MODUL PRAKTIKUM IMPLEMENTASI *LOAD BALANCE* PADA *WEB SERVER* DENGAN ALGORITMA *ROUND ROBIN* MENGGUNAKAN *BANANA PI*

Candra Setiya Fajriawan[(1)], Rachmad Saptono[(2)], Nugroho Suharto[(3)]
[123] Program Studi Jaringan Telekomunikasi Digital, Jurusan Teknik Elektro, Politeknik Negeri Malang
Jl. Soekarno Hatta No. 9 Malang, Telp : (0341)-404424 / 404425, Fax : (0341)-40420
candrafajriawan@gmail.com

ABSTRAK

Web server merupakan perangkat yang digunakan untuk menerima *request* berupa halaman *web* melalui protokol *HTTP* atau *HTTPS* dari klien yang kemudian hasil dari *request* tadi akan direspon ke dalam bentuk halaman-halaman *web* berbentuk dokumen *HTML* [1].

Situs *web* yang memliki tingkat kunjungan yang tinggi akan memiliki beban akses yang besar, maka performa dan kinerja dari *web server* tersebut dituntut maksimal. Jika *web server* tersebut menggunakan topologi *server* tunggal maka akan membuat kerja dari *web server* semakin berat dalam menerima *request* tersebut, sehingga muncul permasalahan seperti *web server* yang bersangkutan crash ataupun down akibat dari banyaknya permintaan *request* dari klien.

Pada skripsi ini menerapkan sebuah metode *load balance* pada *web server*. Metode *load balance* merupakan teknik penggabungan sekelompok *server* agar bekerja bersama-sama untuk membagi beban yang diberikan oleh klien yang kemudian diterima oleh sekelompok *server* tersebut. Dan algoritma yang digunakan adalah algoritma round robin dimana algoritma round robin merupakan algoritma dengan penjadwalan membagi *request* sama rata ke *web server* yang ada.

Dari hasil pengujian didapatkan hasil waktu respon *load balance* lebih lambat dibandingkan dengan *web server* tunggal. Jumlah klien maksimum *load balance* tanpa ada *request* loss adalah 200 klien dengan 4000 *request* sedangkan *web server* tunggal 100 klien dengan 2000 *request*. *Request* loss tertinggi pada *load balance* yaitu pada jumlah klien 500 dengan 10000 *request* sebesar 4,058% sedangkan *web server* tunggal pada jumlah klien 500 dengan 10000 *request* sebesar 10,622%. Nilai *throughput load balance* dan *web server* tunggal memiliki nilai rata-rata yang sama, namun *load balance* mampu memeratakan *throughput* pada semua klien dan *web server* tunggal tidak mampu melakukan hal tersebut. Perangkat *Banana Pi* yang digunakan sebagai *web server* mampu menangani simulasi serangan *DoS* pada protokol HTTP dengan nilai maksimum penggunaan *CPU* sebesar 52% sedangkan dengan menggunakan *load balance* yang membuat 2 buah *web server* bekerja bersama-sama mampu mengurangi penggunaan *CPU* yaitu sebesar 19%.

Kata Kunci : *Load Balance, Round robin, Banana Pi,* Modul Praktikum

I. Pendahuluan

1.1 Latar Belakang

Berbagai macam teknologi *server* kini hadir demi memudahkan masyarakat dalam mencari informasi yang mereka inginkan. Teknologi *server* yang dimaksud antara lain adalah *ftp server, mail server, proxy server, dhcp server, web server dan dns server. Web Server* merupakan perangkat yang digunakan untuk menerima *request* berupa halaman *web* melalui protokol *HTTP* atau *HTTPS* dari klien yang kemudian hasil dari *request* tadi akan direspon ke dalam bentuk halaman-halaman *web* berbentuk dokumen *HTML* [1]. Situs *web* yang memliki tingkat kunjungan yang tinggi akan memiliki beban akses yang besar, maka performa dan kinerja dari *web server* tersebut dituntut maksimal. Jika *web server* tersebut menggunakan jenis *server* tunggal maka akan membuat kerja dari *web server* semakin berat dalam menerima *request* tersebut, sehingga muncul permasalahan seperti *web server* yang bersangkutan *crash* ataupun *down* akibat dari banyaknya permintaan *request* dari klien [11].

