Comparative Analysis of HAProxy & Nginx in Round Robin Algorithm to Deal with Multiple Web Request

Raka Yusuf*, Rizza Syah Pasha Ulin Nuha**
*(Mercubuana University, Faculty of Computer Science, South Meruya St 1 No. 1 Kembangan, 11650, Indonesia)
** (Mercubuana University, Faculty of Computer Science, South Meruya St 1 No. 1 Kembangan, 11650, Indonesia)

Abstract:

Web based service providers like Google, have a large number of transactions every day. By using one web server, certainly all the transactions cannot be handled by it alone because of the excess user demand. Therefore we need a technology for distribution system and load balancing to minimize system failure. Load balancing server accepts request from each user then distributes to each of the back-end server. The purpose of this study is to verify software-based load balancing that runs on virtualization technology docker can run with its full capacity by distributing the load to each back-end server.

Keywords — Load Balancing, HAproxy, Nginx, Virtualization, Docker

I. INTRODUCTION

With the rapid development in the world of technology and increase number in users, one server node can be accessed by many users simultaneously. The result of these conditions increases the load on the server node, because one server node cannot handle a large number of users request which can cause a system failure. Therefore a solution is needed to handle simultaneous users request using cluster technology on the web server. With the existence of a web server cluster, user requests are divided based on the load on the server node located on the back-end cluster. The use of servers with distributed systems requires a method to provide load evenly on each server. Various studies have been conducted to regulate the division of workload on clustering servers in order to optimize system performance by applying load balancing methods. The application of load balancing technology in dividing the load on the server plays an important role in a system that has a heavy load on a daily basis, so that it can increase scalability on distributed systems. Other than requiring load balancing in a cluster system to divide the load, the load balancing also plays an important role in optimizing resource utilization, maximizing throughput, reducing latency and ensuring system failure tolerance.

II. RELATED RESEARCH

The main use of working with load balancing is to divide the load on each request to the back-end server according algorithm used. Factors that influence the load balancing performance are the resources used in each back-end server such as CPU, RAM and the load of each user request. Apache Performance Tuning, 2013 states performance of web server is mostly dependent to the memory. The higher the memory, the throughput will be good on the web server and on the contrary the lower the memory, the overall throughput will be poor on the web server [1].

Research conducted by Sameer Tamrakar, Anand Singh and Manoj Shakya, 2015, used exponential smoothing forecasting algorithm in HAProxy load balancer. The web server application used in this research is using apache and for the testing is using
apache jmeter. The results obtained from this test is memory plays an important role in determining engine performance, 2 nodes have 60% memory utilization and 1 node has 70% memory utilization [1].

Research conducted by Dongsheng Zhang, 2018, used software load balancer namely Nginx by applying multidimensional load balancing algorithm to distribute loads to 3 back-end web servers. The result obtained from the test is multidimensional load balancing algorithm is 9 times faster when compared to default load balancing from nginx [2].

Research conducted by Sukarno and Nur Laila, 2018, used round robin algorithm in HAProxy. 3 nodes on web server and 1 database server run under Virtual Machine. Docker device hub then use to connect each node on the web server. The results from the test that has been carried out using a round robin algorithm is HAProxy can divide the load into each server node alternately. Also, HAProxy can reduce overhead, service response time and decrease back end server resources [3].

Research conducted by EkoWidianto, 2018, used the least connection, round-robin and source algorithm are tested on HAProxy using 3 nodes of web server. The scenario used in this test sends high amount of transaction to HAProxy for each algorithm. The result is maximum load of each algorithm is 65connections per second. Source algorithm has poor throughput meanwhile least connection has the best throughput. Total number of connections that fail in source algorithm has the highest valued, round-robin and least connection algorithm can handle the load properly [4].

In this journal the writer will do the test using round-robin algorithm with different load balancer software such as HAProxy and Nginx. The difference in this journal with the previous literature is by using 2 software load balancers along with 3 web server nodes will be installed and configured above the docker virtualization. The expected output is 2 load balancer software can run optimally above docker virtualization by comparing response time, CPU and memory utilization for each load balancer software.

III. COMPARATIVE IMPLEMENTATION

A. Docker

Docker is an open source software under the Apache Version 2.0 license that can be used for free. It is avirtualization software that functions as a container that can hold a complete software along with its supporting components needed by the application. In virtualization technology in the machine level (Virtual Machine), Guest OS is generally needed where each server requires its own resources. Unlike Docker, applications that run on Docker virtualization use the same kernel so it can minimize the use of excessive resources such as CPU, memory, hard drive and network adapter.

B. HAProxy

HAProxy is a popular open source software used to handle TCP / HTTP request that can run on Operating Systems such as Linux, Solaris and freebsd. The cronjob which runs every few minutes is responsible for reading back-end information and updating configuration so that various request are transferred to member clusters if one of the nodes in the cluster fails. Generally this software is used to improve performance and reliability on servers by distributing workload to several servers (web, application and database).

C. Nginx

Nginx is an open source software that is generally used as a database server. Nginx can also act as a load balancer by distributing loads to the back-end server.

