E-ISSN : 2540 - 8984

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Volume 01, Nomor 02, Desember : 7 – 15

ANALISIS KINERJA PENERAPAN CONTAINER UNTUK LOAD BALANCING WEB SERVER PADA RASPBERRY PI

M. Agung Nugroho, M.Kom [1], Rikie Katardi [2]
[1] Teknik Informatika STMIK Akakom Yogyakarta
Jl. Raya Janti No. 143, Karang Jambe, Banguntapan, Bantul, Yogyakarta
[2] Pendidikan Teknologi Informasi STKIP PGRI Tulung Agung
Jl. Mayor Sujadi Tim. No.24, Plosokandang, Kedungwaru, Kabupaten Tulungagung, Jawa Timur
e-mail: m.agung.n@akakom.ac.id[1], rikie@stkippgritulungagung.ac.id[2]

ABSTRAK

Container merupakan teknologi virtualisasi terbaru, dengan container memudahkan system administrator dalam mengelola aplikasi pada server. Docker container dapat digunakan untuk membangun, mempersiapkan, dan menjalankan aplikasi. Dapat membuat aplikasi dari bahasa pemrograman yang berbeda pada lapisan apapun. Aplikasi dapat di bungkus dalam container, dan aplikasi dapat berjalan pada lingkungan apapun dimana saja. Dalam perkembangannya container ini dapat digunakan untuk load balancing, dengan memanfaatkan HA Proxy. Load Balancer dapat digunakan untuk menyelesaikan permasalahan beban kinerja web server yang terlalu berat (overload) terhadap permintaan.Load Balancing merupakan salah satu metode untuk meningkatkan skalabilitas web server sekaligus mengurangi beban kerja web server. Ujicoba dilakukan dengan memberikan beban request pada single container dan multi container, dan membandingkan kinerjanya. Analisis kinerja dapat dilakukan dengan menggunakan parameter performance pada processor, memori dan proses layanan. Penerapan ujicoba dilakukan pada raspberry pi 2.

Diharapkan hasil yang diperoleh melalui ujicoba multi container dengan menerapkan load balancing dapat menunjukkan pembagian beban resource dari masing-masing container sehingga aplikasi dapat berjalan walaupun dengan beban request tinggi. Penelitian ini dapat dikembangkan dengan menambahkan jumlah container atau mengganti variabel untuk pengukuran.

***Kata kunci** : docker, container, raspberry pi, load balancing*

ABSTRACT

Container is new virtualization technology. The system adminitrator has easy to maintain apps in server with container. Docker container can build, ship, and run your application. The application can use different programming language without layer. Application have encapsulated in container and also running on different environment. The development of container used to load balancing application on server with HAProxy. Load balancer can solve performance problem that came from overload request in webserver, application scalable, and reduces webserver load. This research uses 2 different scenario, single container and multi container. Performance analyze based on processor, memori, and load services. The research uses raspberry pi as server and the result is that multi container can be use for load balancing application, also showing the performance raspberry pi.

The result of research is a data that show how to multi container could balancing a resources, in order to make application run smoothly even requests is big. The research can develop with increase number of container and/or change the variable for measuring the performances.

***Keywords**: docker, container, raspberry pi, load balancing*

I. PENDAHULUAN

Infrastruktur layanan perlu di desain dengan memiliki standar availability dan minim gangguan sehingga layanan dapat berjalan dengan optimal dengan minimal layanan tidak dapat diakses (downtime). Layanan cloud dapat menanggani banyak permintaan request dalam satu waktu. Ketika sebuah sistem mengalami kenaikan jumlah request sampai ribuan per hari dapat menyebabkan kinerja sistem menjadi sangat lambat karena kelebihan beban (overload), hal ini dapat menimbulkan banyak permasalahan, salah satunya adalah keluhan dari sisi pengguna [1].

