Komparasi Teknik *Load Balancing* Menggunakan Algoritma *Round Robin* Dan *Least Connection* Pada *Container Docker*

Oleh:
Ramdan Nugraha, Arip Solehudin, dan Agung Susilo Yuda Irawan
Universitas Singaperbangsa Karawang, Indonesia
Email: ramdan.nugraha17174@student.unsika.ac.id

Abstract

Single servers that often go down due to overload or high workload, this can be handled by the Cluster Server Load Balancing scheme, but the performance of this scheme needs to be measured. This research purpose to compare the performance of two algorithms, namely the Round Robin and Least Connection algorithms in implementing the Apache load balancer in Docker container-based virtualization. The performance test used is load testing using the Apache JMeter tool. The parameters measured are throughput and response time. This research uses the SIDLC (System Infrastructure Development Life Cycle) method. The research results show that the Apache load balancer with the least connection algorithm is superior because it can handle 300,000 sample requests with a throughput value of 2120.9 requests/seconds and a response time of 238 milliseconds. The Apache load balancer with least connection can handle http-requests in 141 seconds, this is 9 seconds faster than the round robin algorithm which is able to handle 300,000 sample requests in 150 seconds, then the error value for the least connection algorithm is still below 0.05% during the highest test peak, 300,000 sample requests.

Keywords: ***Apache, Load Balancing, Round Robin, Least Connection, Docker, Load Testing, SIDLC.***

A. Pendahuluan

Infrastruktur IT saat ini sedang mengalami perkembangan yang pesat dan semakin canggih salah satunya infrastruktrur *cloud*. Bagi tim IT Infrastruktur pada sebuah organisasi atau perusahaan agar aplikasi/sistem yang telah dibuat oleh *developer* dapat diakses dengan baik oleh banyak pengguna, maka tim infra atau *operation* perlu menyiapkan infrastruktur yang handal dengan *downtime* seminimal mungkin. Berbicara terkait infrastruktur IT tak terlepas dari yang namanya *server*. Teknologi virtualisasi merupakan salah satu teknik yang digunakan untuk membuat banyak *server* dalam satu mesin fisik. Berkat teknologi ini organisasi atau perusahaan dapat memangkas anggaran, sumberdaya, infrastruktur dan operasional yang dibutuhkan sehingga dapat menekan biaya pengeluaran yang cukup besar untuk infrastruktur IT.

Server tunggal seringkali mengalami *overload* atau kelebihan beban kerja yang menyebabkan server tersebut *down* karena meningkatnya permintaan (*request*) kebutuhan penggunanya. Meningkatnya kebutuhan informasi di Internet menyebabkan lalu lintas data semakin besar akibat banyaknya permintaan *(request)*, sehingga menyebabkan kelebihan beban pada salah satu layanan seperti *web server* dan kegagalan *server (overload)*[1].

Maka diperlukannya suatu konsep yakni *clustering computing* yang terdiri dari beberapa *server* virtual yang digabungkan sehingga diperlukannya juga penggunaan teknik *load balancing* supaya kinerja beban dapat lebih mudah didistribusikan ke beberapa *server virtual* yang terhubung. Berdasarkan permasalahan tersebut peneliti memahami penelitian terdahulu terkait bagaimana penerapan teknik *load balancing* untuk mengatasi beban kerja yang berat dan trafik yang tinggi pada *web server* serta penggunaan algoritma yang tepat dalam menyelesaikan masalah tersebut.

Sebagai solusi dalam meningkatkan keandalan serta ketersediaan *server web* bagi tim pengelola infrastruktur IT harus mempertimbangkan penerapan *clustering web server*[2]. Dalam penerapannya dapat menggunakan *Docker* agar dapat membuat *cluster server* secara *virtual*. *Docker* merupakan *platform opensource* yang menjalankan aplikasi secara otomatis ke dalam *container*[3]. Salah satu parameter pengujian yakni througput antara *single server* dengan *load balancing* yang menggunakan algoritma *round robin* dan *least connection*, hasilnya menunjukan teknik *load balancing* lebih baik daripada *single server*, karena dapat lebih mudah melayani *request* yang datang secara bersamaan.

