TELKOMNIKA, Vol.16, No.3, June 2018, pp. 1402~1408
ISSN: 1693-6930, accredited **A** by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v16i3.5601

Weighted Round Robin Load Balancer to Enhance Web Server Cluster in OpenFlow Networks

Yuggo Afrianto[1], Heru Sukoco*[2], Sri Wahjuni[3]
[1,2,3]Department of Computer Science, Bogor Agriculture University, Indonesia
[1]Department of information technology, University of Ibn Khaldun Bogor, Indonesia
*Corresponding author, e-mail: yuggo.a@ft.uika-bogor.ac.id[1], hsrkom@ipb.ac.id[2], my_juni04@ipb.ac.id[3]

Abstract

Web server clusters require a reliable network management for increasing the quality of service (QoS). A load balancer system installed in a software-defined network (SDN) is one method that can improve the performance and availability of web server services. SDN is a dynamic and a programmable network management approach, and one protocol that supports it is OpenFlow. This research aims to design and analyse a model of a load balancer on OpenFlow networks, implementing a Weighted Round Robin (WRR) algorithm. The analysis process is conducted by measuring the value of a QoS web server performance parameters, such as response time, throughput, HTTP success, and loss connection. The results showed the WRR algorithm can be implemented for balancing a network system with dynamic resource allocation. The weight workload of each service can be obtained from the needs and existing network resources. The performance of a load balancer on an OpenFlow network is 57% better than in a traditional one for testing of response time conducted in a high connection. However, the throughput and HTTP success connection decreased by 2% and 10%, respectively, while HTTP loss connection increased by 49%.

Keywords: *load balancer, openflow, web server, weighted round robin*

1. Introduction

Computer networks have become an important part in many fields, such as business, entertainment, and education [1]. An example of applying computer networks in the education field is the campus network, where campuses applied a network that supports high-performance computing (HPC), This network comprises several services, such as web servers, database servers, storage area network, hosting, internet access, proxy, often use multiple servers to achieve reliability and high availability [2]. According to the ITU-T G.1000 documentation, the quality of network services will certainly be obtained from reliable network management, one can measure by looking at the performance of QoS parameters.

Challenges in network management are to operate, to maintain, and to protect computer networks [3]. The existing computer network is still traditional and static. Most existing computer networks still use device layer 2 and layer 3, such as switches and routers [4]. Software-defined networking (SDN) is a new idea for the computer network, which simplifies control and management function networks. SDN allows innovation through a dynamic network and virtualized network and is capable of being programmed. One of the protocols that implement SDN is OpenFlow [5]. OpenFlow can be implemented in a data center, based on cloud computing, wherein cloud computing provides convenience and efficiency through virtualization management resources that can be accessed anytime and anywhere, meeting the demand for IT resources [6]. Load balancing system is one of the solutions to establish a reliable and scalable design of infrastructure to handle load request. The load balancer is a bridge between the network and server, which can enhance the performance by immediately sending incoming requests to several servers. So a load of each server is much lighter [7-8].

This research aims to analyze and to design a load balancer system for a web server cluster using Weighted Round Robin (WRR) algorithm in OpenFlow networks. WRR needs two dynamic parameters, i.e. totals connection and throughput to servers. In addition, different Web service users have different QoS attribute preference [9]. So we analyze the performance of QoS of a load balancer in OpenFlow by comparing in the existing load balancer. Finally, the

Received January 10, 2017; Revised February 18, 2018; Accepted March 25, 2018

paper uses the dynamic weighted round robin algorithms to solve the optimization of load balancer on a traditional network to OpenFlow network.

2. Research Method
2.1 Design of load balancer system with WRR algorithm

The first step is designing a load balancer system, conducting a WRR in the OpenFlow-based network as shown in Figure 1. In this research, the M/M/3 queue model is used for our simulating network. In the system, arrival process is assumed as a Poisson process [10]. Packets delivered through the queue system are assumed to come at random, using stochastic processes.

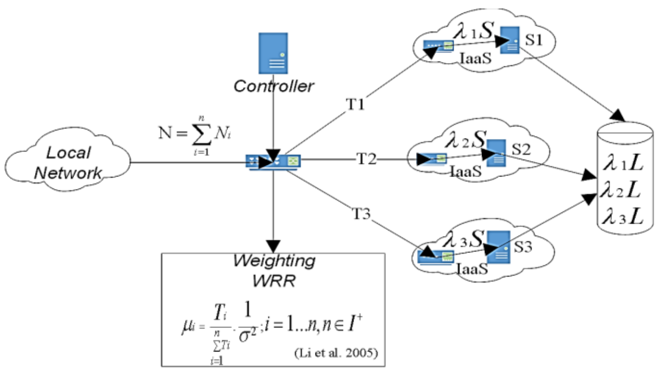

Figure 1. Topology design of load balancer system using WRR

In our simulation, we established a queuing network simulation and testing that follows Jackson theory [11]:

a. The arrival of the packet is assumed to be random, following the Poisson process set by the researchers used as a test scenario.
b. Mean service time is exponentially distributed.
c. Discipline queue used is the order of priority.
d. The number of servers used is 3 servers.