Penggunaan server dengan sistem terdistribusi membutuhkan suatu metode agar dapat membagi beban dengan merata disetiap server. Berbagai penelitian telah dilakukan untuk mengatur pembagian beban kerja pada server klustering agar dapat mengoptimalkan kinerja sistem yaitu dengan menerapkan metode load balancing. Penerapan load balancing dalam web server sangat penting dan dapat menjadi solusi dalam menangani beban

server yang sibuk sehingga dapat meningkatkan skalabilitas pada sistem terdistribusi [2].

Konsolidasi server adalah sebuah proses dalam melakukan sentralisasi beban kerja komputasi untuk mengurangi biaya, kompleksitas, jalur komunikasi, manajemen biaya dan untuk mengoptimalisasi dan simplikasi infrastruktur IT yang sedang berjalan dan untuk menghasilkan solusi investasi dan implementasi [3]. Lebih detail lagi konsolidasi server bertujuan untuk meningkatkan efisiensi pemanfaatan sumber daya komputer dengan salah satu caranya yaitu mengurangi jumlah server. Dengan konsolidasi server maka semua fungsi yang sebelumnya ditangani oleh beberapa server yang berbeda akan ditangani oleh sebuah server dengan kapasitas yang lebih besar. Namun dalam pengimplementasian konsolidasi server ini membutuhkan konfigurasi yang kompleks sehingga cukup menyulitkan bagi yang baru memulai menerapkannya. Salah satu solusi dalam mengatasinya adalah menggunakan virtualisasi server karena mudah untuk diimplementasikan [4].

Tidak seperti virtualisasi mesin, docker container tidak menggunakan hardware untuk virtualisasi. Program berjalan dalam docker container berhubungan langsung dengan linux kernel pada host operating system. Karena tidak ada tambahan lapisan antara program yang berjalan didalam container dengan sistem operasi pada komputer, sehingga tidak ada resources yang habis karena redudansi aplikasi atau simulasi virtual hardware. Container memungkinkan mengisolasi lingkungan program, sehingga program dapat berjalan tanpa gangguan dari permasalahan di sistem operasi. Dalam perkembangannya, kemudahan pengelolaan container menjadi basis untuk melakukan skalabilitas [5].

Load balancing dilakukan untuk menangani kelebihan beban pada aplikasi, ketika beban tersebut menghabiskan seluruh resource yang ada, maka aplikasi akan mengalami kegagalan atau down. Container memiliki keunggulan dari sisi kemudahkan deployment, maintenance, dan ringan, sehingga sangat mungkin untuk melakukan proses load balancing. Fokus utama dalam penelitian ini adalah melakukan ujicoba terhadap sebuah aplikasi, dengan mengukur tingkat maksimum kemampuannya pada sebuah container di raspberry pi, lalu membandingkan dengan aplikasi yang berjalan dengan konsep multi container dan menerapkan load balancing.

II. TINJAUAN PUSTAKA

A. Web Server

Web server merupakan software yang memberikan layanan data yang berfungsi yang menerima permintaan http ataupun https (hypertext transfer protocol security) dari client melalui web browser dan mengirimkan kembali hasilnya dalam bentuk halaman web yang umumnya berbentuk dokumen dalam format HTML.

B. Docker

Docker adalah 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 di bawah Lisensi Apache Versi 2.0 yang bisa dipergunakan secara gratis. Docker menggunakan arsitektur client-server. Docker client menghubungi Docker daemon, yang melakukan pekerjaan berat, menjalankan, dan mendistribusikan Docker container anda. Kedua Docker client dan daemon dapat berjalan pada sistem yang sama. Docker client dan daemon berkomunikasi via sockets atau lewat API yang disediakan Docker [6].

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 untuk mempercepat proses deployment, optimalisasi aplikasi dan isolasi. Platform ini dapat digunakan untuk membangun, mempersiapkan, dan menjalankan aplikasi. Dapat membuat aplikasi dari bahasa pemrograman yang berbeda pada lapisan apapun. Aplikasi dapat di bungkus dalam container, 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 [7].