Maka dari itu tujuan dari penelitian ini untuk mengimplementasikan teknik *load balancing* pada pengembangan infrastruktur virtualisasi denga metode *Systen Infrastructure Development Life Cycle (SIDLC)* menggunakan *docker* dan menganalisis kinerja dengan cara membandingkan dua algoritma yakni *round robin* dan *least connection* pada *apache* yang dijadikan sebagai *server load balancer*. Hasil nilai perbandingan dapat dijadikan sebagai acuan untuk penggunaan konsep

[1] Muhammad Aldi Aditia Putra, Iskandar Fitri, and Agus Iskandar, "Implementasi High Availability Cluster Web Server Menggunakan Virtualisasi Container Docker," *Jurnal Media Informatika Budidarma* 4, no. 1 (2020): 9, https://doi.org/10.30865/mib.v4i1.1729.

[2] Dimas Setiawan Afis, Mahendra Data, and Widhi Yahya, "Load Balancing Server Web Berdasarkan Jumlah Koneksi Klien Pada Docker Swarm" 3, no. 1 (2019): 925–30.

[3] J Turnbull, *The Docker Book: Containerization Is the New Virtualization* (James Turnbull, 2014), https://books.google.co.id/books?id=4xQKBAAAQBAJ.

load balancing pada virtual *server* berbasis *docker* sehingga dapat meminimalisir *cost server* fisik serta tanpa virtualisasi berbasis *hypervisor*.

Penelitian yang dilakukan diharapkan dapat mengoptimalisasi kinerja dari layanan *web server* dengan virtualisasi *docker*. Selain itu akan diterapkan dua algoritma ketika menjalankan skenario pengujian *load balancing web server* menggunakan *apache* pada virtualisasi *docker* antara algoritma *round robin* dan *least connection* sebagai perbandingan mana yang lebih unggul dan tepat untuk diterapkan pada keadaan banyak permintaan (*high request*) dan trafik yang tinggi agar *server* dapat berjalan dengan baik dengan meminimalisir *downtime server*.

B. Pembahasan

1. Kajian Teori

Beberapa kajian yang sudah digali terkait teori-teori atau *tech stack* dalam penelitian ini mengacu pada kebutuhan pengembangan infrastruktur server dengan konsep clustering server menggunakan teknik load balancing.

a. *Apache Web Server*

Apache adalah *software* untuk layanan *web server* gratis yang sangat populer, *Apache* banyak digunakan sebagai layanan *HTTP web server* untuk menjembatani komunikasi antara browser dengan aplikasi di *server*. *Apache* merupakan *web server* yang dapat bekerja dengan beberapa sistem operasi *(Unix, BSD, Linux, Mocrosoft dan Novell Netware* serta *platform* lainnya), yang berguna untuk melayani dan menjalankan suatu website [4].

b. *Container*

Container adalah bentuk virtualisasi yang menjalankan aplikasi didalamnya secara terisolasi, mirip dengan virtualisasi berbasis *hypervisor* atau sering disebut *VM (Virtual Machine)* yakni virtualisasi pada level *hardware*. Ada perbedaan yang mendasari keduanya ialah pada kernel, apabila *Virtual Machine* memiliki kernel dan *operating system* yang terpisah, berbeda dengan *container* bentuk virtualisasi yang berada pada level *system* sehingga menggunakan kernel yang sama dengan *host* diatasnya dan hanya mengisolasi aplikasi yang berjalan didalam wadah atau *container* tersebut beserta dependensi nya, maka dari itu *container* ini memiliki ukuran yang sangat ringan serta cepat saat dijalankan.

