ANALISA IMPLEMENTASI LOAD BALANCING ROUND ROBIN DAN LEAST CONNECTION PADA WEB SERVER (STUDI KASUS PT UCC)

(Analysis of the Implementation of Load Balancing Round Robin and Least Connection on a Web Server (Case Study of PT.UCC) **)**

Khoif Ekmawan*
Prodi Teknik Informatika, Universitas Mercubuana
Jl. Raya, RT.4/RW.1, Meruya Sel., Kec. Kembangan, Jakarta, Daerah Khusus Ibukota Jakarta 11650

Email: xoifasd@gmail.com
**Penulis Korespondensi*

Abstract

The web server is used at PT UCC to manage and store various information so that its presence must always be there when accessed by its workers both in low and high traffic conditions. Load balancing can be used to overcome slow connections and there is more than 1 server. In this study the author aims to compare the load balancing algorithm round robin and least connection using the nginx web server. The research is implementing the nginx web server because it is known for its opensource nature and has a load balancer function. Due to resource limitations, the server is run on a virtual machine that looks like a physical server. Server testing using httperf tools using throughput and response time measurement parameters. Testing by giving a load via httperf with a rate value of 30, 60 and 120. The results of the load balancing test, namely the least connection algorithm has a better throughput value and response time value than round robin. At a given low connection rate, a single server still has good performance. When providing more connection rates, load balancing systems have better capabilities than single server.

Keywords: *Load balancing, Roundrobin, Least connection, Nginx, Httperf*

1. PENDAHULUAN

Web server digunakan untuk memuat informasi yang di tempatkan pada halaman *web*. Fungsi utama *web server* adalah mentransfer berkas atas permintaan pengguna melalui protokol komunikasi yang telah ditentukan dan mentransfer kedalam sebuah halaman *web*[1]. Pada PT UCC *web server* digunakan untuk mengelola informasi untuk menunjang pekerjaan menjadi lebih mudah. Informasi yang dimuat merupakan informasi yang penting berhubungan dengan jalannya pekerjaan. Setiap karyawan maupun admin yang diberikan hak untuk mengelola informasi, mereka lakukan melalui halaman *web*. Maka dari itu, keberadaan dari *web server* diperlukan untuk mempermudah pengelolaan pekerjaan yang berlangsung. Informasi harus mudah diakses dan selalu ada ketika dibutuhkan. Dengan menggunakan *single server* yang berjalan, membuat koneksi menjadi melambat seiring banyaknya karyawan atau admin yang mengakses halaman *web* pada waktu yang hampir bersamaan. Kemampuan dari *server* perlu ditingkatkan mengingat pekerjaan setiap waktu mengalami peningkatan. Dengan mengetahui adanya permasalahan tersebut, penulis akan melakukan skenario implementasi terhadap algortima load balancing menggunakan *web server nginx*.

Load balancing merupakan metode jaringan komputer untuk mendistribusikan beban kerja pada seluruh beberapa sumber daya komputasi, seperti komputer, *cluster* komputer, tautan jaringan, unit pemrosesan pusat atau *drive disk*[2]. Penyeimbang beban meningkatkan distribusi beban kerja di berbagai sumber daya komputasi. *Load balancing* akan membagi jumlah pekerjaan yang harus dilakukan komputer antara dua atau lebih komputasi sehingga lebih banyak pekerjaan diselesaikan dalam jumlah waktu yang sama dan secara umum semua penggunaan dilayani lebih cepat[3]. Pada metode *load balancing* terdapat metode *round robin* dan *least connection*.

Penerapan dari *load balancing* pada *web server nginx* memiliki alasan karena *nginx* merupakan salah satu *software web server* yang *powerful* dan mempunyai performa tinggi, selain itu *nginx* juga di

desain untuk *server* dengan sumber daya yang kecil[4]. Penelitian akan membandingkan performa kinerja dari algoritma *load balancing round robin* dan *least connection*. *Round Robin* adalah salah satu teknik penyeimbangan beban yang ada yang mendistribusikan beberapa tautan jaringan untuk mencapai *throughput* maksimum dan waktu respons minimum untuk menghindari kelebihan beban[5].