C. Node JS

NodeJS adalah sebuah platform untuk membangun real-time application. NodeJS dapat menangani event input-output server, dengan kata lain NodeJS dapat memungkinkan para developer Javascripts untuk membuat event-driven servers dalam JavaScript.

E-ISSN : 2540 - 8984

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Volume 01, Nomor 02, Desember : 7 – 15

D. Load Balancing

Load balancing adalah sebuah metodologi jaringan komputer untuk mendisitribusikan beban kerja di beberapa komputer atau cluster komputer untuk mencapai pemanfaatan optimal sumber daya, memaksimalkan throughput meminimalkan waktu respon dan menghindari kelebihan beban. Load balancing merupakan penyeimbang beban kepada dua atau lebih server dan salah satu alasan utama cluster server.

E. Raspberry Pi

Raspberry Pi, sering disingkat dengan nama Raspi, adalah komputer papan tunggal (single-board circuit; SBC) yang seukuran dengan kartu kredit yang dapat digunakan untuk menjalankan program perkantoran, permainan komputer, dan sebagai pemutar media hingga video beresolusi tinggi.

III. PERANCANGAN PENELITIAN

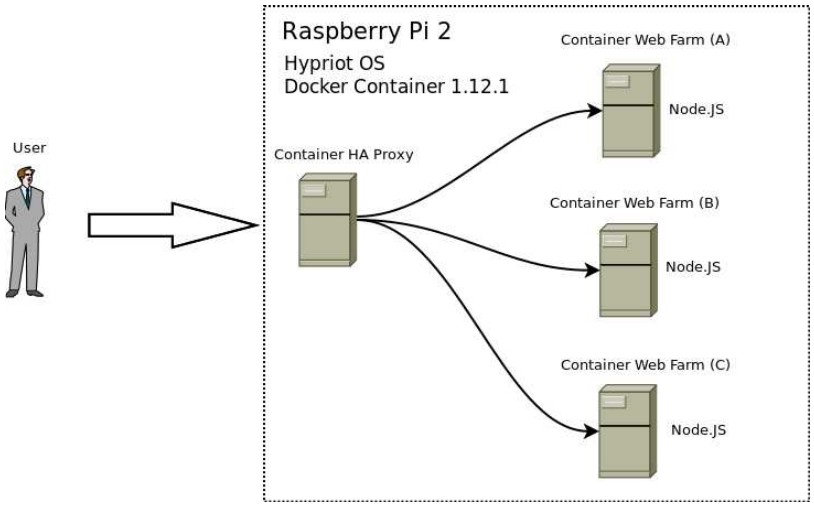
A. Deskripsi Umum

Pada penelitian ini melakukan eksperimen container docker. Platform ini dapat digunakan untuk membangun, mempersiapkan, dan menjalankan aplikasi. Dapat membuat aplikasi dari bahasa pemrograman yang berbeda pada lapisan apapun. Aplikasi dapat di bungkus dalam container, 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.

B. Perancangan Penelitian

Docker awalnya di jalankan pada single container, dan menjalankan aplikasi berbasis node.js. Kemudian dilakukan analisis performance dari kinerja apps pada container tersebut. Untuk performance analisis tersebut, peneliti menggunakan apache benchmark, yang dapat dimodifikasi untuk memberikan request dalam jumlah tertentu pada layanan yang akan dianalisis. Setelah mengetahui hasil dari single container, container tersebut kemudian di balancing menjadi 3 container, dengan balancer menggunakan HA Proxy. 1 container digunakan untuk menjadi master dalam proses load balancing. User dapat mengakses layanan dari mana pun, tetapi request dari user tersebut, kemudian akan dibagikan ke 3 container web farm yang berfungsi sebagai balancer.

Dalam mempersiapkan rancangan tersebut, beberapa tahapan diperlukan seperti instalasi, persiapan container, dan persiapan multi container web apps.



