Performance Comparisons of Web Server Load Balancing Algorithms on HAProxy and Heartbeat

Agung B. Prasetijo, Eko D. Widianto and Ersya T. Hidayatullah
Department of Computer Engineering, Faculty of Engineering – Diponegoro University
Jl. Prof. Soedarto, SH, Tembalang, Semarang, Indonesia 50275
E-mail: agungprasetijo@ce.undip.ac.id; didik@live.undip.ac.id; ersya77@ce.undip.ac.id

***Abstract*— Popular websites such as Google and Facebook are accessed by an extremely large clients and providing such clients only with a single web server is absolutely insufficient. To support service availability, two or more servers are required. This, however, needs a load balancing system. A load balancing server receives web traffic and distributes the requests to such multiple servers. Load balancing can be implemented with special hardware, software or a combination of both. The purpose of this research is to develop a load balancing system with HAProxy as a software-based load balancer and Heartbeat as failover software. The research also provides comparisons of the performance of several balancing algorithms on the system. The results show that without a load balancer, the load cannot equally be distributed. The system average failover time when an active server down is 10ms. The Leastconn algorithm, in general, outperformed the Round-Robin and Source algorithms in terms of connection rate, response time, throughput, and failed connection.**

Keywords— HAProxy; Heartbeat; Leastconn Algorithm; Round-Robin Algorithm; Source Algorithm

I. LITERATURE REVIEW

Popular websites have a huge number of clients to be served. To maintain a tolerable service time, several servers must be provided behind a load balancer. In literature, research on the implementation of load balancing mechanisms can be found as follows.

Literature [1] claimed that the low-overhead Distributed Packet Rewriting (DPR) to redirect TCP connections outperforms the legacy Round-Robin scheme and DPR algorithm [2]. In the proposed scheme, each server stores information about the remaining servers in the system as well as load information is maintained using periodic multicast amongst the cluster servers. Kaushal [3] explored the architectural design of virtual cloud using HAProxy. The result showed the application server is able to recover the system only in several milliseconds. Handigol et al. [4] proposed a system that tries to minimize the response time by controlling the load both on the network and on servers with the use of customized flow routing. They experiment the dynamic addition or removal of computer resource within the system.

Jakupović [5] emulated the load balancing in a clustered environment using Virtualbox, GNS3, HAProxy and Hearbeat. Wang He [6] explained the basic principle in using Heartbead and DRBD to increase the switching speed and stability. Literature [7] deployed Rucio, the ATLAS Distributed Data Management system which uses a three-layered scaling and mitigation strategy on OpenStack, Puppet, Graphite, and HAProxy. Experiments were done for knowing the interplay between such components. Literature [8] experimented on load-balancing the iRODS system (the integrated Rule Oriented Data Management System). It is claimed that inserting a load balancer on the iRODS is transparent and the impact to the client is negligible.

This paper explores the performance comparison between several legacy/traditional algorithms on a combination of HAProxy and Heartbeat system: Leastconn, the legacy Round-Robin and Source algorithms over unlimited/uncontrolled connections in terms of connection rate, response time, throughput and failed connections.

II. SYSTEM DESIGN

The environment for implementing 3 (three) web servers as well as the load balancer HAProxy and Heartbeat can be seen in Table I. The server is able to replicate the database from the other servers and able to conduct synchronization with the other two servers. The load balancing system may distribute traffic to the three servers. Besides, one laptop (Asus A43SD) worked as client and one unit switch is used (Cisco Catalyst 2960) to interconnect computers are also employed.

TABLE I. COMPUTER SPECIFICATIONS

Component	Server	
	Load balancer	**Website**
Processor	Intel Pentium CPU G620 2.60 GHz	Intel Pentium CPU E2200 2.20 GHz
Motherboard	ECS H61H2-M12	HP FT917AA-AR6
RAM	4 GB	4 GB
HDD	500 GB	320 GB
NIC	2 Fast Ethernet	1 Fast Ethernet

The HAProxy and Heartbeat as well as the three web servers are run on Ubuntu server 16.04 LTS x64. The database is backed up by MySQL Server 5.7 installed on the three Apache web servers. Unison is employed to support file synchronization on web servers. It replicates files and directories at once and store to any different hosts or different

hard drives. The HAProxy is used as a load balancer and mostly used as a reverse proxy purposes, while the Heartbeat is used for failover purposes to ensure a high web service availability. A detailed physical topology of this research can be seen in Fig. 1 and the related logical topology can be seen in Fig. 2.