2.2 Implementation of load balancer system with WRR algorithm

The second step is developing a load balancer system using WRR algorithm. There are several stages of this process, which can be seen in Figure 2, including data acquisition, preprocessing of data, and the application WRR in OpenFlow network.

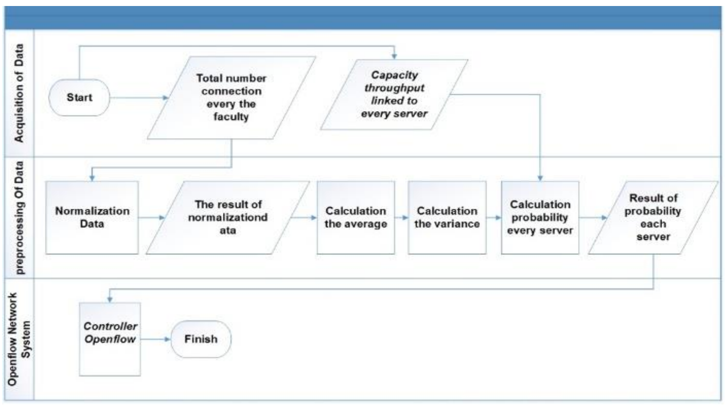
Acquisition of Data
Start
Total number connection every the faculty
Capacity throughput linked to every server
preprocessing Of Data
Normalization Data
The result of normalizationd ata
Calculation the average
Calculation the variance
Calculation probability every server
Result of probability each server
Openflow Network System
Controller Openflow
Finish

Figure 2. Process diagram of load balancer system with WRR algorithm

2.2.1 Data acquisition

There are two stages in acquiring data:

a. Total connection of each faculty

The value of total connection at every faculty is collected by recording total connection to the web server, from an existing database application. Data is required for each faculty per one day and calculated using the Formula (1).

$$N\ (request)\ = \textstyle\sum_{i=1}^{n} N_i \qquad (1)$$

Where N_i is the total number of connection of each faculty.

b. Throughput to each server link

Throughput data of each linked server are generated by Iperf application. Iperf is tested for its suitability to data by comparing the average value of Iperf and Wireshark application. Different test paired samples are used in the testing process, using application PSPP. In this research, three link servers are used.

2.2.2 Preprocessing of data

There are four stages in preprocessing of data:

a. Normalisation Data

Data normalisation of total connection each faculty calculated, because there are large differences in the range of data. Based on the total connection every faculty earlier, at the beginning of the data acquisition process. The range data will be set to 0 to 1 using Formula (2).

$$N_i = \frac{n_i - min}{\max - min} \qquad (2)$$

Where N_i is the total value of the normalisation of connection for every faculty; n_i is the total number connection for every faculty; min is the lowest value of the data set, and max is the highest score of the data set.

b. Calculation the average

The average values of total number connection for every faculty is calculated using Formula (3).

$$\overline{\mathrm{N}} = \frac{\sum_{i=1}^{n} N_i}{n} \qquad (3)$$

Where $\overline{\mathrm{N}}$ is the average values of total number connection every faculty.

c. Calculation the value of variance

The value of variance as one of key determine the probabilities value of each service server [9]. It is calculated using Formula (4).

$$\sigma^2 = \frac{\sum_{i=1}^{n}\left(N_{t,i} - \overline{N_t}\right)^2}{n} \qquad (4)$$

Where σ^2 is the value of connection variance; $N_{t,i}$ the total value normalisation of total connection for every faculty.

d. Calculation probability each server

Based on the variance and throughput value of each linked server, the probabilities of services server of each web server based on OpenFlow will be calculated using Formula (5).

$$\mu_i = \frac{T_i}{\sum_{i=1}^{n} T_i} \cdot \frac{1}{\sigma^2}\ ; i = 1 \ldots n, n \in I^+ \qquad (5)$$

Where μ_i is the value the define probabilities of the server services; T_i is the value of throughput capacity of each linked server.