Gambar. 1. Rancangan load balancing menggunakan docker container pada raspberry pi

1). Instalasi Sistem Operasi

Untuk proses instalasi, Raspberry Pi 2 yang digunakan dalam penelitian ini menggunakan OS Hypriot

E-ISSN : 2540 - 8984

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Volume 01, Nomor 02, Desember : 7 – 15

(Turunan dari Raspbian).

a) Sebelum instalasi, terlebih dahulu download image OS di https://downloads.hypriot.com/hypriotos-rpi-v1.0.0.img.zip
b) Instalasi OS Raspberry Pi harus menggunakan perangkat laptop atau PC berbeda, karena sistem operasi akan diinstall ke SD Card yang menjadi media penyimpanan dari Raspberry Pi. Salah satu tools yang dapat digunakan adalah dd, dd merupakan tools untuk melakukan overwrite sistem pada media, untuk penggunaannya dengann menambahkan if sebagai input file / image yang digunakan, dan of sebagai output file, biasanya diarahkan ke media penyimpanan yang digunakan. Proses instalasi dapat dilakukan dengan unzip image tersebut dan install di SDCard (dalam contoh ini berada di /dev/sdd) dengan menggunakan perintah di Linux : dd bs=4M if=hypriotos-rpi-v1.0.0.img of=/dev/sdd
c) Kemudian masukkan SD Card kembali pada slot Raspberry 2. Dan nyalakan kembali Raspberry Pi 2.

Proses instalasi hanya berlangsung 10 – 20 menit saja, tergantung dari image yang digunakan. Raspbian merupakan distro linux untuk ARM berbasis Debian sehingga package management tools menggunakan APT. Penggunaan APT akan memberikan kemudahan dalam proses instalasi dan konfigurasi package kebutuhan riset.

2). Perancangan single container docker

Container pada penelitian ini menggunakan docker. Docker adalah light virtualization dan sudah default terinstall pada sistem operasi Hypriot untuk raspberry pi 2. Untuk melihat versi dari docker yang digunakan, dapat menggunakan perintah “docker -v”.

Untuk mempersiapkan rancangan, diperlukan container apps dengan node.js. Apps node.js yang digunakan ini adalah apps sederhana yang digunakan untuk keperluan eksperimen. Untuk eksperimen ini, peneliti hanya menggunakan 2 file index.js dan package.json.

a). Membuat file src/index.js untuk web apps

```
var express = require('express');
var os = require("os");

var app = express();
var hostname = os.hostname();

app.get('/', function (req, res) {
res.send('<html><body>Hello from Node.js container ' + hostname + '</body></html>');
});

app.listen(80);
console.log('Running on http://localhost');
```

b). Membuat file src/package.json.

File ini merupakan depedensi untuk membangun dan menjalankan apps Node.js:

```
{
"name": "node-hello-world",
"private": true,
"version": "0.0.1",
"description": "Node.js Hello world app on docker",
"author": "hypriot.com",
"dependencies": {
"express": "4.12.0"
}
}
```

Proses selanjutnya adalah membuat docker images untuk keperluan node.js apps. Docker images merupakan

sistem operasi yang dilengkapi dengan pustaka-pustaka pengembangan, node.js dan tools pengembang lain. Docker images tersebut kemudian dapat dijalankan sebagai layanan container. Docker images bisa didapatkan melalui dockerhub. Namun untuk memenuhi kebutuhan karena penggunaan Raspberry Pi pada penelitian ini, peneliti membuat sendiri docker images. Untuk membangun docker images tersebut, diperlukan Dockerfile.