Permasalahan tersebut bisa diatasi dengan cara pembagian beban pada sekelompok *server* atau *load balance server*. *Load balance server* merupakan teknik penggabungan sekelompok *server* agar bekerja bersama-sama untuk membagi beban yang diberikan oleh user yang kemudian diterima oleh sekelompok *server* tersebut [11].

Dalam penerapan *load balance server*, diterapkan algoritma yang efektif untuk melakukan pembagian beban yang diterima oleh sekelompok *server*, algoritma yang tepat adalah algoritma *round robin*. Algoritma penjadwalan *round robin* akan memperlakukan semua *server* dengan sama tanpa memperhatikan kondisi dari *server* itu sendiri. Pada penelitian sebelumnya, telah dilakukan simulasi perbandingan 2 algoritma yaitu Round Robin dan Least Connection. Simulasi pada penelitan tersebut mengggunakan LVS via NAT. Hasil dari simulasi tersebut menyimpulkan bahwa *load balance* menggunakan algoritma Round Robin lebih handal dalam mengoptimalkan throughput, *CPU* Utilization, dan waktu respon dari web server jika

dibandingkan dengan menggunakan algoritma Least Connection [7].

Dalam implementasinya, biaya yang dikeluarkan tidak sedikit, itu dikarenakan jumlah *server* yang dibutuhkan minimal dua atau lebih dan harga dari sebuah PC juga tidak murah. Maka dari itu dalam penelitian ini, akan menggunakan *mini PC* yaitu Banana Pi. Meskipun memiliki ukuran yang kecil dan spesifikasi yang berbeda dengan sebuah PC, namun Banana Pi memiliki kapabilitas untuk menjadi sebuah *server* khususnya *web server*.

Sedangkan dalam pembelajaran khususnya pada jaringan komputer, teknik *load balance* untuk mengatasi *request* tinggi yang akan diterima oleh *server* masih jarang ditemui. Oleh karena itu, di penelitian ini akan dibuat modul praktikum untuk menjelaskan bagaimana mengimplementasikan teknik *load balance* pada sekelompok *web server* menggunakan algoritma *round robin* dan parameter yang dianalisa adalah *throughput*, waktu respon, dan Beban *CPU* dengan jumlah klien yang berbeda-beda. Maka judul yang diambil untuk penelitian ini adalah "Modul Praktikum Implementasi *Load balance* Pada *Web Server* Dengan Algoritma *Round robin* Menggunakan Banana Pi".

1.2 Rumusan Masalah

Berdasarkan latar belakang, dapat dirumuskan perumusan masalah sebagai berikut:

1. Bagaimana membuat modul praktikum implementasi *load balance* pada *web server* dengan algoritma *round robin* menggunakan *banana pi*?
2. Berapa nilai *throughput*, waktu respon, *request loss* pada sisi klien saat sebelum dan sesudah menerapkan *load balance*?
3. Bagaimana rasio penggunaan *CPU* pada *web server* satu dan *web server* dua saat *load balance* diterapkan?

1.3 Tujuan Penelitian

Tujuan penelitian ini adalah sebagai berikut:

1. Membuat modul praktikum implementasi *load balance* pada *web server* dengan algoritma *round robin* menggunakan *banana pi*.
2. Mengetahui nilai *throughput*, waktu respon, *request loss* pada sisi klien saat sebelum dan sesudah menerapkan *load balance*.
3. Mengetahui rasio penggunaan *CPU* pada *web server* satu dan *web server* dua saat *load balance* diterapkan.