Sedangkan algoritma *least connection* melakukan pembagian beban berdasarkan banyaknya koneksi yang sedang dilayani oleh sebuah *server*. *Server* dengan koneksi yang paling sedikit akan diberikan beban berikutnya, begitu pula *server* dengan koneksi banyak akan dialihkan bebannya ke *server* lain yang bebannya lebih rendah[6]. Perbandingan performa akan dilakukan menggunakan tools httperf untuk mengetaui kinerja web server menggunakan kedua algortima load balancing tersebut.

2. TINJAUAN PUSTAKA

Berikut merupakan tinjaun pustaka dari penelitian sebelumnya yang pernah dilakukan mengenai topik penelitian kali ini.

Dalam jurnal yang berjudul "Perancangan dan Pengujian *Load Balancing* dan *Failover* Mengunakan Nginx" dilakukan penelitian *load balancing* dengan teknik *failover* yang diimplementasikan pada sistem operasi Ubuntu. Penelitian ini digunakan untuk mengatasi beban kerja yang tinggi pada sebuah *server*. Pengujian ini menggunakan *software JMeter* untuk mengetahui hasil rancangan sistem yang dibuat. Hasil dari perancangan sistem ini berhasil dalam melakukan pembagian beban kerja *server*. Dalam pengujian lain juga menunjukan bahwa penggunaan *load balancing* dapat menurunkan waktu respon dan *throughput* menjadi meningkat[7].

Dalam jurnal internasional berjudul *"A High Availability Clusters Model Combined with Load Balancing and Shared Storage Technologies for Web Servers"* penulis mendesain dan mengimplementasikan *high avaibility cluster* dan digabungkan dengan infrastruktur *load balancing web server*. Sistem dapat menyediakan penuh fasilitas ke penyedia *hosting* situs *web* dan organisasi bisnis besar. Sistem ini dapat memberikan layanan berkelanjutan meskipun setiap komponen sistem gagal secara tidak pasti. Dengan bantuan teknologi *Linux Virtual Server (LVS) load balancing cluster* dan dikombinasikan dengan virtualisasi serta teknologi *shared storage* untuk mencapai arsitektur tiga tingkat dari *cluster server web*. Teknologi ini tidak hanya meningkatkan ketersediaan, tetapi juga mempengaruhi keamanan dan kinerja layanan aplikasi yang diminta. Manfaat dari sistem ini yaitu mengatasi *failover*; mengatasi kegagalan jaringan; mengatasi penyimpanan terbatas dan distribusi beban[2].

Pada Jurnal "Analisis Perbandingan Kinerja Web Server Apache dan Nginx Menggunakan Httperf Pada Portal Berita (Studi Kasus beritalinux.com)" penulis melakukan pengujian terhadap kinerja *web server* Apache dan Nginx sesuai dengan permintaan dari pengguna. Penulis melakukan pengujian menggunakan aplikasi Httperf untuk membandingkan *throughput*, koneksi, *request*, *reply*, dan *error* pada portal berita beritalinux.com. Setelah melakukan pengujian didapatkan hasil dimana dalam merespon dan menghubungkan data dari *request* pengguna, *web server* Nginx lebih unggul daripada Apache[8].

Dalam penelitian "Performa Kinerja Web Server Berbasis Ubuntu Linux dan Turnkey Linux" digunakan dua buah *server* yaitu sistem operasi Ubuntu Linux dan Turnkey Linux yang bertujuan membandingkan performa kinerja *web* dengan nilai *response time* dan nilai *throughput*. Pengujian melakukan *request rate* sebanyak 10 - 100 *request* dengan jumlah 1000 koneksi dan 2000 koneksi. Hasil dari pengujian diperoleh bahwa *web server* dengan Ubuntu Linux lebih baik dengan nilai *response time* kecil dan nilai *throughput* besar[9].

3. METODE PENELITIAN

3.1 Analisis Kebutuhan

Analisis kebutuhan digunakan untuk menentukan kebutuhan apa saja yang akan digunakan dalam penelitian. Dengan terpenuhinya kebutuhan akan mendapatkan hasil dari penelitian dan melakukan analisis. Berikut ini merupakan daftar kebutuhan yang dibutuhkan dalam penelitian ini.