Dalam penelitian ini, akan dijalankan aplikasi web di dalam docker container. Untuk membangun container tersebut, peneliti menggunakan Dockerfile.

c). Dockerfile

```
FROM hypriot/rpi-node:0.12.0

ADD src/ /src
WORKDIR /src

RUN npm install

EXPOSE 80

CMD ["node", "index.js"]
```

d). Menjalankan 1 container

Untuk melakukan percobaan web server node.js bekerja, perlu membuat terlebih dahulu docker application images dan menjalankan sebagai container baru.

```
$ docker build -t node-hello .

$ docker run -p 80:80 --name web -d node-hello
```

e) Menguji server berjalan

```
$ curl http://localhost
<html><body>Hello from Node.js container 38f69acbdf13</body></html>
```

As we have published the port 80 to the host we also can access our web application from another machine.

3). Perancangan multi container docker

a). Persiapan tools docker-compose.

Docker-compose memberikan kemudahan untuk mengontrol banyak container sekaligus. Selain itu untuk membuat web farm yang mendukung load balancing, diperlukan HA Proxy yang berjalan dibelakang web server. Untuk menginstall docker-compose pada ARM Processor Raspberry Pi, peneliti mengambil dari repository https://github.com/hypriot/compose

```
$ sudo sh -c "curl -L https://github.com/hypriot/compose/releases/download/1.1.0-raspbian/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose; chmod +x /usr/local/bin/docker-compose"
```

b). Konfigurasi docker-compose

Docker-compose adalah file konfigurasi untuk docker-compose, dapat mengatur container melalui file docker-compose.yml, kemudian menjalakannya dengan perintah docker-compose. Selanjutnya akan dibuat file docker-compose.yml yang mengatur 3 container weba, webb dan webc berbasis *application image* dan 1 container ha proxy.

```
weba:
```

E-ISSN : 2540 - 8984

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Volume 01, Nomor 02, Desember : 7 – 15

```
build: .
expose:
- 80

webb:
build: .
expose:
- 80

webc:
build: .
expose:
- 80

haproxy:
image: hypriot/rpi-haproxy
volumes:
- haproxy:/haproxy-override
links:
- weba
- webb
- webc
ports:
- "80:80"
- "70:70"

expose:
- "80"
    - "70"
```

Setiap web aplikasi (weba, webb, webc) menggunakan port 80. Sebagai tambahan container HAProxy akan diletakkan pada jaringan yang sama dengan ketiga container web aplikasi. Selanjutnya HAProxy akan menjalan kan proses *load balancing* melalui port 80 sebagai *central entry point* untuk aplikasi *webserver farm*.

IV. IMPLEMENTASI DAN PENGUJIAN

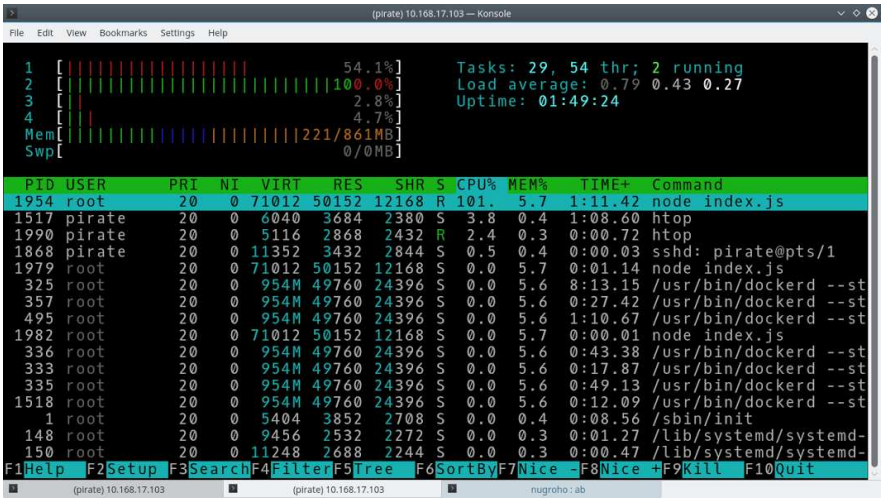
A. Pengujian single container

Dengan menggunakan single container diperoleh penggunaan processor hanya maksimum pada 1 processor saja, sementara penggunaan memori hanya 221 MB.