1.4 Batasan Masalah

Batasan pada penelitian ini adalah:

1. Jumlah *server* yang digunakan 3 *server* meliputi 2 *server* sebagai *web server* dan 1 *server* sebagai *load balance*.
2. Algoritma yang digunakan adalah algoritma *round robin.*
3. Sistem operasi yang digunakan pada *server* adalah raspbian versi 4.0
4. Jaringan yang digunakan adalah jaringan lokal.
5. *Software* yang digunakan untuk melakukan *load balance* adalah *Haproxy*.
6. *Software* yang digunakan untuk *web service* adalah *Nginx.*
7. *Software* yang digunakan untuk *file syncronize* adalah *Rsync.*
8. *Software* yang digunakan untuk pengujian pada sisi klien adalah *Web Server Stress Tool.*
9. *Software* yang digunakan untuk pengujian pada sisi *server* adalah *DoSHTTP.*

1.5 Manfaat Penelitian

Manfaat dari membuat modul praktikum implementasi *load balance* pada *web server* dengan algoritma *round robin* menggunakan *banana pi* adalah sebagai berikut:

1. Memberikan pengetahuan kepada pembaca berupa modul praktikum tentang bagaimana cara mengimplementasikan *load balance* pada *web server* dengan algoritma *round robin*.
2. Memberikan tingkat ketersediaan yang tinggi pada klien saat mengakses sebuah situs *web* pada saat web memiliki trafik yang sangat padat.

II. Dasar Teori

2.1 Penelitian Sebelumnya

Pada jurnal 'Analisis Kinerja Algoritma *Load balance*r dan Implementasi pada Layanan *Web*' [5], dimana Yogi Kurniawan mengajukan sebuah penelitian tentang membandingkan kinerja dan efisiensi algortima penjadwalan *round robin*, *least connection, weighted round robin*, dan *weighted least connection* pada pembagi beban berdasarkan TCP/IP yang diimplemntasikan pada *web server*. Berdasarkan penelitian ini, pada konfigurasi *server* yang sama penggunaan algoritma least connection memiliki peforma yang lebih baik pada 3000 *request*/detik dan 6000 *request*/detik akan tetapi memiliki peforma yang setara dengan *round robin* pada 9000 *request*/detik dengan layanan *web* dinamis, dan pada konfigurasi *server* berbeda weighted least connectionmemiliki peforma yang setara dengan algoritma least connectionakan tetapi pembagian beban lebih adil terhadap tiap *server*.

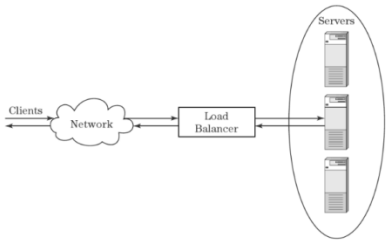
Pada jurnal 'Implementasi dan Analisis Kinerja *Load balance* Pada Virtual *Server* Menggunakan Zen *Load balance*r' [3], dimana Radiv Herdian mengajukan sebuah penelitian tentang menggunakan Zen *Load balance*r untuk mengatur distribusi pengolahan data ke beberapa *server*. Pada penelitian ini telah diimplementasikan *load balance* pada virtual *server*. Didapatkan nilai *throughput* sebesar 13202 Kbps, dan jumlah layanan per-detik dengan nilai 1592 *request per-second*. Terjadi penurunan

65,527% penggunaan *CPU* virtual *server* pada skenario *round robin* dan terjadi penurunan 20,7124% penggunaan *CPU* real *server* pada skenario weighted. Pada skenario failover didapatkan nilai rata-rata *down*time sebesar 9622 ms. Nilai *throughput* LVS mencapai 11916 Kbps, dan LVS dapat melayani maksimal 1321 *request* per-second.