3.1.1 Kebutuhan Perangkat Keras

Perangkat keras yang dibutuhkan dalam perancangan sistem penelitian ini yaitu:

a. PC *Server*

Server yang dibutuhkan yaitu 4 buah *server*. Dengan rincian 3 buah *server* sebagai *server* utama dan 1 buah *server* sebagai pengontrol *load balancing.* Karena keterbatasan perangkat PC maka pada penelitian ini menggunakan sebuah mesin *virtual* sebagai solusi dari permasalahan tersebut. 4 buah *server* dijalankan pada mesin *virtual* pada sebuah PC dengan masing-masing 2 inti untuk *processor*-nya.

Disesuaikan dengan spesifikasi PC *host* agar tidak keberatan menjalankan mesin *virtual*.

b. PC *Client*

Perangkat keras ini berfungsi untuk melakukan *request* ke PC *server*. Penelitian ini membutuhkan 3 buah PC *Client* untuk melakukan pengujian terhadap algoritma *load balancing*. Dimana *server* yang digunakan yaitu sebanyak 4 buah *server.*

c. Router

Perangkat keras yang terakhir yaitu 1 buah router (*Mikrotik routerboard hAp series*) digunakan untuk membuat sebuah jaringan *wireless* yang digunakan untuk menyambungkan antar PC *client* dan PC *server*. Router akan menjembatani *request* dari PC *client* dan diteruskan ke PC *server*.

3.1.2 Kebutuhan Perangkat Lunak

Perangkat lunak dibutuhkan untuk menjalankan perangkat keras agar dapat digunakan dalam perancangan sistem. Berikut ini merupakan perangkat lunak yang dibutuhkan dalam penelitian:

a. Oracle VM Virtualbox

Merupakan sebuah perangkat lunak virtualisasi yang dapat digunakan sebagai media untuk menjalankan sistem operasi tambahan (tamu) pada sistem operasi utama *(host)*. Dengan adanya virtualbox, *server* virtual yang dibangun dapat menyerupai *server* nyata.

b. Winbox

Digunakan untuk menghubungkan *Mikrotik* melalui *MAC Address* atau protokol IP. Winbox dapat melakukan konfigurasi terhadap *Mikrotik RouteOS* dan *Routerboard*.

c. Browser

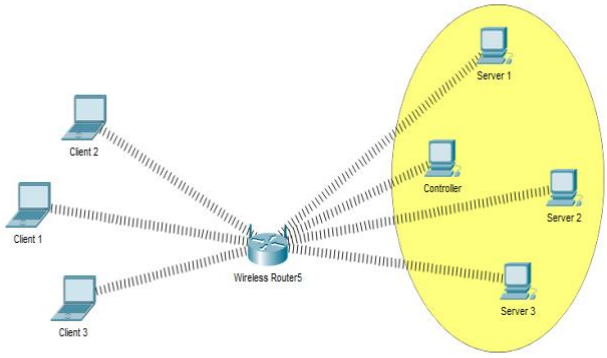
Digunakan untuk pengetesan *web server* apakah berjalan atau tidak. *Browser* juga sebagai indikator apakah *server* telah hidup dan dapat melayani permintaan *client*.

d. Sistem Operasi

Pada penelitian ini menggunakan sistem operasi Linux Ubuntu 20.04 sebagai *server* utama. Dimana Ubuntu memiliki utilitas yang memadai, *user friendly*, dan merupakan perangkat lunak bebas.

3.2 Perancangan Sistem

Perancangan sistem jaringan menggunakan sebuah router sebagai penghubung antara *client* dan *server*. Sebuah *server* digunakan untuk pengontrol *load balancing* antar *server utama*. Berikut adalah gambar dari perancangan topologi jaringan.



