in table IV. Each testing procedure will be carried out three times in each scenario as shown in table V. Other tests were also carried out using the Leastconn algorithm for comparison, as shown in table VI.

TABLE IV. ATTRIBUTES USED IN APACHE BENCHMARKS (AB)

Attribute	**Value**	**Description**
-c	10	Level of concurrency, number of hits in one request (client)
-n	500	Number of requests (client)
-k	-	Use the HTTP keep alive feature, to run multiple requests in one HTTP session

After performing various server load balancing scenarios and getting results with an average, then it will be evaluated and compared as shown in table V and table VI.

TABLE V. AVERAGE EVALUATION RESULTS OF LOAD BALANCING SERVER PERFORMANCE USING ROUND-ROBIN ALGORITHM

LBS Evaluation	**Throughput**		
	HAProxy	***Nginx***	***Nginx KA***
Time taken for tests [sec]	0.682	0.812	0.563
Requests per second [#/sec]	733.65	615.93	902.61
Time per request [ms]	13.631	16.236	11.264
Transfer rate [Kbytes/sec]	2743.33	2303.13	3375.1

TABLE VI. AVERAGE EVALUATION RESULTS OF LOAD BALANCING SERVER PERFORMANCE USING LEASTCONN ALGORITHM

LBS Evaluation	**Throughput**		
	HAProxy	***Nginx***	***Nginx KA***
Time taken for tests [sec]	12.409	12.159	8.41
Requests per second [#/sec]	40.29	41.127	59.453
Time per request [ms]	248.181	243.176	168.205
Transfer rate [Kbytes/sec]	150.667	153.773	222.31

Average evaluation results of Load Balancing Server using Apache Benchmark (ab) in table V shows the load balancing value of HAProxy hava smaller time than Nginx. Whilst if the KA (Keep Alive) configuration on Nginx is activated, it indicating the smallest time taken for tests than others, this indicates that Nginx with KA can quickly complete the request load from the user/client, for the second order is obtained from HAProxy and the last is from standard Nginx. The highest value of the persecond request shows an indication that in every second the load balancing server will handle many requests, the highest value is obtained from Nginx with KA followed by the standard HAProxy and Nginx. The smallest time per request indicates the speed of the server load balancing in completing the request, obtained the smallest number of Nginx with KA, followed by HAProxy and Nginx. In the evaluation of the transfer rate of each server load balancing, the largest value indicates the flow of data traffic in handling the increasing load requests, the largest value obtained from Nginx with KA, followed by HAProxy and Nginx.

V. CLOSING REMARKS

By default HAProxy already uses Keep Alive in the program code, while Nginx uses a separate configuration and optionally which Keep Alive feature is not activated by default. But when the Keep Alive feature is enabled on Nginx, it shows a very significant performance improvement that exceeds the performance of HAProxy. From the research and experiments that have been carried out, by implementing the Round-robin algorithm on load balancing evaluations of HAProxy and Nginx servers, the conclusion is that HAProxy has a good performance among Nginx in default configuration. If the Keep Alive configuration on Nginx is activated, then Nginx has very good performance and faster than the others. As an alternative comparison evaluation test, the Leastconn algorithm produces value characteristics similar to the Round-robin algorithm. The difference in test results is only found in the click test, where the results obtained from testing using the Leastconn algorithm have the opposite value of the results obtained from the test using the Round-robin algorithm.

As a suggestion for further research, it can be tested using another load balancing algorithm as a comparison of the performance evaluation of server load balancing. Other environments can also be implemented for evaluation, such as cloud infrastructure.

ACKNOWLEDGMENT

Thank you to STMIK AKAKOM for providing research and publication funds to the research team.

REFERENCES

[1] Youssef M Essa., A Survey of Cloud Computing Fault Tolerance: Techniques and Implementation, International Journal of Computer Applications, vol. 138 (13), Foundation of Computer Science (FCS), NY, USA, March 2016, pp. 34-38.

[2] F. Alam, V. Thayananthan and I. Katib, "Analysis of round-robin load-balancing algorithm with adaptive and predictive approaches," 2016 UKACC 11th International Conference on Control (CONTROL), Belfast, 2016, pp. 1-7.

[3] P. K. , Mohit Saxena, “A Round-Robin based Load balancing approach for Scalable demands and maximized Resource availability”, International Journal of Engineering and Computer Science, vol. 5, no. 8, May 2016.

[4] F. Mbarek and V. Mosorov, "Load balancing algorithms in heterogeneous web cluster," 2018 International Interdisciplinary PhD Workshop (IIPhDW), Swinoujście, 2018, pp. 205-208.

[5] Luthfi, Muhammad, Mahendra Data, & Widhi Yahya. "Perbandingan Performa Reverse Proxy Caching Nginx dan Varnish Pada Web Server Apache," Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer [Online], 2.4 (2018): 1457-1463. Web. 21 Agu. 2018.

[6] Sambit Kumar Mishra, Bibhudatta Sahoo, Priti Paramita Parida, Load Balancing in Cloud Computing: A big Picture, Journal of King Saud University - Computer and Information Sciences, 2018.

[7] L. Chen, K. Wu, and Y. Li, “A Load Balancing Algorithm Based on Maximum Entropy Methods in Homogeneous Clusters,” Entropy, vol. 16, no. 11, pp. 5677–5697, Oct. 2014.

[8] Andrea Marin, Simonetta Balsamo, Jean-Michel Fourneau, LB-networks: A model for dynamic load balancing in queueing networks, Performance Evaluation, vol. 115, 2017, pp 38-53.

[9] Jiani Guo and L. N. Bhuyan, "Load Balancing in a Cluster-Based Web Server for Multimedia Applications," in IEEE Transactions on Parallel and Distributed Systems, vol. 17, no. 11, pp. 1321-1334, Nov. 2006.

[10] R. Pushpa and M. Siddappa, "A comparative study on load-balancing algorithms/or cloud environments," 2017 3rd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), Tumkur, 2017, pp. 316-321.

[11] Nakai, A., Madeira, E. & Buzato, L.E., "On the Use of Resource Reservation for Web Services Load Balancing," Journal of Network and Systems Management, vol. 23(3), July 2015, pp 502–538.

[12] Sumair Khan, Babar Nazir, Iftikhar Ahmed Khan, Shahaboddin Shamshirband, Anthony T. Chronopoulos, Load balancing in grid computing: Taxonomy, trends and opportunities, Journal of Network and Computer Applications, vol. 88, 2017, pp 99-111.

[13] Deepak Kumar Patel, Devashree Tripathy, C.R. Tripathy, Survey of load balancing techniques for Grid, Journal of Network and Computer Applications, vol. 65, 2016, pp 103-119.

[14] Einollah Jafarnejad Ghomi, Amir Masoud Rahmani, Nooruldeen Nasih Qader, Load-balancing algorithms in cloud computing: A survey, Journal of Network and Computer Applications, vol. 88, 2017, pp 50-71.

[15] Dong Hyun Youm, Ravi Yadav, "Load Balancing Strategy using Round Robin Algorithm," Asia-pacific Journal of Convergent Research Interchange, vol.2, no.3, September 30 (2016), pp. 1-10.