2.2 Landasan Teori

2.2.1 *Load balance Server*

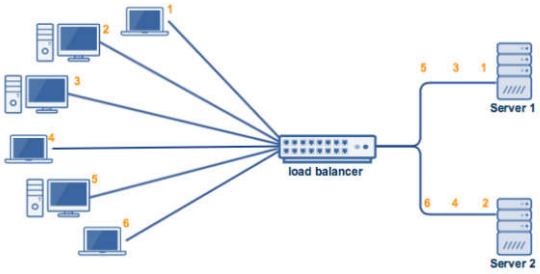
Load balance server adalah teknik yang diterapkan pada *server* untuk mendistribusikan beban kerja pada dua atau lebih pada *server* secara seimbang agar pekerjaan dapat berjalan secara optimal dan tidak overload beban pada salah satu *server*.



Gambar 2.1 *Load balance Server* [12]

2.2.2 Algoritma *Round robin*

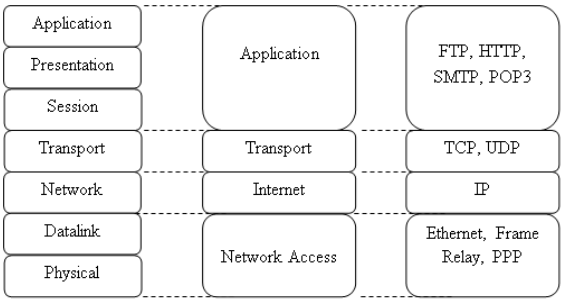
Algoritma *round robin* melakukan metode pendistribusiannya yaitu mendistribusikan *request* klien sama rata ke seluruh backend *server* tanpa mempedulikan kapasitas *server* ataupun beban *request* yang diterima oleh *server* [13].



Gambar 2.2 Ilustrasi Penjadwalan Algoritma *Round robin* [14].

2.2.3 Arsitektur dan Protokol Jaringan TCP/IP

Dalam arsitektur jaringan komputer, terdapat suatu lapisan-lapisan (layer) yang memiliki tugas spesifik serta memiliki protokol tersendiri. ISO (International Standard Organization) telah mengeluarkan suatu standard untuk arsitektur jaringan komputer yang dikenal dengan nama Open System Interconnection (OSI). Standard ini terdiri dari 7 lapisan protokol yang menjalankan fungsi komunikasi antara 2 komputer. Dalam TCP/IP hanya terdapat 5 lapisan sebagai berikut :



Gambar 2.3 Perbandingan Arsitektur OSI dan TCP/IP [15]

2.2.4 *Web Server*

Web Server merupakan perangkat yang digunakan untuk menerima *request* berupa halaman *web* melalui protokol HTTP atau HTTPS dari klien yang kemudian hasil dari *request* tadi akan direspon ke dalam bentuk halaman-halaman *web* berbentuk dokumen HTML.

2.2.5 Banana Pi

Banana Pi merupakan perangkat single-board computer yang diproduksi oleh Shenzhen LeMaker Technology Co.,LTD dengan desain yang terinspirasi dari Raspberyy Pi.

2.2.6 HAProxy

HAProxy adalah tool yang digunakan untuk membangun sebuah high availability dan *load balance* untuk aplikasi yang berbasis TCP dan HTTP.

2.2.7 Rysnc

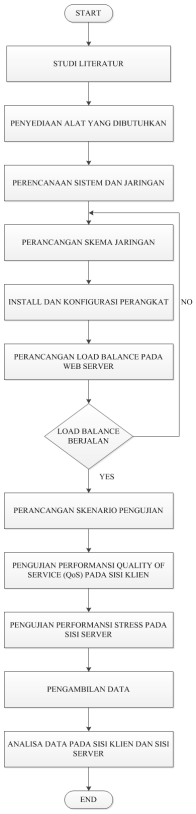
Rsync adalah tool yang ada pada sistem operasi open source yang digunakan untuk syncronize file melalui jaringan secara satu arah.

2.2.8 Nginx

Nginx adalah aplikasi *web* service yang ditujukan untuk mendirikan sebuah *web server*.

