# Analisis Performansi Web Server Menggunakan Load Balancing Pada Virtualisasi Docker Container

Thoyyib Abdillah[1], IGL. Putra Eka Prismana[2]
[1,2] Jurusan Teknik Informatika Fakultas Teknik Universitas Negeri Surabaya
UNESA Kampus Ketintang Surabaya
[1]thoyyib.17051204058@mhs.unesa.ac.id
[2]lanangprismana@unesa.ac.id

***Abstrak*— Pada saat ini teknologi mengalami perkembangan yang sangat pesat, termasuk pada performansi web server. Container merupakan teknologi virtualisasi terbaru, dengan adanya container akan memudahkan system administrator dalam mengelola aplikasi pada server. Docker container dapat digunakan untuk membangun, mempersiapkan, dan juga menjalankan aplikasi. Docker juga dapat membuat aplikasi dari bahasa pemrograman yang berbeda pada lapisan apapun. Aplikasi dapat di bungkus dalam container, dan aplikasi juga dapat berjalan pada lingkungan apapun dimana saja. Penelitian ini bertujuan untuk menguji kinerja web server ketika diakses, jika web server memiliki beban traffic yang tinggi maka sangat berpengaruh, kemampuan Web Server dipengaruhi oleh konfigurasi perangkat lunak pada IT infrastruktur. Untuk melihat ada atau tidaknya perbedaan antara sebelum dan sesudah konfigurasi pada web server dilakukan dengan melakukan uji beda paired sample test. Pada tes uji tingkat performa penulis menggunakan tools berupa httperf dan juga ab (apache benchmark) sebagai uji beban serta teknik analisis yang digunakan adalah Independent Sample Test (riset pengujian) terhadap dua sample.**

**Pada hasil penelitian ini menunjukkan bahwa dalam pengujian terhadap load balancing dengan menggunakan 2 node dan juga load balancing menggunakan3 node, menghasillan waktu dalam melakukan tes dengan 2 node yang lebih cepat daripada 3 node. Untuk request per second, load balancing dengan 3 node memiliki waktu request per detik lebih kecil daripada 2 node. Kemudian time per request yang dibutuhkan untuk load balancing 2 node lebih sedikit.**

***Kata Kunci*— *Virtualisasi, Container, Docker, Web Server, Load Balancing*.**

## I. PENDAHULUAN

Pada saat ini teknologi mengalami perkembangan yang sangat pesat, terutama pada teknologi Virtualisasi. Teknologi Virtualisasi merupakan sebuah konsep pendekatan dan penyatuan teknologi dengan berbagi sumber daya untuk meningkatkan penggunaan aset dan menyederhanakan manajemen sehingga sumber daya TI dapat lebih mudah memenuhi permintaan bisnis. Teknologi virtualisasi mengemulasi sumber daya fisik komputasi, seperti komputer desktop, server, prosesor dan memori, sistem penyimpanan, jaringan, dan aplikasi individu yang membentuk sebuah "lingkungan virtual". Dalam server atau jaringan, virtualisasi digunakan untuk mengambil sumber daya fisik tunggal lalu membuatnya dapat beroperasi seolah-olah sumber daya fisik tunggal tersebut memiliki lebih dari satu sumber daya. Hal ini dapat meningkatkan pemanfaatan dan efisiensi sumber daya aset, serta mengurangi biaya dengan mengurangi kebutuhan untuk aset fisik [1].

Container merupakan sebuah sistem yang memungkinkan dalam pembuatan layanan dan sumber daya yang berbeda dengan cara membagikannya di dalam sebuah operating system yang berjalan menggunakan libraries, drivers dan binaries yang berbeda. Konsep ini akan mengurangi jumlah sumber daya yang telah terbuang selama komputasi dikarenakan container hanya menjalankan dan menyediakan kebutuhan binary atau library yang sesuai dengan aplikasi ataupun layanan yang telah dijalankan [2]. Docker merupakan sebuah platform terbuka untuk siapapun yang bertujuan menggunakan sebuah platform untuk membangun, mendistribusikan dan menjalankan aplikasi dimanapun seperti laptop, data center, virtual machine ataupun cloud. Docker merupakan open source software yang berada di bawah Lisensi Apache Versi 2.0 yang bisa dipergunakan secara gratis. Docker dijalankan dengan menggunakan arsitektur client-server. Nantinya Docker client akan bertugas untuk menghubungi Docker daemon, yang melakukan pekerjaan berat, menjalankan, dan mendistribusikan Docker container pengguna. Kedua Docker client dan juga Docker daemon dapat berjalan pada sistem yang sama. Docker client dan Docker daemon berkomunikasi via sockets atau lewat API yang telah tersedia pada Docker. Docker memberikan berbagai macam keuntungan dan kemudahan dalam proses deployment software, dimana dalam infrastructure cloud dapat menjalankan banyak pekerjaan sekaligus dengan menggunakan docker dan AWS dalam mempercepat proses deployment, optimalisasi aplikasi dan isolasi. Platform ini dapat digunakan dalam melakukan pembangunan, mempersiapkan dan menjalankan aplikasi. Aplikasi dapat dibuat oleh Docker dari bahasa pemrograman yang berbeda pada lapisan apapun. Container dapat membungkus aplikasi didalamnya, dan aplikasi dapat berjalan pada lingkungan apapun dimana saja. Kelebihan docker pada layanan cloud ini dapat membantu permasalahan pada aplikasi yang menggunakan multi container dan layanan pada server cluster shared. Nantinya Kontainer Docker akan membungkus perangkat lunak dalam file dengan lengkap beserta sistem yang berisikan semua data yang diperlukan ketika menjalankan: kode, runtime, alat sistem, pustaka sistem apa pun yang dapat diinstal pada server.