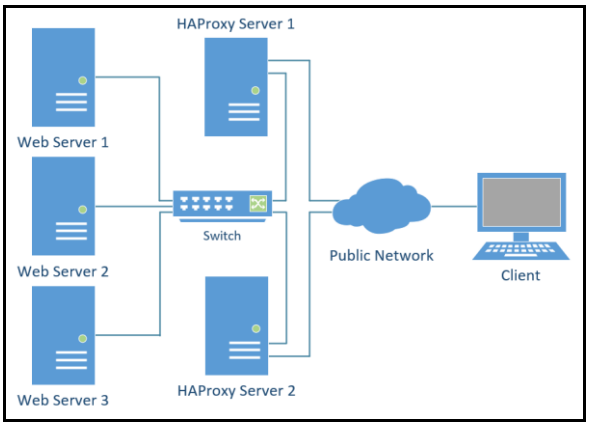

Fig. 1. Physical topology design

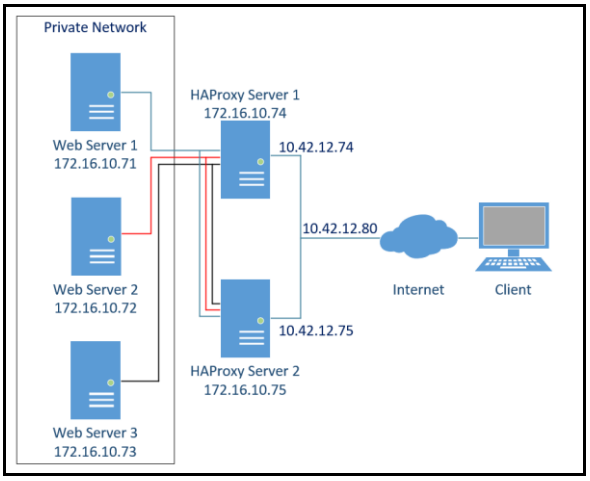

Fig. 2. Logical topology design

The replication system in MySQL being used is master – master: both the first and the second servers acts as master and slave, hence there are two masters and two slaves. The replication scheme is conducted in sequence from server #1 to server #2 to server #3 and back to server #1. A master – master synchronization mechanism is also used. Unison uses a star topology, where 1 server acts as a hub and the rests act as two spokes.

For the failover design, the two load balancing servers will be used as active – standby so when any one of these two servers fails, the other server will take over servicing the requests. The down-time in this research is limited to only 10ms, either by the disconnection of communication channel or by the failure of any one of the load balance servers.

III. Results And Discussions

There are two scenarios to measure the performance of the system with the help of Httperf: (1) webstress test without the use of a load balancer (affected by number of connections, connection rate, and timeouts); and (2) webstress test employing the load balancer with unlimited connections (HAProxy with Leastconn, Round-Robin, or Source algorithm).

Scenario (1) works as follows: requests are generated and directed to a single server only and records the failed connections. This is done for every server. Fig. 3 shows the number of maximum connections for such a scenario. Without a load balancer, requests handled by every web server are different. In this experiment the web server #1 performs the highest rejection rate. This means that without a load balancing algorithm, different load may be obtained for every server.

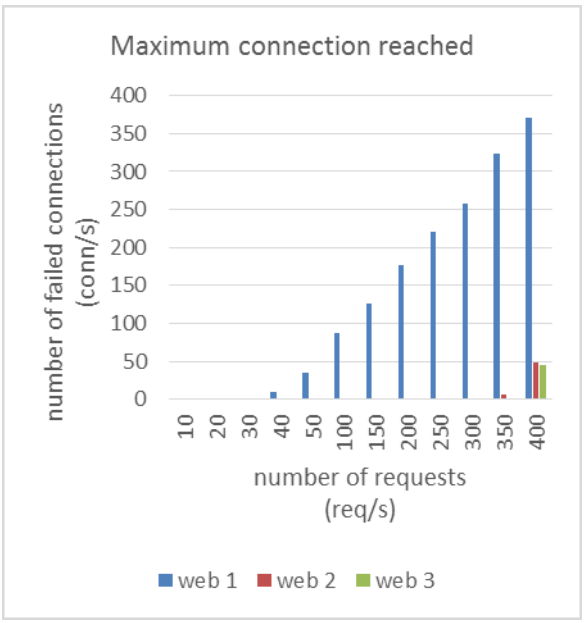

Fig. 3. Maximum connection on every web server in scenario (1)

For scenario (2), Fig. 4 shows the connection speed over the number of requests. Here, the requests are increased until the web server is unable to service such requests (some request timeouts occur).