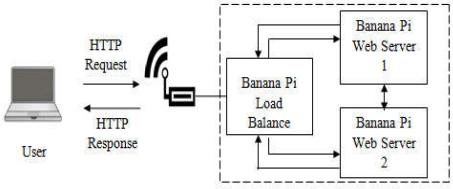
III. Metode Penelitian

3.1 Tahapan Penelitian



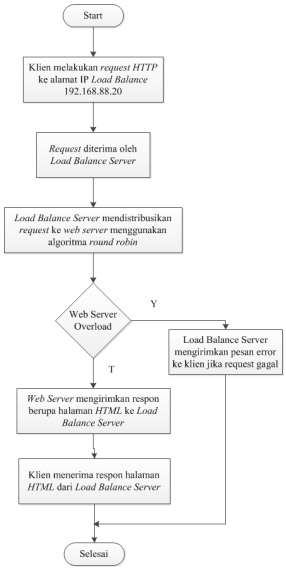
Gambar 3.1 Flowchart Tahapan Penelitian

3.2 Perencanaan Sistem



Gambar 3.2 Diagram Blok Rancangan Sistem

3.3 Flowchart Perencanaan Sistem



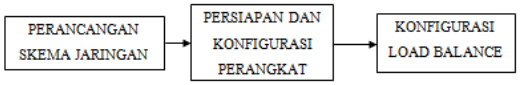
Gambar 3.3 Flowchart Perencanaan Sistem

3.4 Bahan dan Alat Penelitian

No	Nama Bahan	Keterangan
1	*HAProxy*	Aplikasi *Load balance*
2	*Nginx*	Aplikasi *Web Server*
3	*Web Server Stress Tool*	Aplikasi pengujian *web server* di sisi klien
4	*DosHTTP*	Sebagai aplikasi pengujian *web server* di sisi *server*
5	*Rsync*	Aplikasi sinkronisasi file pada *webserver*
No	**Nama Alat**	**Keterangan**
1	*Banana Pi*	Perangkat *Load balance* dan *Web Server* 1 dan 2

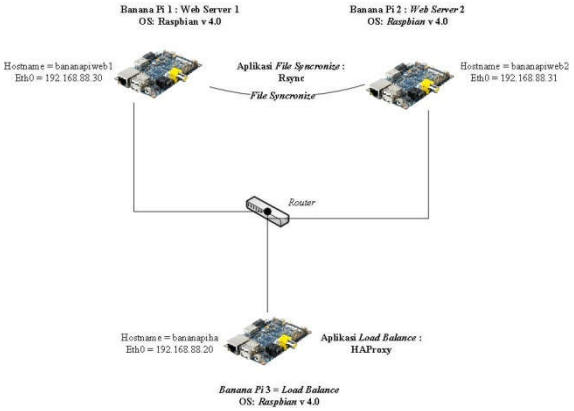
IV. Perancangan Sistem

4.1 Blok Diagram Perancangan Sistem



Gambar 4.1 Blok Diagram Perancangan Sistem

4.2 Perancangan Skema Jaringan



Gambar 4.2 Perancangan Skema Jaringan

4.3 Persiapan dan Konfigurasi Perangkat

Persiapan yang dimaksud yaitu pertama menghubungkan semua perangkat yaitu bananapi *load balance*, bananapi *web server* 1 dan *web server* 2 dan router mikrotik. Yang kedua yaitu install sistem operasi pada bananapi *load balance*, bananapi *web server* 1 dan *web server* 2. Ketiga yaitu melakukan update repository pada perangkat bananapi *load balance*, bananapi *web server* 1 dan *web server* 2. Keempat pemberian alamat IP dan hostname pada bananapi *load balance*, bananapi *web server* 1 dan *web server* 2. Kelima adalah instalasi software yang digunakan pada bananapi *load balance*, bananapi *web server* 1 dan *web server* 2. Keenam adalah konfigurasi software yang digunakan pada bananapi *load balance*, banana pi *web server* 1 dan *web server* 2.