Ini dapat menjamin bahwa perangkat lunak selalu dapat berjalan secara bersamaan, terlepas dari lingkungannya [6].

Analisa Performansi Web Server ini dilakukan oleh PC host OS terhadap PC rekayasa yang dijalankan pada Virtual Machine. Virtual Machine atau VM merupakan versi akan melakukan fungsi komputasi lainnya serta memerlukan pemeliharaan seperti pembaruan dan pemantapan sistem karena Virtual Machine merupakan versi dari komputer fisik. Perangkat lunak mesin virtual dapat menjalankan program dan sistem operasi, menyimpan data, terhubung ke jaringan, dan melakukan fungsi komputasi lainnya, dan memerlukan pemeliharaan seperti pembaruan dan pemantauan sistem. Beberapa VM dapat di-*host* pada satu mesin fisik, seringkali *server*, dan kemudian dikelola menggunakan perangkat lunak (*software*) mesin virtual. Jika dilakukan ini akan memberikan fleksibilitas untuk sumber daya komputasi secara lengkap (komputasi, penyimpanan, jaringan) untuk didistribusikan di antara VM sesuai kebutuhan, sehingga meningkatkan efisiensi secara keseluruhan. Arsitektur ini menyediakan blok-blok pembangun dasar untuk sumber daya ter-virtualisasi canggih yang bisa digunakan saat ini, termasuk komputasi *cloud* atau yang dikenal dengan *cloud computing*. Virtual Machine Monitor atau Hypervisor lapisan perangkat lunak yang dapat menampilkan dan memvirtualisasikan sumber daya mesin cloud. Tingkat perangkat lunak membuat properti virtual seperti CPU, recall dan sistem operasi terdekat atau jauh [7]. Jika kita ingin dapat memunculkan halaman dari situs yang dimiliki oleh pengguna pada halaman web browser dan dapat diakses oleh satu ataupun banyak orang, maka dibutuhkan suatu web server. Dibutuhkan sebuah web browser seperti Iceweasel Konqueror, Internet Explorer, dan juga Opera pada sisi klien nya. Pada lingkungan GNU/Linux, apache merupakan web server yang paling banyak digunakan saat ini. Oleh Debian GNU/Linux, paket software apache telah dipaketkan dalam distribusinya.

Load balancing merupakan suatu proses dalam melakukan pendistribusian beban trafik pada dua atau lebih jalu koneksi secara merata antara dua atau lebih computer, link jaringan, CPU, hard drive, ataupun sumber daya dengan tujuan untuk memanfaatkan sumber daya yang optimal, memaksimalkan throughput, memperpendek waktu tanggap dan juga berupaya untuk menghindari terjadinya overload. Penerapan Load Balancing biasanya digunakan pada server virtual dengan melakukan virtualisasi, virtualisasi server merupakan teknologi untuk membagi sumber daya fisik secara virtual yang dapat melayani permintaan layaknya server. Pada penelitian Bansal & Kaur, (2015) yang berjudul "An Implementation of Servers using Lightweight Virtualization / Containerization", Server dapat di implementasikan pada mesin virtual menggunakan pendekatan hypervisor dari virtualisasi, tetapi karena kekurangan hypervisor, lebih baik menggunakan virtualisasi yang berbasis Sistem Operasi [5].

Benchmark/testing Apache workbench merupakan suatu standar atau tolak ukur yang memiliki manfaat dalam melakukan perbandingan antara satu hal dengan hal lainnya yang sejenis. Secara sederhana dengan menggunakan tolak ukur tersebut, maka berbagai hal akan dapat diukur dengan standar baku yang umum. Tool dalam melakukan pengukuran performance apache, dengan ab kita dapat melihat kapabilitas apache untuk melayani request dari client, Jika developer selesai menginstall Apache HTTP Server maka AB ini otomatis akan terinstal, jika apache sudah terinstal maka tool benchmarking ini secara otomatis juga sudah ada. Biasanya lokasi tool ab ini letaknya pada direktori. Tool apache benchmark ini dapat digunakan tidak hanya untuk Apache, tetapi juga untuk web server lainnya seperti Lighthttpd, Nginx ataupun Miscrosof IIS.

Dalam penelitian ini, penulis ingin melakukan perancangan arsitektur load balancing web server pada docker container yang berjalan pada komputer lokal. Perencanaan load balancing dapat menggunakan haproxy atau nginx. Dengan adanya NginX membuat server web tersebut menjadi lebih kuat, fleksibel, dan juga mampu dalam memberi keputusan pada server web mana yang akan diadopsi sepenuhnya tergantung pada kebutuhan pengguna [4]. Secara *default* routing pada load balancing menggunakan metode round robin yang meneruskan *request* secara urut dan merata pada setiap node. Peneliti menggunakan metode weighted round robin yang mana pendistribusian *request* berdasarkan beban yang didefinisikan pada masing-masing node. Kemudian peneliti melakukan *stress test* web server. Pengujian dilakukan untuk mengetahui tingkat performa web server menggunakan load balancing dengan web server *native*. Beberapa parameter yang digunakan untuk mengukur performa web server diantaranya *request per second, time to test* dan *time per request*.