Gambar 1. Topologi Jaringan

Pada gambar diatas merupakan diagram perancangan sistem. Berikut ini merupakan penjelasan mengenai topologi jaringan.

a. *Client*

Client berperan dalam melakukan *request* terhadap *server*. *Request* dari *client* akan diterima oleh *wireless router* untuk diatur ke *server controller*. Kemudian *request* akan diterima oleh *server*.

b. *Server*

Sebagai bagian dalam penyedia layanan yang mampu melayani permintaan *client (request). Server* akan menyediakan data yang dibutuhkan oleh *client* yang secara bergantian sesuai algoritma pada *server controller*.

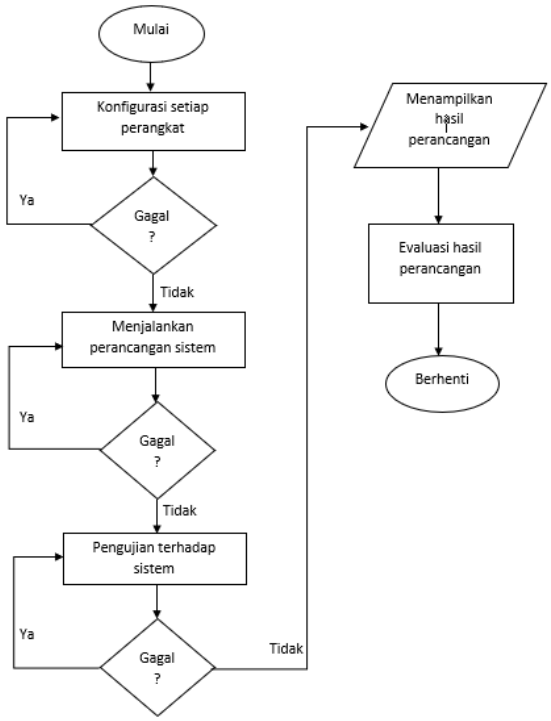
c. *Wireless Router*

Wireless router akan menghubungkan setiap perangkat yang terhubung. Dimana bagian ini *router* meneruskan paket *request* dalam jaringan yang telah diarahkan sesuai algortima dalam *server controller*.

d. *Controller*

Controller sebagai *server load balancing* merupakan pusat dalam mengontrol terhadap sistem yang berjalan. *Controller* digunakan untuk menjalankan sistem *load balancing*. *Request* yang telah diterima oleh *controller* dari *router*, akan diteruskan dengan melakukan pemilihan *server* terlebih dahulu sesuai algoritma *load balancing*. Setelah ditentukan maka *request* akan langsung diteruskan ke *server*.

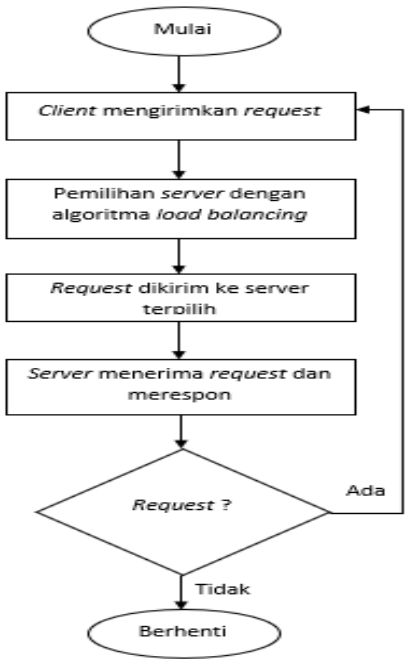
3.2.1 Diagram Alur Perancangan Sistem



Gambar 2. Alur Perancangan Sistem

3.2.2 Perancangan Load Balancing

Pada penelitian ini menggunakan perancangan sistem jaringan dengan menerapkan algoritma *load balancing* sebagai mekanisme dalam pemilihan *server*. Berikut ini merupakan diagram alur dari proses kerja *load balancing* :



Gambar 3. Diagram alur *Load Balancing*

Pada gambar 3 dapat dijelaskan bagaimana proses *load balancing* secara umum, yaitu pembagian beban pada *server* dalam membagi *traffic* secara otomatis. Pada awal proses, *client* melakukan request kepada *server*. *Request* dari *client* akan masuk ke dalam *controller load balancing* dan dilakukan pengecekan *server*. Kemudian *controller* akan memilih *server* sesuai algoritma *load balancing*. *Request client* akan diteruskan ke dalam antrian *server*. *Server* akan merespon dan mengirimkan data kepada *client*. Sistem *load balancing* akan terus berjalan hingga *request* dari *client* telah berhenti.