2.2.3 Implementing load balancer system in openflow network

There are two stages in implementing load balancer system in OpenFlow network:

1. Topology Design

Topology design builds to implement scenarios for load balancer system based on OpenFlow network. Several components are created such as OpenFlow switch as data plane, OpenFlow controller as control plane, web server and host. Topology design how data communication of OpenFlow protocol can be seen in Figure 3.

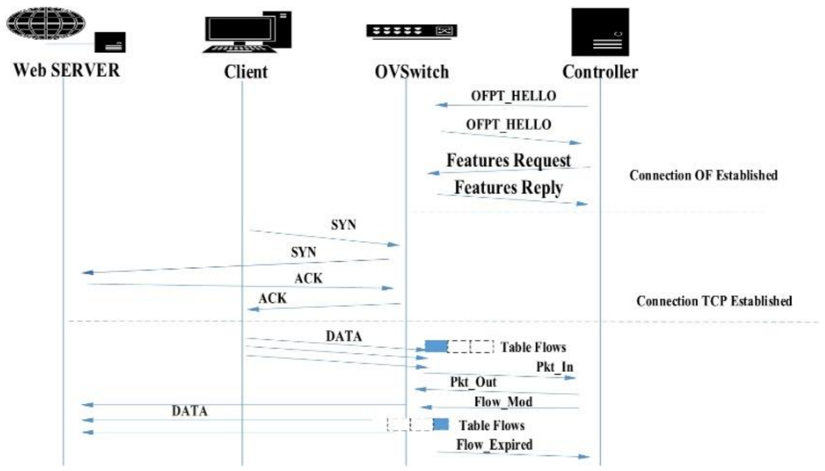

Figure 3. OpenFlow protocol communication schema

Figure 3 shows the initial process of OpenFlow protocol communication, where there are initial messages between the controller and OpenFlow switches, including:

a. OFPT_HELLO
   Controller to switch is following the TCP handshake, the controller sends its version number to the switch, and switch to the controller is the switch replies with its supported version number.
b. Request Features
   Controller to switch is the controller asking to see which ports are available.
c. Features Reply
   Switch to the controller is the switch replies with a list of ports, port speeds, and supported tables and actions.

Next, after the OpenFlow protocol communications are established, the client tries communication to a server through a switch, where the TCP three-way handshake between the client and server to switch has been established. The switch will handle client and server communication, based on knowledge flow table, where knowledge flow table of switch is obtained from the controller, where there are several formats of messages between the controller and OpenFlow switches, including:

a. Packet-In
   Switch to the controller is a packet received, and it matched no entry in the switch's flow table, causing the packet to be sent to the controller.
b. Packet-Out
   Controller to switch is a controller to send a packet out one or more switch ports.
c. Flow-Mod
   Controller to switch is instructed as a switch to add a particular flow to its flow table.
d. Flow_Expired
   Switch to the controller is a flow timed out after a period of inactivity.

2. Flow Process

Communication logic and WRR algorithm on OpenFlow network in a load balancer system are implemented on the switch as a data plane process. It is performed in control plane using Pox controller in python programming language. Figure 4 shows how the communication process is handled by the OpenFlow switch. Packets arrive from the network will be checked for the packet header information. If it is matched to *I*, where *I* is a number of index rule on *N*, and *N* is a total number of available table flow rules. If appropriate, the parameters of the action will be processed to WRR algorithm, next it is will be executed and the value of counters will be updated. But if it does not match then the packet will be checked to the next *I* until reach N. If it is not matched then the packet will be sent to the controller to further process.