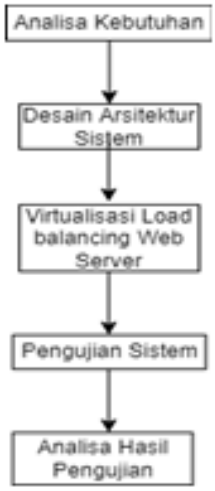
Penelitian terdahulu yang telah dilakukan oleh (M. Agung Nugroho et al, 2016) yang berjudul "Analisis Kinerja Penerapan Container Untuk Load Balancing Web Server Pada Raspberry PI" menyimpulkan bahwa hasil yang diperoleh setelah melalui ujicoba multi container dengan menerapkan load balancing dapat menunjukkan pembagian beban resource seperti memori, dan processor dari masing-masing container. Dan untuk single container hanya mampu menahan beban request yang sama dengan multi container, namun hanya akan optimal di salah satu core processor saja, sehingga beban tersebut dapat mengakibatkan server aplikasi menjadi down (tidak dapat diakses). Dengan melakukan penerapan load balancing pada multi container, beban akan terdistribusi seacra merata ke seluruh core processor, sehingga beban resources tidak akan mempengaruhi performa dari satu server aplikasi. Dari penelitian tersebut dapat dilakukan penelitian untuk mengukur variabel pengujian web server seperti request per second(rps), time per request dan lama waktu untuk menjalankan pengujian web server [3].

Berdasarkan analisa dari topik penelitian tersebut, penulis ingin melakukan suatu penelitian yang berjudul "Analisis Performansi Web Server Menggunakan Load Balancing Pada Virtualisasi Docker Container". Penelitian ini mengimplementasikan arsitektur load balancing web server pada virtualisasi berbasis docker container. Kemudian melakukan pengujian server menggunakan apache benchmark

untuk mengetahui performansi web server dengan load balancing sehingga diharapkan dapat menjadi pertimbangan dalam proses operasional web server secara efektif dan memiliki performa yang baik.

## II. METODE PENELITIAN

Penelitian ini merupakan penelitian berjenis eksperimen, yaitu metode penelitian dengan cara melakukan pengamatan tentang seberapa besar pengaruh dari Performansi Web Server dengan menggunakan Load Balancing yang dinaungi oleh Docker Container. Load Balancing berfungsi untuk melakukan pendistribusian traffic pada tiap-tiap server. Metode merupakan suatu tata cara yang dirancang dan juga dipakai untuk mencapai suatu tujuan tertentu. Berikut merupakan skenario penelitian :




Gambar 1. Skenario Penelitian

### *A. Analisa Kebutuhan*

Dalam analisis performansi web server ini, terdapat beberapa kebutuhan yang perlu dianalisa. Nantinya kebutuhan tersebut akan dipakai sebagai bahan dalam membantu analisis ini. Analisa kebutuhan dibagi menjadi beberapa bagian, yaitu:

#### *1. Kebutuhan Data*

Penelitian ini memerlukan data diambil dari beberapa referensi. Untuk melakukan mengukur variabel pengujian web server seperti request per second(rps), time per request dan lama waktu untuk menjalankan pengujian web server. terbagi menjadi dua jenis, yaitu studi literatur dan observasi.

*a. Studi literatur*

Peneliti mendapatkan data referensi dari laporan, jurnal nasional dan internasional, makalah, situs resmi dan juga beberapa sumber dari internet.

*b. Observasi*

Penelitian ini juga melakukan observasi dengan mengunjungi beberapa situs referensi Load Balancing dan Web Server serta situs dokumentasi docker.

#### *2. Kebutuhan Alat*

Spesifikasi perangkat yang diperlukan untuk melakukan penelitian ini adalah :

a. Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz 1.99 GHz
b. RAM 3 GB
c. Hardisk 30 GB
d. Sistem Operasi Ubuntu 20.04 LTS

Sedangkan perangkat lunak yang diperlukan dalam penelitian ini adalah sebagai berikut :

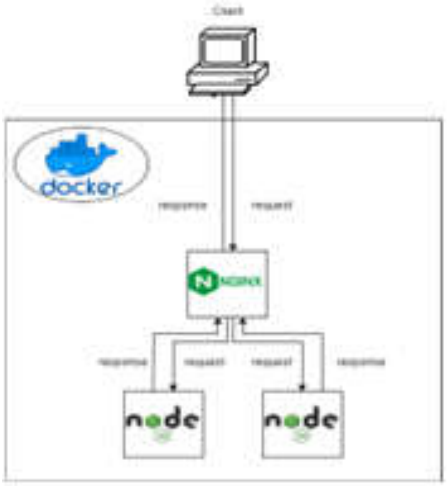
a. Ubuntu sebagai media dalam membangun aplikasi dalam container
b. Docker sebagai wadah untuk menjalankan web server berbasis container.
c. Apache benchmark sebagai alat untuk melakukan pengujian