3.3 Konfigurasi Perangkat

Pada bab sebelumnya telah dibahas mengenai perancangan sistem. Dimana dalam perancangan sistem tersebut terdapat langkah dalam konfigurasi perangkat. Konfigurasi perangkat merupakan langkah awal dalam melakukan analisa terhadap algoritma *load balancing*. Berikut merupakan konfigurasi perangkat yang digunakan:

a. Konfigurasi pada perangkat *Router*

Router mikrotik yang digunakan pada penelitian ini adalah *Mikrotik routerboard hAp series. Router* dapat kita lakukan konfigurasi melalui aplikasi *Winbox*. Pada dasarnya routerboard mikrotik bisa dikonfigurasi *wifi hotspot* dengan mode GUI. Pada penelitian ini penulis menggunakan mode CLI dalam konfigurasi. Berikut konfigurasi *router*:

```
>Interface enable wlan1
>Interface wireless set 0 mode=ap-bridge band=2ghz-b/g/n
ssid=Router
>ip address add address=192.168.1.1/24
interface: wlan1
>ip hotspot setup
hotspot interface: wlan1
local address of network: 192.168.1.1/24
masquerade network: yes
address pool of network: 192.168.1.2-192.168.1.20
select certificate: none
ip address of smtp server: 0.0.0.0
dns servers: 192.168.1.1
dns name: router.com
```

Gambar 4. Konfigurasi hotspot router

b. Konfigurasi pada perangkat *Server*

Penelitian ini menggunakan *server* virtual dengan memanfaatkan aplikasi *Oracle VM Virtualbox* dikarenakan keterbatasan perangkat komputer sebagai *server*. Dimana terdapat 4 buah *server*, sebuah *server* sebagai *controller load balancing* dan 3 buah *server* sebagai *web server*. Konfigurasi pada *server* dengan melakukan pengaturan terhadap alamat IP, jumlah *core*, *adapter* yang dipakai dan sebagainya.

1. Pengaturan pada *virtualbox* dimana jumlah inti tiap *server* virtual yaitu 2 inti. Kemudian pada bagian *adapter* menggunakan jaringan *Bridged Adapter* dengan memanfaatkan *Wireless Network Adapter.*

2. Melakukan pengaturan IP dengan perintah ***$ sudo nano /etc/netplan/00-installer-config.yaml*** pada setiap *server* dengan alamat IP statis:

Tabel I Alamat IP statis *server*

Nama server	Alamat IP
Server 1	192.168.1.12
Server 2	192.168.1.13
Server 3	192.168.1.14
Server *Load Balancing*	192.168.1.15

3. Instalasi *web server* pada setiap *server*. Pada penelitian ini penulis menggunakan *web server nginx* dengan perintah instalasi ***$ sudo apt install nginx.*** *Website* yang dipakai dalam pengujian ini yaitu menggunakan *Wordpress* yang telah dipasang pada setiap *server*.

c. Konfigurasi pada perangkat *Client*

Konfigurasi pada perangkat *client* yaitu melakukan pengaturan terhadap alamat IP. Pada perangkat *client* dilakukan pengaturan alamat IP secara otomatis (*DHCP*).

Dimana *range IP* yang dipakai oleh *client* yaitu **192.168.1.2 - 192.168.1.20.** Untuk melakukan pengujian sistem maka pada *client* harus terpasang *tools httperf*.

d. **Konfigurasi Algoritma *Load Balancing***

Pengaturan algoritma *load balancing* dilakukan pada *server controller load balancing*. Konfigurasi dilakukan pada file *default* pada direktori *etc/nginx/sites-available.* Berikut merupakan *source code* dari *load balancing round robin* dan *least connection*.