It is shown that in light to moderate traffic, the connection rate of Leastconn algorithm outperformed the other two algorithms. The maximum load of every server is 65 connections per second regardless of the algorithm used.

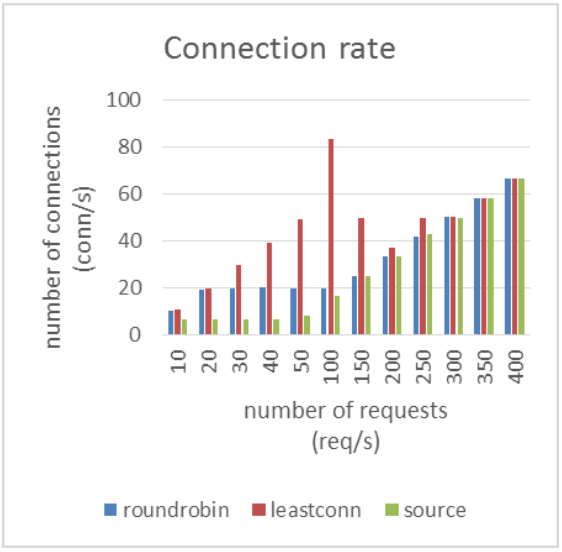

Fig. 4. Connection rate of every algorithm in scenario (2)

Whilst, the response time of the algorithms in scenario (2) is shown in Fig. 5.

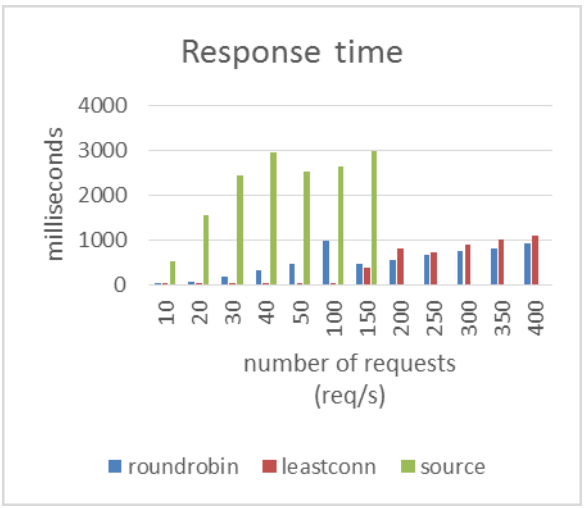

Fig. 5. Response time of every algorithm in scenario (2)

The Source algorithm performs the worst compared to the other two algorithms. The response time goes to 0 when the requests exceed 150 per second. This indicates the algorithm is not able to process the requests. Meanwhile, the Round-Robin algorithm performs the best in handling high traffic, but for low to average traffic, the Leastconn algorithm shows the least response time.

For throughput performance, the Leastconn algorithm outperforms the other two (see Fig. 6). The Source algorithm shows unsatisfactory results as this algorithm awaits the client connection finishes.

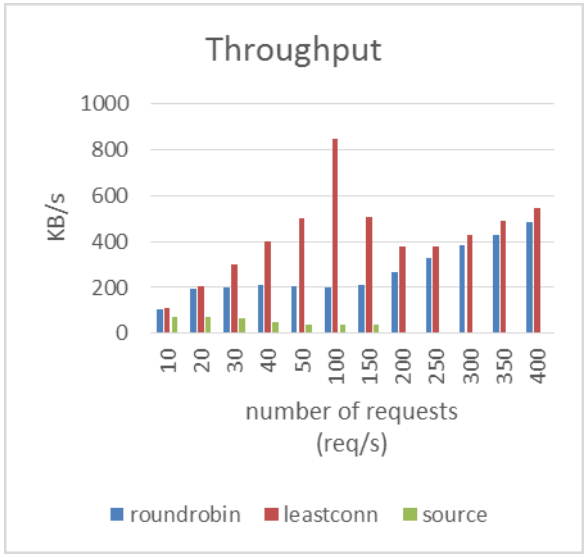

Fig. 6. Throughput of every algorithm in scenario (2)