[4] Muhammad Sadeli, *Aplikasi Bisnis Dengan PHP Dan MySQL*, 1st ed., vol. 2 (Palembang: Maxikom, 2014).

c. *Docker*

Docker merupakan aplikasi *opensource* yang mengautomatisasi proses *deploy* aplikasi ke dalam *container*[5]. *Docker* sendiri yakni salah satu *platform* yang dibuat berdasarkan teknologi *container*. *Docker* berfungsi sebagai *container* yaitu sebuah wadah agar suatu aplikasi dapat di kemas dengan mudah dan lengkap dimana semua hal yang dibutuhkan aplikasi tersebut dapat berfungsi dan berjalan dengan baik. Ada perbedaan antara virtualisasi dengan *container* adalah dari segi ukuran file dimana *container* tidak perlu menyiapkan sistem operasi secara penuh sehingga memiliki ukuran file lebih kecil daripada virtualisasi.

d. Load Balancing

Load balancing adalah skema yang digunakan untuk penyeimbangan beban kerja pada *server* dimana pendistribusian trafik dilakukan secara seimbang melalui dua atau lebih jalur koneksi yang masuk ke *server*. *Load Balancing* yaitu teknik pemerataan di *server* terhadap pendistribusian beban kerja, jaringan, atau lalu lintas aplikasi[6]. Tujuannya adalah untuk meningkatkan fungsi sistem dari keandalan dan kinerja agar dapat menghilangkan terjadinya satu titik kegagalan dalam sistem. Beberapa algoritma yang umum dipakai untuk teknik *load balancing* ini yaitu algoritma *least connection* dan algoritma *round robin*.

e. *Algoritma* Round Robin

Algoritma *round robin* merupakan algoritma penjadwalan yang merutekan setiap permintaan masuk ke *server* berikutnya dalam daftar antrian[7]. Sebagai contoh pada saat ada 4 *request* yang masuk ke 3 *server cluster* (*server* X, Y, dan Z) maka *request* 1 akan masuk ke *server* X, *request* 2 akan masuk ke *server* Y, *request* 3 akan masuk ke *server* Z, dan *request* 4 akan masuk ke *server* X, sehingga menyelesaikan siklus perputaran pada *server*. Dengan algoritma ini memperlakukan semua *server* setara melihat dari jumlah *request* yang masuk.

---

[5] Nitin Naik, "Migrating from Virtualization to Dockerization in the Cloud: Simulation and Evaluation of Distributed Systems," in *2016 IEEE 10th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments (MESOCA)*, 2016, 1–8, https://doi.org/10.1109/MESOCA.2016.9.

[6] Shiang Liang Chen, Ethan Yun-Yao Chen, and Suang-Hong Kuo, "CLB: A Novel Load Balancing Architecture and Algorithm for Cloud Services," *Computers & Electrical Engineering* 58 (November 2016), https://doi.org/10.1016/j.compeleceng.2016.01.029.

[7] M E Mustafa and Amin MubarkAlamin Ibrahim, "Load Balancing Algorithms Round-Robin (RR), Least-Connection and Least Loaded Efficiency," *Vol* 1 (2017): 25–29.

f. Algoritma Least Connection

Algoritma *least connection* merupakan algoritma penjadwalan dinamis yang harus menghitung koneksi langsung pada setiap *server* secara dinamis. Algoritma penjadwalan ini artinya memiliki koneksi terendah dimana itu baik untuk kelancaran distribusi saat memuat permintaan yang bervariasi[8].