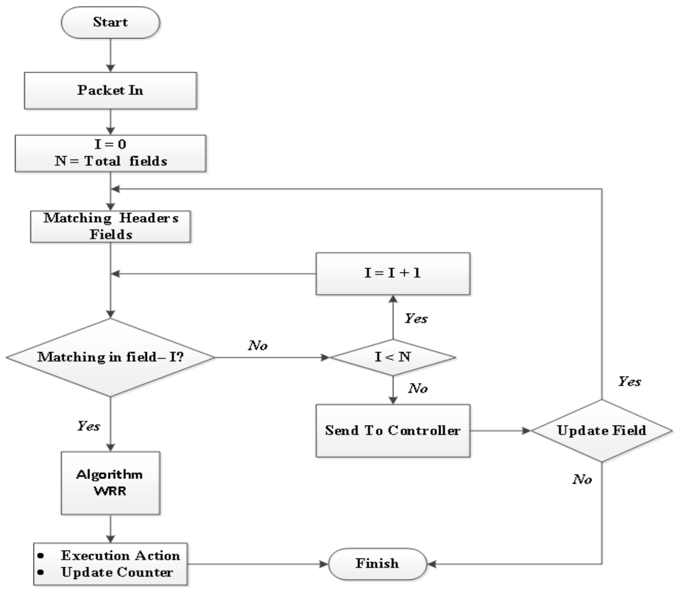

Figure 4. Flow process communication on OpenFlow network

2.3 Performance Comparison load balancer system

The final step is to test the performance of both load balancer system schemes. The two schemes are load balancer with one domain multiple IP (ODMP) methods on an existing network and load balancer with weighted round robin algorithm (WRR) on OpenFlow network. Connection data are generated randomly with varying length of 4000, 6000, 8000, 10000, and 100000 total connection, with the inter-arrival varying length of 100, 200, 300, 400, and 500 connection per second as a load of connection.

The test process is repeated 10 times for each total connection and sequence inter-arrival for both load balancer systems. The value of QoS is compared to get the best performing load balancer system. The tests are conducted using three PC units, operating system Ubuntu 14.04 GNU/Linux 64-bit, and applications Tsung as the generator request connection.

3. Result and Analysis

3.1 The results of testing performance throughput link server

The value of measurement performance of each link server is obtained from measuring Throughput value. Tools for measuring throughput value is using Iperf and Wireshark applications. The result of each application will be analysed using paired two-sample test method with 10 time testing. Table 1 and 2 show the result analysing the process.

Table 1. Result of throughput

Application	Result of Throughput (kbps)									
	1	2	3	4	5	6	7	8	9	10
Iperf	7.740	7.430	7.970	8.240	9.500	7.210	6.880	9.740	10.000	8.360
Wireshark	7.810	7.524	7.968	8.377	9.567	6.838	7.112	10.038	10.117	8.399

Table 2 shows the value of the correlation between the two variables is 0,99 with sig is 0. It indicates the correlation for the value of average throughput testing Iperf and Wireshark is strong and significant. Table 2 shows value is t 1,20 with sig 0,259. The value of significant (sig) is greater than 0.05, so it can be concluded that Ho is not rejected. The average throughput from testing with Iperf and Wireshark is the same, and Iperf can be a tool to collect data throughput value, which will be one of the parameters to calculate the value of probability weighting server.

Table 2. Result paired two-sample test

	Paired Differences					t	Df	sig. (2-tailed)
	Mean	Std, Deviation	Std, Error Mean	95% Confidence interval of the difference				
				Low er	Upper			
Pair 1 Wireshark-Iperf	.07	.18	.06	.06	.20	1.20	9	.259

3.2 The results of connection monitoring on load balancer system

The result of connection load distributed to each server is obtained by monitoring of both load balancer system. The monitoring is conducted using simulation experiments with total connection=100, throughput capacity link server 1=498 Kbps, server 2=167 Kbps, and server 3=493 Kbps. The result of monitoring can be seen in Table 3.

Table 3. Result divider load distribution connection to the server

No	Load Balancer System	IP Server	Result Distribution
1	Existing netw ork load balancer w ith ODMP	Server 1	33
		Server 2	35
		Server 3	32
2	OpenFlow netw ork load balancer w ith WRR	Server 1	48
		Server 2	9
		Server 3	43

Table 3 shows the results of distributed connection success to each web servers. The existing network resulting in a weight ratio of each server is s1: s2: s3=1: 1: 1 and distributed 33% on server 1, 35% in server 2, and 32% on server 3, where probability value of weight ratio in each test is static. The OpenFlow network resulting in a weight ratio of each server is s1: s2: s3=4:1:4 and distributed 48% on server 1, 9% on server 2, and 43% on server 3, where probability value of weight ratio is dynamic in each test.

3.3 The results of testing comparison system load balancer

The results value of performance testing of both load balancer system is taken the comparative value of QoS (Throughput, response Time, HTTP Success Connection, and HTTP Loss Connection). The mean aggregation value of total connection of each test is acquired by performing 10 repetitions and conducting the test data selecting, which only takes the results of the test data by selecting data generating the same total connection by Tsung application. The result of testing can be seen in Table 4 and Figure 5.

Figure 5 shows a balancing system on OpenFlow networks having better performance for large network connection. The response time value is required in the web-based application service transaction process. The value of throughput is not much different. The more successful HTTP packages in the web server connection the greater throughput. it indicated by the Pearson correlation test result is 0.668, this mean correlation for the number of HTTP success connection and throughput is strong. The total HTTP success connection and HTTP loss connection obtained in this test are affected by the success of the Tsung application in generating total requests, where Tsung applications have limitations in the ability to write file descriptors in the operating system kernel in large numbers, in addition to the time propagation in determining data forwarding in the algorithm of load balancer WRR systems on OpenFlow networks that make HTTP success connection and HTTP loss connection in the ODMP load balancer system better.