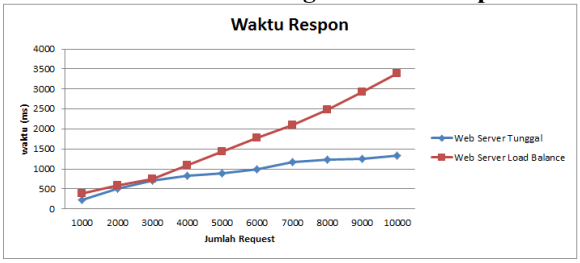
V. Hasil dan Pembahasan

5.1 Perancangan Skenario Pengujian

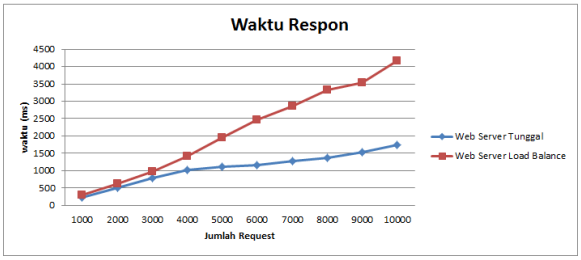
Untuk mengukur kinerja dari *load balance* maka dirancang skenario pengujian dengan melakukan perbandingan antara *web server* dengan *load balance* dan tanpa menggunakan *load balance* via kabel dan via wireless dengan menggunakan aplikasi yang digunakan untuk melakukan pengujian. Pengujian dilihat dari 2 sisi yaitu dari sisi klien dengan mengamati tingkat waktu respon, *request loss* dan *throughput* menggunakan software pengujian *Web Server* Stress Tool dan dari sisi *server* dengan mengamati tingkat beban *request* yang diterima menggunakan software DoSHTTP.

5.5 Grafik Perbandingan *Web Server* Tunggal dan *Web Server Load balance*

5.5.1 Grafik Perbandingan Waktu Respon



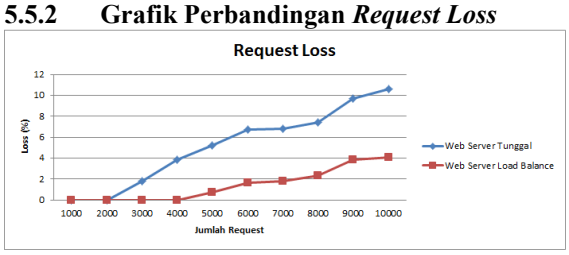
Gambar 5.8 Grafik Perbandingan Waktu Respon (ms) Antara *Web Server* Tunggal Dan *Web Server* Dengan *Load Balance* Via Kabel



Gambar 5.9 Grafik Perbandingan Waktu Respon (ms) Antara *Web Server* Tunggal Dan *Web Server* Dengan *Load Balance* Via Wireless

Pada gambar 5.8 dan gambar 5.9 yang merupakan grafik perbandingan waktu respon , pada waktu respon saat menggunakan metode *load balance* dan tanpa menggunakan *load balance*, didapatkan hasil bahwa waktu respon saat tidak menggunakan metode *load balance* lebih cepat dibandingkan saat menggunakan metode *load balance*. Ini dikarenakan saat metode *load balance* diterapkan, alamat ip yang diakses oleh klien adalah alamat ip dari *load balance*, pada saat klien melakukan *request*, klien mengakses alamat ip dari *load balance* kemudian *load balance* akan mendistribusikan *request* ke *web server* baik *web server* satu ataupun *web server* dua. Saat *web server* menerima *request* dari klien, *web server* akan memberikan respon namun respon tersebut dikembalikan terlebih dahulu ke *load balance* yang kemudian *load balance* akan meneruskan respon yang diberikan oleh *web server* ke klien. Hal itu menyebabkan waktu respon memakan waktu lebih lama dibandingkan dengan saat klien langsung mengakses alamat IP dari *web server* tanpa harus melewati *load balance*.