1. Algoritma Round Robin

Round robin merupakan metode *default* apabila parameter algoritma tidak ditentukan. Maka otomatis metode yang digunakan menggunakan algoritma *round robin.* Beban akan dibagi rata ke seluruh *server*.

```
upstream backend {

        server 192.168.1.12;
        server 192.168.1.13;
        server 192.168.1.14;
}

server {
        server_name 192.168.1.15;
        listen 80;

        location / {
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_pass http://backend;
        }
}
```

Gambar 5. *Sorce code* algoritma *round robin*

2. Algoritma Least Connection

Algoritma *least connection* membagi beban dengan memperhatikan *server* yang koneksinya paling sedikit. *Server* dengan koneksi paling sedikit akan diberikan beban berikutnya, begitu pula *server* dengan koneksi banyak akan dialihkan bebannya ke *server* lain yang bebannya lebih rendah. Sehingga beban setiap *server* berbeda-beda.

```
upstream backend {
        least_conn;
        server 192.168.1.12;
        server 192.168.1.13;
        server 192.168.1.14;
}

server {
        server_name 192.168.1.15;
        listen 80;

        location / {
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_
                proxy_pass http://backend;
        }
}
```

Gambar 6. *Source code* algoritma *least connection*

3.4 Pengujian Perancangan

Perancangan sistem yang telah dilakukan selanjutnya dilakukan pengujian. Tujuan dari pengujian yaitu untuk mengetahui perbandingan performa dari algoritma *round robin* dan *least connection* pada *web server*. Pengujian juga dilakukan untuk mengetahui nilai rata-rata pada setiap parameter yang diberikan. Pemberian *request* digunakan untuk memberi beban kepada *server* dalam satu waktu. Hasil dari setiap percobaan akan dibandingkan dan dilakukan analisa. Sehingga akan didapat perbandingan perfoma antara algoritma *round robin* dan *least connection* yang dapat disimpukan pada akhir penelitian.

Pengujian yang dilakukan menggunakan *httperf* dengan parameter *throughput* (KB/s) dan *response time (ms/req)*.

a. *Throughput* merupakan *bandwith* aktual yang terukur pada ukuran waktu dalam jaringan. Pengujian ini digunakan untuk melihat performa jaringan dari segi kecepatan *bandwith* dalam pengiriman paket.

b. *Response time* digunakan untuk mengetahui seberapa cepat *server* dalam menerima permintaan dari *client*.

3.5 Skenario Pengujian

Pada skenario pengujian sistem menggunakan *tools httperf* dengan memberikan banyak koneksi dengan masing-masing *rate* berbeda. Untuk menjalankan pengujian pada *tools httperf*, berikut perintah yang digunakan:

$ sudo httperf --server 192.168.1.15 --port 80 --num-conns a --rate b

Pengujian dilakukan dalam kondisi setiap *server* telah menjalankan *web server* nginx dan dalam keadaan tidak menjalankan program aplikasi lain (*idle*). Skenario pengujian dengan melakukan pengujian terhadap *throughput* dan *response time* dari algoritma *load balancing* pada server 192.168.1.15. Parameter *throughput* yaitu dengan satuan *KB*/s dan *response time* dengan satuan *ms*/req.

Pengujian pada parameter *throughput* dan *response time* dilakukan dengan sebanyak 4 kali percobaan dengan dibagi menjadi 3 kategori:

- 30 *req/s (rate)* dengan banyak koneksi 30.
- 60 *req/s (rate)* dengan banyak koneksi 60.
- 120 *req/s (rate)* dengan banyak koneksi 120.

Hasil dari pengujian dengan mengirimkan *request* akan didapat nilai dari *throughput* dan *response time* yang kemudian diambil nilai rata-ratanya dari hasil 4 kali percobaan. Perekaman hasil pengujian *httperf* dilakukan secara manual yaitu dengan menulis hasil pengukuran. Hal tersebut dilakukan karena pada *httperf* tidak ada fitur untuk menyimpan hasil pengukuran secara otomatis.