### *B. Desain Arsitektur Sistem*

Sistem yang akan dikembangkan ialah Virtualisasi Web Server menggunakan load balancing. Didalam desain sistem terdapat beberapa artitektur web server yang akan dilakukan pengujian diantaranya sebagai berikut :

1. Arsitektur web server 2 node

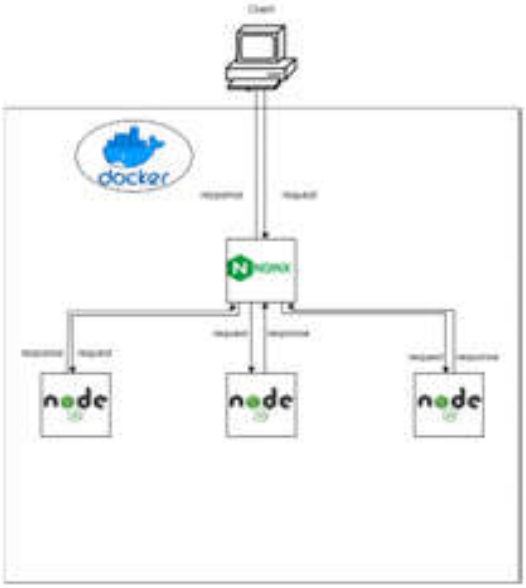
Arsitektur sistem pada penelitian ini adalah rancangan komponen atau teknologi yang digunakan dalam membangun web server yang memungkinkan klien untuk terhubung pada suatu layanan. rancangan arsitektur pertama menggunakan loadbalancing dengan dua node atau web server yang mana dideskripsikan pada gambar dibawah:




Gambar 2. Arsitektur web server 2 node

2. Arsirektur web server 3 node

Arsitektur sistem pada penelitian ini adalah rancangan komponen atau teknologi yang digunakan dalam membangun web server yang memungkinkan klien untuk terhubung pada suatu layanan. rancangan arsitektur pertama menggunakan loadbalancing dengan dua node atau web server yang mana dideskripsikan pada gambar dibawah:




Gambar 3. Arsitektur Web Server 3 Node

### C. Pengujian Sistem

Pada tahap ini, setelah web server berhasil dibuat kemudian dilakukan pengujian. Pengujian ini dimaksudkan untuk mengetahui performansi web server menggunakan load balancing yang sudah dirancang. Pengujian web server menggunakan load balancing dilakukan menggunakan alat tambahan yaitu apache benchmark. Pengujian dilakukan dengan mengirim beberapa request ke web server dengan tiga skenario, yaitu load balancing web server dua node, tiga node dan empat node dengan mengirimkan beberapa request menggunakan apache benchmark.

### D. Analisa Hasil Pengujian

Untuk memudahkan dalam melakukan pengujian performansi load balancing pada web server, dan juga untuk mempermudah pengguna dalam memahami luaran penelitian, dibuat tabel perbandingan performansi dengan skenario menggunakan beberapa node dan request saat benchmarking dengan memperhatikan parameter diantaranya request per second(RPS) ialah metrik yang mengukur throughput sistem, yang merupakan jumlah request dalam satu detik. Dalam apache benchmark RPS didapatkan dari jumlah rata-rata jumlah request per detik. time per request(TPS) adalah jumlah waktu rata-rata yang diperlukan untuk memproses sekelompok permintaan ke server secara bersamaan. Dalam apache benchmark TPS didapatkan dari rata-rata waktu yang diperlukan untuk satu permintaan (termasuk concurrent). concurrent ialah sebuah proses yang dijalankan secara sekaligus atau sebuah proses yang dapat melakukan beberapa hal secara bersamaan yang dilakukan secara sekaligus. dan time taken for test ialah mengukur durasi pada saat ApacheBench pertama kali terhubung ke server dan saat menerima respons akhir, dalam apache benchmark dapat diartikan sebagai waktu yang dibutuhkan untuk melakukan pengujian.

## III. HASIL DAN PEMBAHASAN

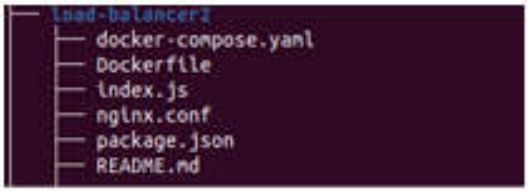
Setelah melakukan perancangan topologi jaringan dan analisa kebutuhan data peneliti melakukan penerapaan :

### A. Instalasi Lingkungan Server

Sebelum melakukan pembuatan loadbalancing pada web server dan pengujian peneliti menyiapkan lingkungan server dan perangkat lunak yang dibutuhkan dalam penelitian. Untuk membangun web server peneliti menggunakan docker container agar web server dapat diringkas menjadi sebuah *bundle* berbentuk images. Perintah instalasi yang digunakan pada ubuntu adalah "sudo apt-get install -y docker.io". Setelah docker terinstal agar dapat melakukan benchmarking, selanjutnya melakukan penginstalan apache benchmark pada ubuntu dengan perintah "sudo apt-get install -y apache2-utils".

### B. Virtualisasi Load Balancing Web Server

Tahap selanjutnya ialah membangun load balancing pada web server berbasis docker container. Dalam membangun image docker peneliti melakukan pendefinisian melalui skrip kode docker compose berformat yaml. docker compose digunakan agar dapat melakukan konfigurasi secara mudah. berikut direktori file dan folder yang didefinisikan untuk docker compose.