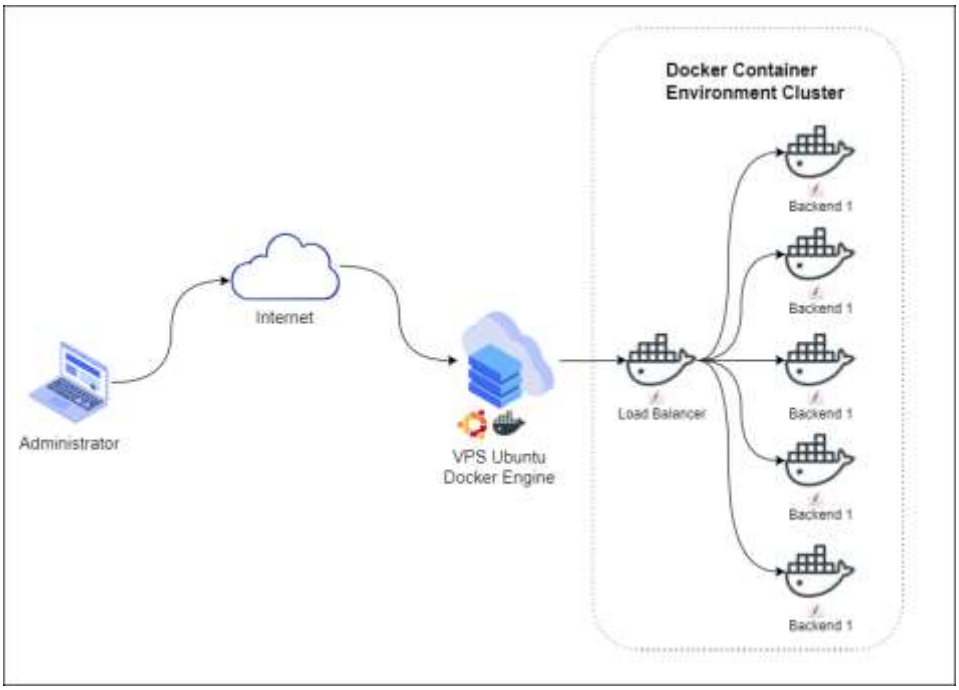
2. Implementasi

Pada penerapannya akan dikembangkan infrastruktur *cluster server* langsung diatas *VPS Cloud Provider AWS* dengan sistem operasi *Ubuntu Server 20.04 LTS*, dimana didalamnya sudah terinstal *Docker* sebagai *environment cluster web server apache* yang terdiri dari 1 *node* bertugas sebagai *Load Balancer* dan 5 *node* berfungsi untuk *server backend*. Berikut dibawah adalah topologi serta skema pengujian terhadap teknik *load balancing* yang diterapkan pada *container docker*.

a. Topologi Logic

Desain topologi yang dipakai pada penelitian ini adalah *Peer to Peer Over Internet Conection* antara *PC Administrator* dengan *server, administrator* akan terkoneksi melewati internet *public* menuju *Virtual Server* yang nantinya dapat mengelola *cluser server* berbasis virtualisasi *container* menggunakan mode *CLI* ataupun *GUI*.

Gambar 1. Topologi Logic

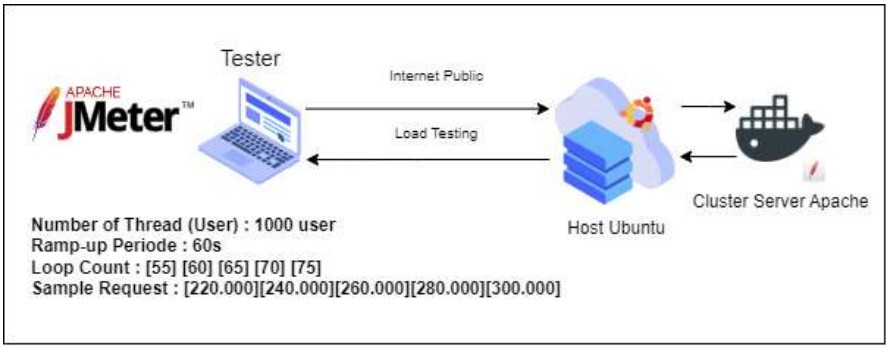


[8] Mustafa and Ibrahim.

b. Skema Testing

Pada skema pengujian *Software Performance Tools* yang dipakai pada penelitian ini adalah *Apache Jmeter*. *Apache Jmeter* akan melakukan *Load testing* dengan mensimulasikan *number of threads* (jumlah user) dan *Ramp-up periode* (waktu yg dibutuhkan) untuk mengakses atau mendapatkan *HTTP Request* dari sebuah *web server* pada setiap grup *server cluster* yang melewati *load balancer*.