5.5.2 Grafik Perbandingan *Request Loss*

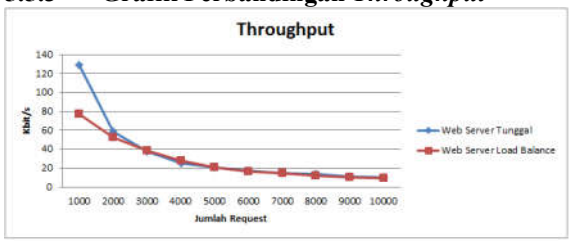


Gambar 5.10 Grafik Perbandingan *Request Loss* (%) Antara *Web Server* Tunggal Dan *Web Server* Dengan *Load Balance* Via Kabel

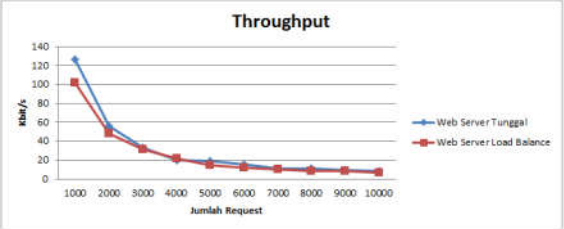
Gambar 5.11 Grafik Perbandingan *Request Loss* (%) Antara *Web Server* Tunggal Dan *Web Server* Dengan *Load Balance* Via Wireless

Dari hasil grafik perbandingan *request loss* sesuai pada gambar 5.10 dan gambar 5.11, nilai *request loss* saat menggunakan *load balance* lebih kecil dibandingkan tanpa menggunakan *load balance*. Itu dikarenakan pembagian *request* yang merata pada masing-masing *web server*, tidak seperti *web server* tunggal yang menangani semua *request* pada sebuah *web server* saja.

5.5.3 Grafik Perbandingan *Throughput*



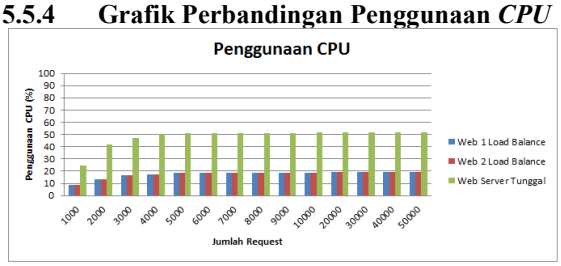
Gambar 5.12 Grafik Perbandingan *Throughput* (Kbit/s) Antara *Web Server* Tunggal Dan *Web Server* Dengan *Load Balance* Via Kabel



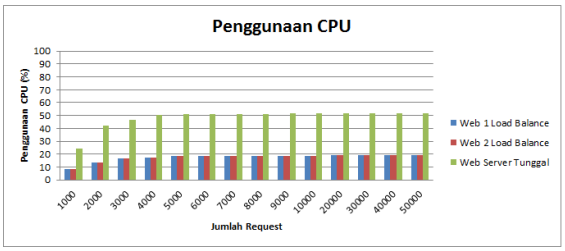
Gambar 5.13 Grafik Perbandingan *Throughput* (Kbit/s) Antara *Web Server* Tunggal Dan *Web Server* Dengan *Load Balance* Via Wireless

Pada gambar 5.12 dan gambar 5.13 yang merupakan grafik perbandingan waktu *throughput*, metode *load balance* mampu memeratakan *throughput* pada saat jumlah klien dan tingkat *request* bertambah (pada lampiran 2 halaman 88). Tidak seperti saat tanpa menggunakan *load balance*, tiap-tiap klien tidak mendapatkan nilai *throughput* yang sama karena semua hanya ditangani oleh satu *web server* saja,

5.5.4 Grafik Perbandingan Penggunaan *CPU*



Gambar 5.14 Grafik Perbandingan Penggunaan *CPU* (%) Antara *Web Server* Tunggal Dan *Web Server* Dengan *Load Balance* Via Kabel



Gambar 5.15 Grafik Perbandingan Penggunaan *CPU* (%) Antara *Web Server* Tunggal Dan *Web Server* Dengan *Load Balance* Via Wireless