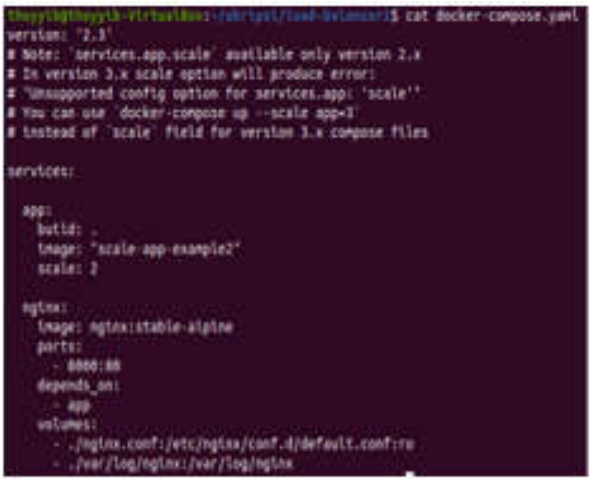

Gambar 4. Direktori file dan folder

Dockerfile digunakan untuk mendefinisikan baris kode pembuatan image docker, base yang digunakan adalah node versi 6 alpine linux. Kemudian mengatur direktori default aplikasi dalam container. Dalam node js penginstalan aplikasi dengan menyalin package dependensi ke folder default dan melakukan penginstalan pada direktori tersebut. selanjutnya file aplikasi di jalankan pada node js seperti gambar dibawah.


Gambar 5. Dockerfile

Setelah file build docker didefinisikan. kemudian menyiapkan file docker-compose dengan format yaml untuk memuat komposisi image dan layanan kontainer dalam membuat load balancing. untuk membuat beberapa web

server pada load balancing digunakan fungsi scalling. berikut gambar docker compose.




Gambar 6. Docker compose

Perintah docker compose digunakan untuk menjalankan image dan container dalam satu konfigurasi. berikut command docker compose. image dan kontainer yang dijalankan akan digunakan sebagai media pengujian seperti gambar dibawah.

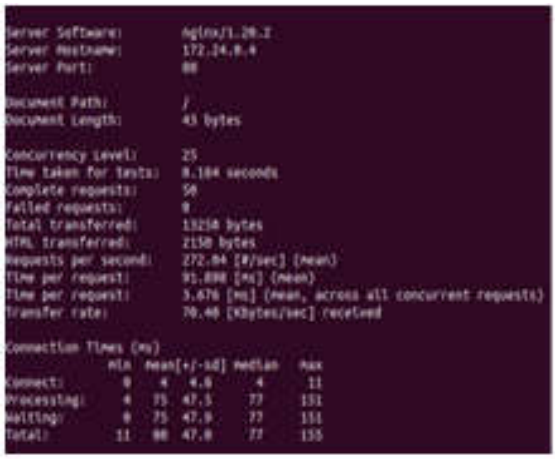

Gambar 7. Perintah docker compose

*C. Analisa Pengujian Web Server*

Pengujian dilakukan dengan apache benchmark dengan perintah "sudo ab -n -c " yang mana dalam sintaks apache benchmark -n adalah jumlah request dan -c adalah concurrent disetiap request.

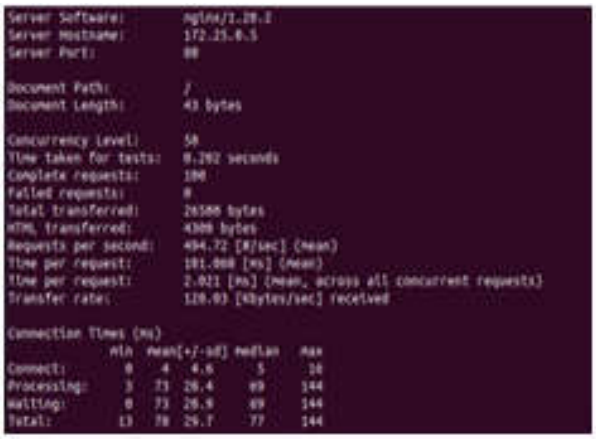
1. Pengujian dua node

Pengujian pertama menggunakan apache benchmark dilakukan dengan 50 jumlah request dan 25 concurent. dari hasil pengujian tersebut didapatkan 0,184 seconds untuk time taken for test,272,04[sec] (mean) request per second dan 91, 898[ms] (mean) time per request. Berikut hasil pengujian menggunakan apache benchmark seperti gambar dibawah;



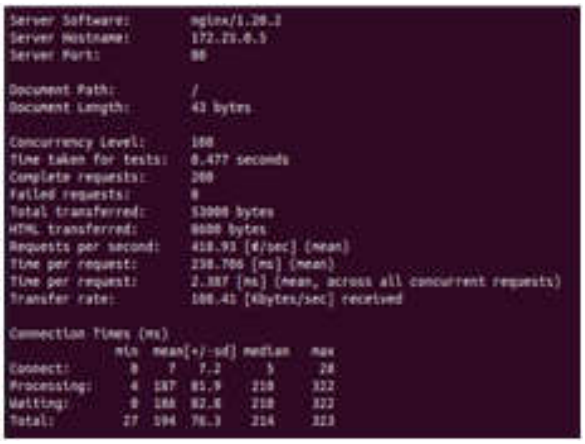

Gambar 8. Pengujian 50 request & 25 concurent

Pengujian kedua menggunakan apache benchmark dilakukan dengan 100 jumlah request dan 50 concurent. dari hasil pengujian tersebut didapatkan 0,202 seconds untuk time taken for test,494,72[sec] (mean) request per second dan 101,068[ms] (mean) time per request. berikut hasil pengujian menggunakan apache benchmark.