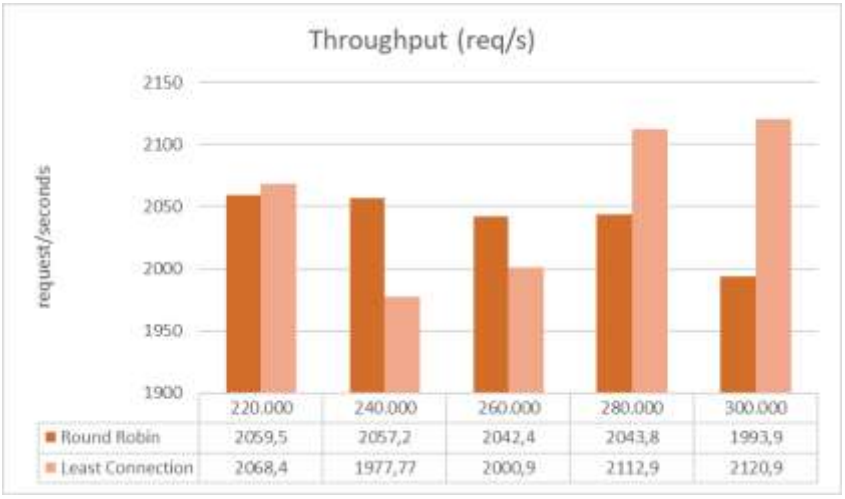
Gambar 2. Skema Pengujian



c. Hasil Pengujian

Hasil pengujian penelitian ini melihat kepada dua parameter yaitu *throughput* dan *response time* sebagai pengukuran seberapa optimal *load balancer* dapat menangani *request* dari rendah sampai tinggi, kemudian hal yang perlu diketahui adalah nilai *throughput* akan semakin baik apabila menghasilkan data yang lebih besar, berbeda dengan nilai *response time* semakin baik ketika menghasilkan data yang lebih kecil. Berikut ini adalah grafik perbandingan antara algoritma *round robin* dan *least connection* pada *apache* yang ditunjukan pada Gambar 3 dan Gambar 4 berikut:

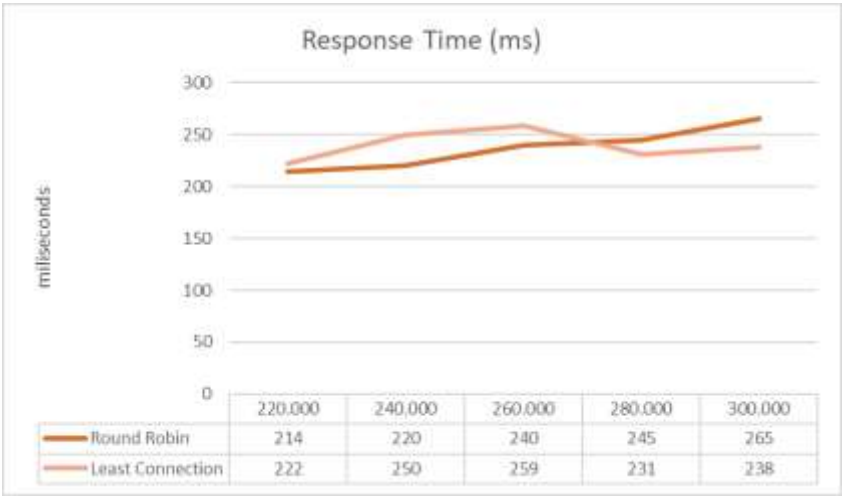
Gambar 3. Perbandingan Throughput Load Balance Apache



Untuk nilai *throughput* ketika menggunakan algoritma *least connection* pada saat awal-awal pengujian mulai dari 220.000 sampai 260.000 *sample*

request algoritma ini tidak stabil, namun menghasilkan nilai 2120,9 *request/second* dengan jumlah *sample request* tertinggi yakni 300.000 *sampel request*, sedangkan pada algoritma *round robin* mendapatkan nilai pada *sample request* tertinggi sebesar 1993,9 *request/second* hasil ini cenderung menurun dengan konsisten sejalan dengan urutan *sample request* yang diberikan dari rendah sampai tinggi.