D. Apache Benchmark

Apache Benchmark (ab) is a tool commonly used to test the performance of HTTP servers for Linux devices. Ab’s way of working is to overrun with a large request towards the URL and retrieve data in the form of HTTP server performance results.

E. Scenario Testing

In this scenario test, installation and configuration will be performed in Docker as virtualization software and runs with the following hardware specifications:

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Processor</th>
<th>Memory</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dell Inspiron 7447</td>
<td>Core i7</td>
<td>8Gb</td>
<td>Ubuntu Xenial 16.04</td>
</tr>
</tbody>
</table>
The reason behind using minimalist hardware in this testing is because in addition to doing a comparison of two load balancer software, the writer will also prove the docker resistance level if it runs on laptop hardware. The system architecture that will be used in this test can be seen in fig 1.

In this test, the writer will install and configure a web server running on the docker with a simple HTTP configuration. The parameters that will be used to compare two load balancer software include CPU utilization, memory utilization, response time and the user success level that can be handled in the load balancer.

The test scenario in this journal can be seen in the below table:

<table>
<thead>
<tr>
<th>Simulation</th>
<th>LB Software</th>
<th>Web Server</th>
<th>User</th>
<th>Consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>HAProxy Container</td>
<td>3container</td>
<td>9000</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Nginx Container</td>
<td>3container</td>
<td>9000</td>
<td>10</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>HAProxy Container</td>
<td>3container</td>
<td>18000</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Nginx Container</td>
<td>3container</td>
<td>18000</td>
<td>10</td>
</tr>
</tbody>
</table>

IV. COMPARISON RESULT

The results of testing with the first scenario, HAProxy and Nginx can distribute the load to each back-end server. In fig 2 by testing 9000 users, response time in Nginx has increased by 2ms with user request of 1100, meanwhile HAProxy has increased response time by 2ms with 4500 user requests.

Average utilization CPU and memory in HAProxy for scenario 1 can be seen in fig 3. Average CPU in the amount of 42% meanwhile memory utilization in the amount of 36%.

Average utilization CPU and memory in Nginx for scenario 1 can be seen in fig 4. Average CPU in the amount of 46% meanwhile memory utilization in the amount of 36%.

The results of testing using apache benchmark with a simulation of 9000 users on HAProxy, there
were no failed transaction. Each user with 10 consistency level transaction in HAProxy can be provided 1.645ms response time, can be seen in fig 5.

The results of testing using apache benchmark with a simulation of 9000 users on Nginx, there were no failed transaction. Each user with 10 consistency level transaction in Nginx can be provided 1.954ms response time, can be seen in fig 6.

The results of testing with the second scenario, HAProxy and Nginx can distribute the load to each back-end server. In fig 7 by testing 18000 users, response time in Nginx has increased by 2ms with user request of 2200, meanwhile HAProxy has increased response time by 2ms with 8400 user requests.

Average utilization CPU and memory in HAProxy for scenario 2 can be seen in fig 8. Average CPU in the amount of 77% meanwhile memory utilization in the amount of 36%.

Average utilization cpu and memory in Nginx for scenario 2 can be seen in fig 9. Average cpu in the amount of 66% meanwhile memory utilization in the amount of 32%.

The results of testing using apache benchmark with a simulation of 18000 users on HAProxy, there were no failed transaction. Each user with 10 consistency level transaction in Nginx can be provided 1.666ms response time, can be seen in fig 10.
The results of testing using apache benchmark with a simulation of 18000 users on Nginx, there were no failed transaction. Each user with 10 consistency level transaction in Nginx can be provided 1.873ms response time, can be seen in fig 11.

V. CONCLUSION

From the results of testing conducted by the writer, for testing with a load of 9,000 users HAProxy has good results compared to Nginx. The average CPU obtained in scenario 1 is 42% and the response time obtained around 1.645ms for each user with 10 consistency level.

The second scenario test results conducted by the writer, for testing with a load of 18,000 users HAProxy gave good results compared to Nginx with response time 1.666ms for each user with 10 consistency level. But average cpu on HAProxy has increased to 77%.

There are no significant memory usage in each scenario, it provide around 36% memory utilization. It means docker virtualization can handle properly using laptop device. In the future research based on this journal, can be developed using docker virtualization with kubernetes orchestrator and can use a different tool for benchmarking HTTP request to perform utilization in each backend server.

<table>
<thead>
<tr>
<th>Scenario 1</th>
<th>User</th>
<th>Response Time</th>
<th>Avg CPU</th>
<th>Avg Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nginx 9000</td>
<td>1.954ms</td>
<td>46%</td>
<td>36%</td>
<td></td>
</tr>
<tr>
<td>HAProxy 9000</td>
<td>1.645ms</td>
<td>42%</td>
<td>36%</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scenario 1</th>
<th>User</th>
<th>Response Time</th>
<th>Avg CPU</th>
<th>Avg Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nginx 18000</td>
<td>1.873ms</td>
<td>66%</td>
<td>32%</td>
<td></td>
</tr>
<tr>
<td>HAProxy 18000</td>
<td>1.666ms</td>
<td>77%</td>
<td>36%</td>
<td></td>
</tr>
</tbody>
</table>

REFERENCES