Pada gambar 5.14 dan gambar 5.15 yang merupakan grafik perbandingan dari hasil pengujian penggunaan *CPU* pada sisi *server*, didapatkan hasil bahwa dengan menggunakan *load balance* dapat membagi request pada kedua web server yang ada dibandingkan dengan *web server* tunggal yang harus menangani sendiri semua *request* yang datang yang mengakibatkan *web server* tunggal harus bekerja dua kali lebih berat dengan rata-rata penggunaan *CPU* diatas 10000 *request* sebesar 52% dibandingkan dengan *web server* dengan *load balance server* yang bekerja bersama-sama untuk membagi beban *request* yang datang dengan rata-rata penggunaan *CPU* diatas 10000 *request* sebesar 19%.

VI. Kesimpulan dan Saran

6.1 Kesimpulan

Berdasarkan hasil implementasi *load balance* pada *web server* yang telah dilakukan pengujian berdasarkan perancangan skenario pengujian yang telah dibuat, dapat disimpulkan sebagai berikut :

1. Metode *Load balance* pada *web server* mampu memeratakan beban *request* dari klien ke jumlah *web server* yang ada.
2. Algoritma *round robin* melakukan mendistribusikan *request* secara bergiliran ke *web server* yang ada.
3. Dengan menggunakan metode *load balance*, waktu respon lebih lambat dibandingkan dengan *web server* tunggal.
4. Metode *load balance* mampu menyeimbangkan dan mengoptimalkan nilai *throughput* ketika terjadi peningkatan jumlah *request* dari klien.
5. Berdasarkan rekomendasi ITU-T Y.1541 tentang nilai *request loss* yang masih dianggap baik yaitu 0.1%, metode *load balance* pada *web server* mampu memnuhi standar tersebut pada jumlah *request* 5000 sedangkan *web server* tunggal pada jumlah *request* 2000.

6.2 Saran

Untuk pengembangan sistem yang telah dibuat, maka diperlukan saran yang dapat membantu terciptanya proses tersebut, antara lain :

1. Implementasi lebih lanjut pada jaringan internet agar mendapatkan hasil yang lebih nyata.
2. Menggunakan database sebagai tempat penyimpanan berkas-berkas *server*.

Daftar Pustaka

[1] Nancy J. Yeager and Robert E. McGrath. 1996. *Web Server* Technology: The Advanced Guide for World Wide *Web* Information Providers. San Franscisco, California. Hal:19-27.

[2] Handoko Yoga Hartono. 2015. Implementasi *Web Server Load balance* Pada Mesin Virtual. Fakultas Komunikasi dan Informatika. Universitas Muhammadiyah Surakarta.

[3] Ariya Kusuma. 2014. Pembuatan Aplikasi *Load balance Web Server* Berdasarkan IPVSADM. Fakultas Teknik. Universitas Bhayangkara Surabaya.

[4] Radiv Herdian. 2015. Implementasi dan Analisis Kinerja *Load balance* Pada Virtual *Server* Menggunakan Zen *Load balance*r. Teknik Telekomunikasi. Universitas Telkom Bandung.

[5] Muhfi Asbin Sagala. 2010. Implementasi *Load balance* Pada *Web Server*. Fakultas Teknik Elektro. Universitas Sumatera Utara Medan.
[6] Yogi Kurniawan. 2013. Analisis Kinerja Algoritma *Load balance*r dan Implementasi Pada Layanan *Web*. Teknik Informasi Ilmu Komputer. Universitas Brawijaya Malang.
[7] Mohammad Rizky Pratama. 2015. Analisis Performansi *Load balance* Dengan Algoritma *Round robin* Dan Least Connection Pada Sebuah *Web Server*. Teknik Telekomunikasi. Universitas Telkom Bandung.
[8] Anaylisis of *Load balance* Algorithms. World Academy of Science.
[9] Ryan O'Hara. 2014. HAProxy. RedHat.
[11] Tony Bourke. 2001. *Server Load balance*. United States of America.