Gambar 4. Perbandingan Response Time Load Balance Apache



Dalam parameter *response time* kedua algoritma memiliki perbedaan yang tidak terlalu signifikan, namun algoritma *least connection* sedikit lebih unggul dalam mengangani banyak *request* dengan waktu respon lebih cepat dibanding algoritma *round robin* ketika dua pengujian terkahir. Dimana algoritma *round robin* memiliki nilai 245 *ms* dan 265 *ms* sedangkan algoritma *least connection* menghasilkan nilai 231 *ms* dan 238 *ms* dalam dua pengujian terkahir berturut-turun sebesar 280.000 sampai 300.000 *sample request.*

C. Kesimpulan

Hasil dari perbandingan antara algoritma *round robin* dan *least connection* pada *load balancer apache* kali ini memiliki perbedaan yang cukup signifikan pada parameter *throughput* namun dari parameter *response time* memiliki hasil data yang cukup mirip dan konsisten untuk kedua algoritma. Algoritma *round robin* memiliki kecenderungan konsisten sejalan dengan *sample request* yang diberikan pada kedua parameter yakni *throughput* maupun *response time*, sedangkan algoritma *least connection* mempunyai perilaku yang cukup baik ketika diberikan 280.000 dan 300.000 *sample request* dengan nilai *throughput* yang besar dan *response time* yang kecil. Sehingga dapat disimpulkan untuk penggunaan algoritma paling optimal pada

load balancer apache adalah menggunakan algoritma *Least Connection* melihat dari keunggulan pada saat menangani *request* dengan parameter *throughput* dan *response time* menghasilkan nilai yang lebih baik daripada algoritma *Round Robin* dilihat dari puncak jumlah pengujian sebesar 300.000 *sample request*. Untuk penelitian selanjutnya dapat disarankan melakukan pengembangan pada *node server backend* yang diperbanyak serta apabila menggunakan *resource server* yang berbeda-beda dapat mendefinisikan bobot pada setiap *server* untuk algoritma *least connection* pada *load balancer Apache*, kemudian menambah jumlah *sample request* dengan menaikan *loop count* pada *tools performance testing Apache Jmeter* dengan catatan pada saat melakukan pengujian harus menggunakan koneksi internet yang stabil apabila menggunakan infrastruktur *cloud*.

Referensi

Afis, Dimas Setiawan, Mahendra Data, and Widhi Yahya. "Load Balancing Server Web Berdasarkan Jumlah Koneksi Klien Pada Docker Swarm" 3, no. 1 (2019): 925–30.

Chen, Shiang Liang, Ethan Yun-Yao Chen, and Suang-Hong Kuo. "CLB: A Novel Load Balancing Architecture and Algorithm for Cloud Services." *Computers & Electrical Engineering* 58 (November 2016). https://doi.org/10.1016/j.compeleceng.2016.01.029.

Mustafa, M E, and Amin MubarkAlamin Ibrahim. "Load Balancing Algorithms Round-Robin (RR), Least-Connection and Least Loaded Efficiency." *Vol* 1 (2017): 25–29.

Naik, Nitin. "Migrating from Virtualization to Dockerization in the Cloud: Simulation and Evaluation of Distributed Systems." In *2016 IEEE 10th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments (MESOCA)*, 1–8, 2016. https://doi.org/10.1109/MESOCA.2016.9.

Putra, Muhammad Aldi Aditia, Iskandar Fitri, and Agus Iskandar. "Implementasi High Availability Cluster Web Server Menggunakan Virtualisasi Container Docker." *Jurnal Media Informatika Budidarma* 4, no. 1 (2020): 9. https://doi.org/10.30865/mib.v4i1.1729.

Sadeli, Muhammad. *Aplikasi Bisnis Dengan PHP Dan MySQL*. 1st ed. Vol. 2. Palembang: Maxikom, 2014.

Turnbull, J. *The Docker Book: Containerization Is the New Virtualization*. James Turnbull, 2014. https://books.google.co.id/books?id=4xQKBAAAQBAJ.