Gambar 9. Pengujian 2 Node

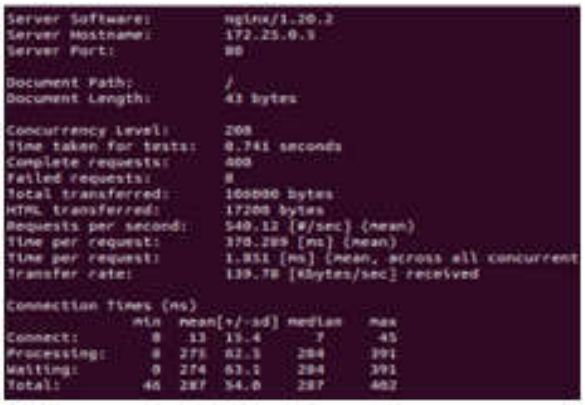
Pengujian ketiga menggunakan apache benchmark dilakukan dengan 200 jumlah request dan 1000 concurent. dari hasil pengujian tersebut didapatkan 0,477 seconds untuk time taken for test418,93[sec] (mean) request per second dan 238,786[ms] (mean) time per request. berikut hasil pengujian menggunakan apache benchmark.




Gambar 10. Pengujian 2 Node

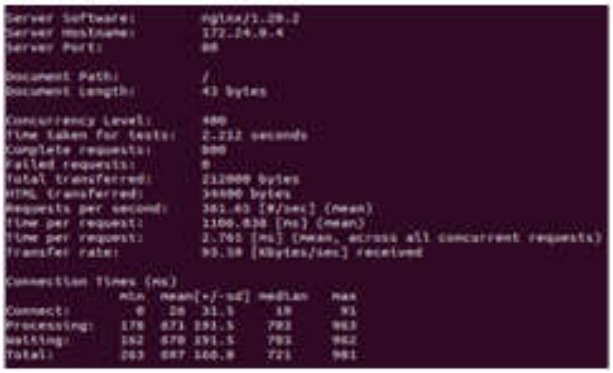
Pengujian ketiga menggunakan apache benchmark dilakukan dengan 400 jumlah request dan 200 concurent.

Dari hasil pengujian tersebut didapatkan 0,741 seconds. Untuk time taken for test, 540,12[sec](mean) request per second dan 370,289[ms](mean) time per request. Berikut hasil pengujian menggunakan apache benchmark.




Gambar 11. Pengujian 2 Node

Pengujian keempat menggunakan apache benchmark dilakukan dengan 800 jumlah request dan 400 concurent. Dari hasil pengujian tersebut didapatkan 2,212 seconds untuk time taken for test 361,65[sec] (mean) request per second dan 1106,038[ms] (mean) time per request. Berikut hasil pengujian menggunakan apache benchmark.




Gambar 12. Pengujian 800request & 400 concurent

Dari pengujian dengan 4 skenario request dan concurrent pada load balancing 2 node. Ringkasan hasil dapat dilihat pada table 1.

Tabel I

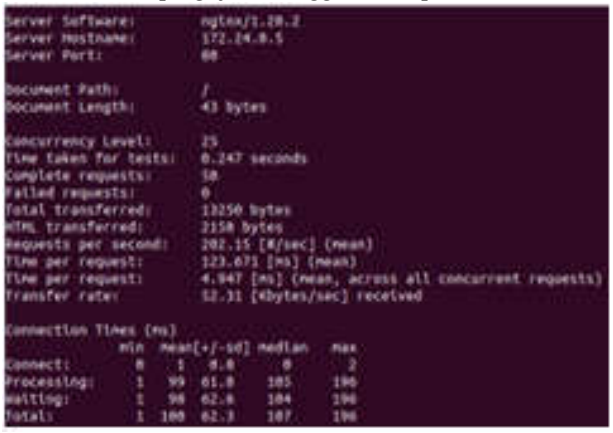
Hasil pengujian load balancing 2 node

| Pengujian | taken for test | request per second | time per request |
|---|---|---|---|
| 50 request 25 concurent | 0,184 seconds | 272,04[sec](mean) | 91,898[ms](mean) |
| 100 request 50 concurent | 0,202 seconds | 494,72[sec](mean) | 101,068[ms](mean) |
| 200 request 100 concurent | 0,477 seconds | 418,93[sec](mean) | 238,786[ms](mean) |
| 400 request 200 concurent | 0,741 seconds | 540,12[sec](mean) | 370,289[ms](mean) |
| 800 request 400 concurent | 2,212 seconds | 361,65[sec](mean) | 1106,038[ms](mean) |
| Rata-rata | 0,7632 seconds | 417,492[sec](mean) | 381,22[ms](mean) |

2. Pengujian tiga node

Pengujian pertama menggunakan apache benchmark dilakukan dengan 50 jumlah request dan 25 concurent. dari hasil pengujian tersebut didapatkan 0,247 second. untuk time taken for test,202,15 [sec](mean) request per second dan 123,671 [ms](mean) time per request.