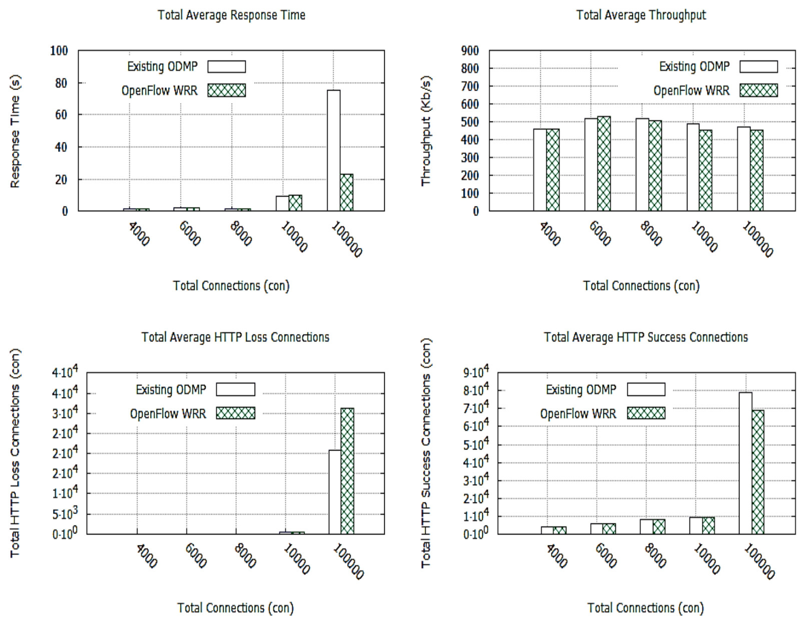

Figure 5. Graph of testing comparison load balancer system

4. Conclusions

The OpenFlow protocols design model can be implemented in data centre campus networks, as the load balancer system on a web server cluster. WRR algorithm can be used for scheduling to determine a load of service tasks to each server. Total connection parameter at every distribution level and transfer link capacity of each server can be used for determining server workload. SDN, based on OpenFlow protocol, can develop an appropriate and dynamic system based on the needs and problems that exist on a campus network.

A load balancer system of OpenFlow networks has better performance compared to the traditional one in many conditions of our testing scenarios. Our simulation resulted that WRR in OpenFlow networks has decreased in throughput by 2% and has increased the loss connection 49%. However, it is also able to increase response time by 57% and HTTP success connection by 10%.

References

[1] N Mckeown *et al.* OpenFlow: Enabling Innovation in Campus Networks. 2008.

[2] W Yahya, A Basuki, J Jiang, A Info. The Extended Dijkstra ' s -based Load Balancing for OpenFlow Network. *International Journal of Electrical and Computer Engineering (IJECE)*. 2015; 5(2): 289–296.

[3] H Kim, N Feamster. Improving Network Management with Software Defined Networking. *IEEE Commun. Mag.*, 2013; 51: 114–119.

[4] V Khatri. Analysis Of OpenFlow Protocol In Local Area Network. Tampere University of Technology, 2013.

[5] S Azodolmolky. *Software Defined Networking With OpenFlow*. Birmingham (UK): Packt Publishing Ltd, 2013.

[6] Z Zhang, L Xiao, Y Tao, J Tian, S Wang, H Liu. *A model based load-balancing method in IaaS cloud*. Proc. Int. Conf. Parallel Process., 2013: 808–816.

[7] M Qilin, S Weikang. *A Load Balancing Method Based on SDN*. Proc. - 2015 7th Int. Conf. Meas. Technol. Mechatronics Autom. ICMTMA 2015: 18–21.

[8] D Lukitasari. Analisis Perbandingan Load Balancing Web Server Tunggal Dengan Web Server Cluster Menggunakan Linux Virtual Server. 2010; 5(2): 31–34.

[9] G Lu, Y Hai, Y Sun. A Reliable Web Services Selection Method for Concurrent Requests. *TELKOMNIKA (Telecommunication Comput. Electron. Control.*, 2014; 12(4): 1053.

[10] Idatriska, RRM, A Mulyana. Perancangan Dan Simulasi Antrian Paket Dengan Model Antrian M/M/N Di Dalam Suatu Jaringan. *JSM STMIK Mikroskil*, 2014; 15(2): 111–120.

MNO Sadiku, SM Musa. Performance Analysis of Computer Networks. 2013.