Fig. 7 shows the failed connections in scenario (2). The Source algorithm shows an extreme number of failed connections compared to the other two algorithms. In contrast, the Leastconn and Round-Robin algorithms show that both can manage the connections better and the Leastconn algorithm outperforms the others.

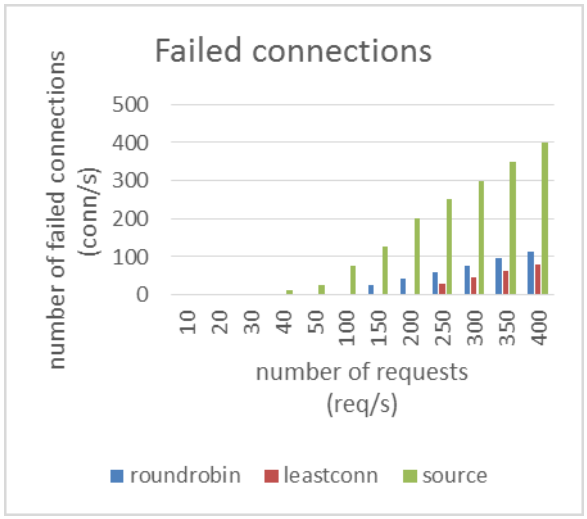

Fig. 7. Failed connections of every algorithm in scenario (2)

IV. Conclusions

The installed load balancing system is working normally and is able to distribute load and conduct failover. The replication process on the three nodes using one-way ring topology is successful. The service availability of this system is 10 milliseconds to handle active loads. The Leastconn algorithm is, in general, the algorithm that outperforms the other two schemes in terms of connection rate, response time, throughput, and failed connections.

REFERENCES

[1] L. Aversa and A. Bestavros. "Load Balancing a Cluster of Web Servers using Distributed Packet Rewriting", Technical Report BUCS-1999-001, Computer Science Department, Boston University, January 6, 1999. [Available from: http://hdl.handle.net/2144/1778]

[2] A. Bestavros, M. Crovella, J. Liu, and D. Martin "Distributed Packet Rewriting and its Application to Scalable Web Server Architectures," in Proceedings of ICNP'98: The 6th IEEE International Conference on Network Protocols, (Austin, TX), October 1998.

[3] V. Kaushal and M. Bala, "Autonomic Fault Tolerance Using HAProxy in Cloud Environment," Int. J. Adv. Eng. Sci. Technol., vol. 7, no. 2, pp. 222–227, 2010.

[4] Hanndigol, N., Seetharaman, S., Flajslik, M., McKeown, N., & Johari, R. (2009). Plug-n-Serve: Load-balancing web traffic using OpenFlow. ACM Sigcomm Demo, 4(5), 6.

[5] A. Jakupović, "Raspoređivanje Opterećenja I Povećanje," J. Polytech. Rijeka, vol. 1, no. 1, pp. 179–195, 2013.

[6] H. Wang, X. Sun, Z. Zhang, C. Li, X. Wang, I. Posov, and S. Pozdniakov, "Heartbeat and Drbd the application in the large capacity OLT," Proc. 2013 Int. Conf. Adv. Ict Educ., vol. 33, no. Icaicte, pp. 490–494, 2013.

[7] M. Lassnig, R. Vigne, T. Beerman, M. Barisits, V. Garonne, C. Serfon, "Scalable and fail-save deployment of the ATLAS Distributed Data Management system Rucio," Journal of Physics: Conference Series, Volume 664, IOP Publishing Ltd.

[8] Y. Kawai and A. Hasan, "High Availability iRODS System (HAIRS)", Proceedings Irods User Group Meeting 2010: Policy-based Data Management, Sharing, and Preservation, Eds. Reagan W. Moore, Arcot Rajasekar, Richard Marciano, pp. 11-15, DIC Foundation, 2010.