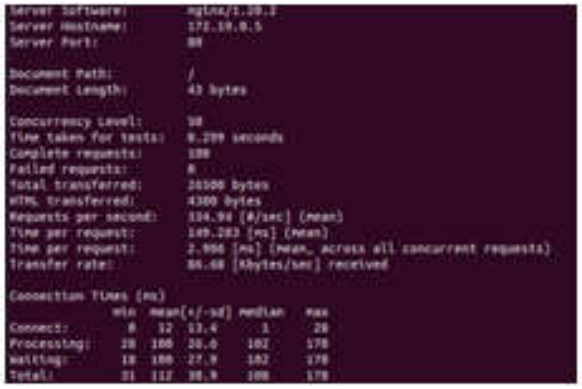
Berikut hasil pengujian menggunakan apache benchmark




Gambar 13. Pengujian 50 request & 25 concuren

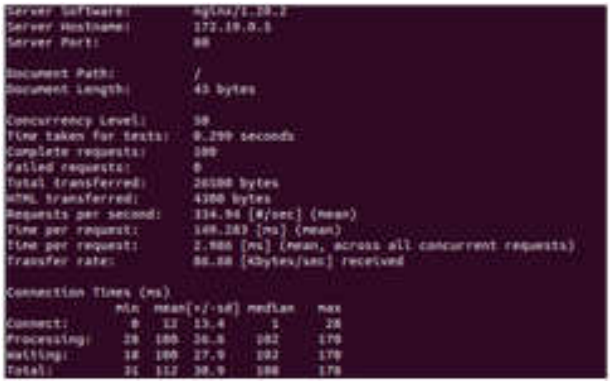
Pengujian pertama menggunakan apache benchmark dilakukan dengan 100 jumlah request dan 50 concurent. dari hasil pengujian tersebut didapatkan 0,299 second. untuk time taken for test,334,94 [sec](mean) request per second dan

149,283[ms](mean) time per request. berikut hasil pengujian menggunakan apache benchmark.



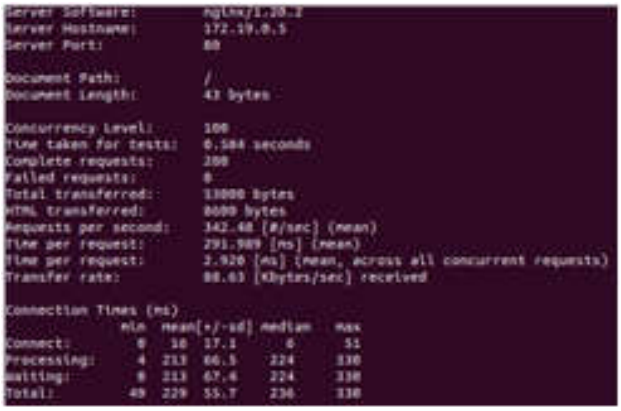

Gambar 14. Pengujian 100 request 50 concurent

Pengujian kedua menggunakan apache benchmark dilakukan dengan 100 jumlah request dan 50 concurent. dari hasil pengujian tersebut didapatkan 0,299 second. untuk time taken for test,334,94 [sec](mean) request per second dan 149,283[ms](mean) time per request. berikut hasil pengujian menggunakan apache benchmark.



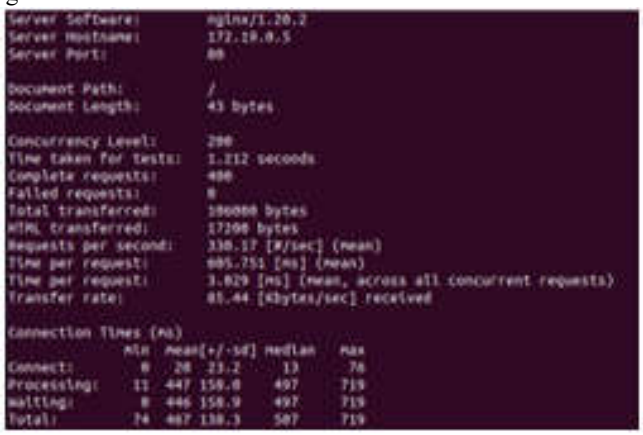

Gambar 15. Pengujian 100 request 50 concurent

Pengujian ketiga menggunakan apache benchmark dilakukan dengan 200 jumlah request dan 100 concurent. dari hasil pengujian tersebut didapatkan 0,584 second untuk time taken for test, request per second 342,48[sec](mean) dan 291,989[ms](mean) time per request. berikut hasil pengujian menggunakan apache benchmark.



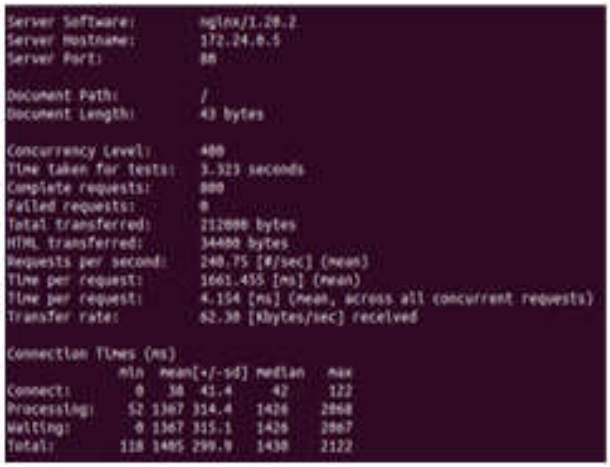

Gambar 16. Pengujian 200 request 100 concurent

Pengujian kempat menggunakan apache benchmark dilakukan dengan 400 jumlah request dan 200 concurrent. dari hasil pengujian tersebut didapatkan 1,212 second. untuk time taken for test,330,17[sec](mean) request per second dan 605,751[ms](mean) time per request. Berikut hasil pengujian menggunakan apache benchmark seperti gambar dibawah.