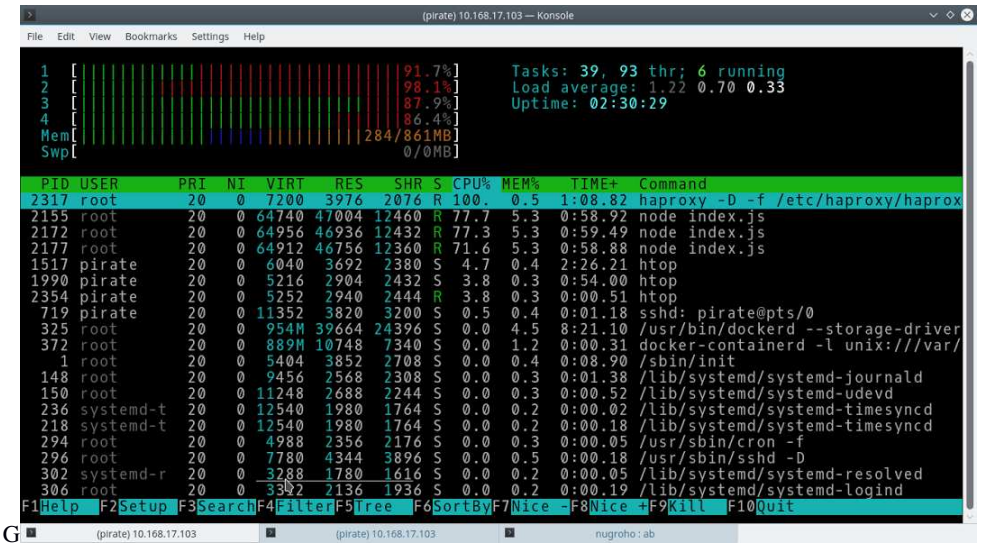
E-ISSN : 2540 - 8984

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Volume 01, Nomor 02, Desember : 7 – 15

G



Gambar 2. Uji performance processor dan memori single container



G

Gambar 3. Uji performance processor dan memori multi container

B. Pengujian multi container

Dengan menggunakan single container diperoleh penggunaan processor hanya maksimum pada 4 processor saja, sementara penggunaan memori hampir sama dengan single container 284 MB.

C. Pengujian beban kerja web server farm

Pada pengujian ini menggunakan apache benchmark dengan request dan proses permintaan yang bisa dimodifikasi. Pengujian ini menggunakan request permintaan 10000 – 20000 dan perulangan permintaan request sampai 40 kali.

1). Langkah pengujian

Menggunakan perintah ab -n (request) -c (perulangan permintaan) (server yang diuji)

$ ab -n 20000 -c 40 http://10.168.17.103/

E-ISSN : 2540 - 8984

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Volume 01, Nomor 02, Desember : 7 – 15

```
This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
Benchmarking 10.168.17.103 (be patient)
Completed 2000 requests
Completed 4000 requests
Completed 6000 requests
Completed 8000 requests
Completed 10000 requests
Completed 12000 requests
Completed 14000 requests
Completed 16000 requests
Completed 18000 requests
Completed 20000 requests
Finished 20000 requests
Server Software:
Server Hostname:        10.168.17.103
Server Port:            80
Document Path:          /
Document Length:        67 bytes
Concurrency Level:      40
Time taken for tests:   58.146 seconds
Complete requests:      20000
Failed requests:        0
Total transferred:      4960000 bytes
HTML transferred:       1340000 bytes
Requests per second:    343.96 [#/sec] (mean)
Time per request:       116.292 [ms] (mean)
Time per request:       2.907 [ms] (mean, across all concurrent requests)
Transfer rate:          83.30 [Kbytes/sec] received
```

2). Hasil pengujian

Dari langkah pengujian yang dilakukan pada single container dan multi container dengan beban kerja ditentukan untuk webserver, maka diperoleh hasil sebagai berikut :

TABEL I
UJI KINERJA BEBAN WEB SERVER

Uji Tahap I	Processor optimum	Memori	Beban request	Request / sec	Time / request	Concurrentcy level
Single container	1	221 MB	10000	249/sec	40 .501 ms	10
Multi container	4	284 MB	10000	343.96/sec	110.30 ms	10
Uji Tahap II	Processor optimum	Memori	Beban request	Request / sec	Time / request	Concurrentcy level
Single container	1	871 MB	20000	300.01/sec	80 .501 ms	40
Multi container	4	300 MB	20000	343.96/sec	116.292 ms	40

E-ISSN : 2540 - 8984

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Volume 01, Nomor 02, Desember : 7 – 15

Berdasarkan informasi tabel 1, single container dapat menangani 10.000 – 20.000 request yang dibebankan oleh server dan mampu menangani sampai beban mencapai titik optimum sebanyak 40x perulangan. Single container terus mengalami kenaikan dari posisi awal memori 221 MB menjadi optimum memori 871 MB pada raspberry pi, namun hanya optimum pada 1 processor saja. Sementara dengan menggunakan multi container, optimal processor digunakan bisa menjadi 4 processor, dan memori stabil. Multi container juga dapat menerima request lebih banyak dan mampu membagi kerja sehingga walaupun beban kinerja tinggi, kestabilan memori dan processor membuat sistem tetap berjalan baik.

V. KESIMPULAN

Hasil yang diperoleh melalui ujicoba multi container dengan menerapkan load balancing dapat menunjukkan pembagian beban resource seperti memori, dan processor dari masing-masing container. Single container hanya mampu menahan beban request yang sama dengan multi container, namun hanya optimal di salah satu core processor saja, sehingga beban tersebut mengakibatkan server aplikasi down (tidak dapat diakses). Dengan penerapan load balancing pada multi container, beban terdistribusi merata ke seluruh core processor, sehingga beban resources tidak mempengaruhi performance dari server aplikasi.

Penelitian ini dapat dikembangkan lagi dengan penambahan cluster container, atau dengan menggunakan orchestration tools swarm atau kubernetes. Variabel pengujian dapat diperluas menjadi parameter I/O storage, bandwidth, dan parameter lain yang memungkinkan.

DAFTAR PUSTAKA

[1] Alimuddin, “Peningkatan Kinerja Siakad Menggunakan Metode Load Balancing dan Fault Tolerance di Jaringan Kampus Universitas Halu Oleo", Thesis, Jurusan Teknik Elektro dan Teknologi Informasi, UGM, Yogyakarta, Indonesia, 2015.
[2] Lukitasari, D., dan Oklilas, A.F. Analisis Perbandingan Load Balancing Web Server Tunggal Dengan Web server Cluster Menggunakan Linux Virtual Server. Jurnal Generic, Vol.5 No.2:2010., ISSN: 1907-4093. Fakultas Ilmu Komputer Universitas Sriwijaya.
[3] Dutta, S., and Mia, I., “Global Information Technology Report 2009-2010: ICT for Sustainability.”, World Economic Forum, 2010.
[4] Eisen, M., “Introduction to Virtualization.”, The Long Island Chapter of the IEEE Circuits and Systems (CAS) Society, 2011.
[5] Nickoloff, J., "Docker in Action", Manning Publication co, 1st edition, Shelter Island, New York, 2016, page 5-7.
[6] Tihfon K.J. Kim and N. Joukov (eds.), Information Science and Applications (ICISA) 2016, Lecture Notes in Electrical Engineering 376, DOI: 10.1007/978-981-10-0557-2_126
[7] Cereghetti, A.N.P, 2012. Global evaluation of CDNs performance using PlanetLab. Thesis. Master in Science in Telecommunication Engineering and Management. Universitat Politecnica Catalunya.