Gambar 17. Pengujian 400 request 200 concurent

Pengujian kelima menggunakan apache benchmark dilakukan dengan 800 jumlah request dan 400 concurent. dari hasil pengujian tersebut didapatkan 3,323 second untuk time taken for test, request per second 415,4[sec](mean) dan 1661,445[ms](mean) time per request. berikut hasil pengujian menggunakan apache benchmark seperti gambar dibawah.




Gambar 18. Pengujian 800 request 400 concurent

Pada gambar 18 merupakan tampilan dari hasil pengujian dengan 5 skenario request dan concurrent pada load balancing 3 node dapat diakumulasi dalam tabel 2 dibawah ini.

Tabel II
Hasil pengujian load balancing 3 node

| Pengujian | taken for test | request per second | time per request |
|---|---|---|---|
| 50 request 25 concurent | 0,247 second | 202,15 [#/sec](mean) | 123,671 [ms](mean) |
| 100 request 50 concurent | 0,299 | 334,94 [#/sec](mean) | 149,283[ms](mean) |
| 200 request 100 concurent | 0,584 second | 342,48[#/sec](mean | 291,989[ms](mean) |
| 400 request 200 concurent | 1,212 second | 330,17[#/sec](mean) | 605,751[ms](mean) |
| 800 request 400 concurent | 3,323 second | 415,4[#/sec](mean) | 1661,445[ms](mean) |
| Rata-rata | 1,133 seconds | 325,028[#/sec](mean) | 565,809[ms](mean) |

## IV. KESIMPULAN

Dari hasil analisa pengujian terhadap load balancing dengan 2 node dan load balancing 3 node, waktu yang diperoleh untuk melakukan tes dengan 2 node lebih cepat. Untuk request per second, load balancing dengan 3 node memiliki waktu request per detik lebih kecil. Kemudian time per request yang dibutuhkan untuk load balancing 2 node lebih sedikit.

## V. SARAN

Berdasarkan hasil yang telah diperoleh, peneliti memberikan saran untuk ke depannya agar melakukan skenario pengujian terhadap load balancing dengan metode routing selain round robin seperti routing berdasarkan beban, latency sehingga didapatkan analisa performa yang lebih kompleks, dan efektif.

## UCAPAN TERIMA KASIH

Penulis senantiasa mengucapkan rasa syukur yang besar terhadap Tuhan Yang Maha Esa atas segala berkah, hidayah dan juga pertolonganNya, sehingga penulis dapat menyelesaikan program dan juga artikel ilmiah ini dengan baik, Terimakasih penulis haturkan juga terhadap kedua Orang tua yang senantiasa memberi nasihat dan juga support, Dosen Pembimbing Skripsi yang selalu memberikan masukan dan juga saran yang membantu penulis, serta sahabat yang selalu memberikan dorongan dan dukungan dalam melakukan penelitian. Terimakasih juga penulis ucapkan kepada diri sendiri karena telah dapat memegang komitmen dalam melakukan penelitian ini hingga selesai.

## REFERENSI

[1] Djomi1, Manzila Izniardi. Dkk (2018). Analisis Performansi Network Function Virtualization Pada *Containers* Mengunakan *Docker*. *e-Proceeding of Engineering Vol5, No.2*. 1974-1981

[2] Fiddin, Chrisna. Dkk (2018). Performance Analysis Of Container Virtualization Using Docker Under Networked Denial Of Service Attacks. e-Proceeding of Engineering : Vol.5, No.1 281-290.

[3] Nugroho M. Agung , Dkk (2016). Analisis Kinerja Penerapan Container Untuk Load Balancing Web Server Pada Raspberry Pi. Jurnal Ilmiah Penelitian dan Pembelajaran Informatika Vol 01, No 02, 7 – 15.

[4] Douglas Kunda, Sipiwe Chihana, Sinyida Muwanei, "Web Server Performance of Apache and Nginx: A Systematic Literature Review," School of Science Engineering adn Technology, Mulungushi University, PO box 80415 kabwe, Zambia, vol. 18, pp. 2222-2863, 2017

[5] S. E. Prasetyo, "Design and Implementation of Lightweight Virtualization Using Docker Container in Distributing Web Application with Experimental," JITE (Journal of Informatics and Telecomunication Engineering), pp. 270-276, 23 December 2021.

[6] T. Chakraborty, "Docker adn Google Kubernetics," International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), vol. 5, no. 4, pp. 24-35, 2018.

[7] Arun Prasath M., Mrs. T. Sathiyabama, "Virtualization in Cloud Computing," International Journal of Trend in Scientific Research and Development (IJTRSD), vol.2, no.6, pp.2456-6470, 2018