4. HASIL DAN PEMBAHASAN

4.1 Hasil Pengujian

Berikut ini disajikan tabel hasil pengukuran terhadap performa *throughput* dan *response time* menggunakan *tools httperf*.

a. *Throughput*

Tabel II Hasil pengujian throughput

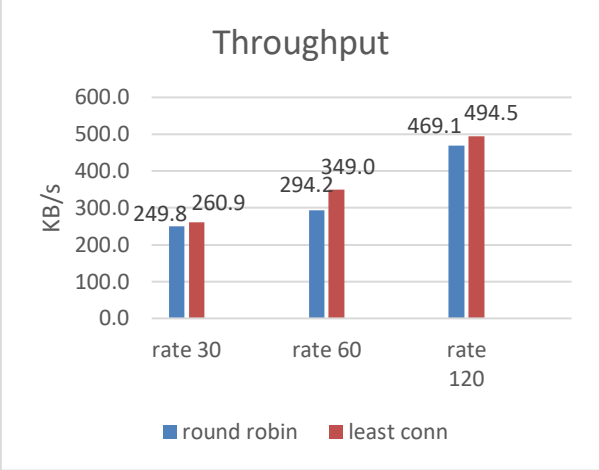
no.	***Throughput (KB/s)***					
	rate 30		***rate 60***		***rate 120***	
	round robin	***least conn***	***round robin***	***least conn***	***round robin***	***Least conn***
1	263.2	261.8	266.9	412.2	420.8	504
2	228	262.6	277.1	283.1	492.4	496.6
3	254.5	255.8	427.4	281.8	490.2	493.6
4	253.3	263.5	205.3	418.8	473.1	483.8
Rata-rata	**249.8**	**260.9**	**294.2**	**349.0**	**469.1**	**494.5**

Pada Tabel II merupakan hasil dari pengukuran *throughput* yang diperoleh dari pengujian *algoritma* menggunakan *httperf*. Nilai didapat pada bagian *Net I/O.* Nilai pada tabel merupakan jumlah total transfer data yang berhasil dilakukan pada *server*. Tabel diatas dapat dijelaskan sebagai berikut:

1. Pada *rate* 30 nilai rata-rata *throughput* dari *round robin* sebesar 249,8 *KB/s* dan *least connection* sebesar 260,9 *KB/s.*

2. Pada *rate* 60 nilai rata-rata *throughput* dari *round robin* sebesar 294,2 *KB/s* dan least connection sebesar 349,0 *KB/s*.

3. Pada *rate* 120 nilai rata-rata *throughput* dari *round robin* sebesar 469,1 *KB/s* dan *least connection* sebesar 494,5 *KB/s*.



Gambar 7. Perbandingan throughput round robin dan least connection

Dari hasil pengukuran nilai *throughput* pada algoritma *round robin* dan *least connection* dapat dilakukan perbandingan dengan melihat masing-masing nilai pada setiap *rate* pada gambar 7. Dari keseluruhan pengujian *throughput* pada *rate* 30, *rate* 60 dan *rate* 120, algoritma *least connection* lebih unggul dengan menghasilkan nilai *throughput* yang lebih besar dari algoritma *round robin*. Semakin besar *throughput* yang dihasilkan maka semakin baik *server* dalam mengirimkan paket kepada *client*.

b. *Response time*

Tabel III. Hasil pengujian response time

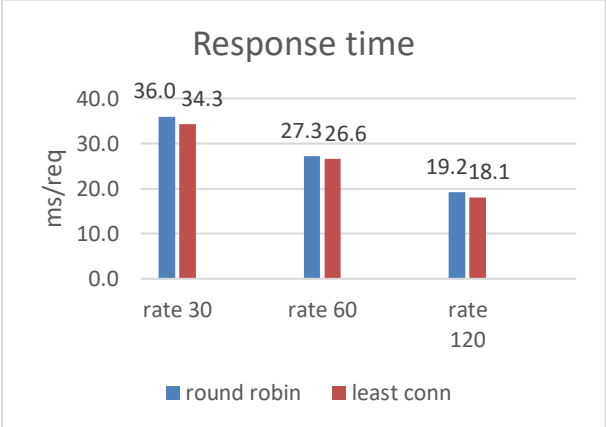
Response time (ms/req)						
no.	***rate 30***		***rate 60***		***rate 120***	
	round robin	***least conn***	***round robin***	***least conn***	***round robin***	***least conn***
1	34	34.2	33.6	21.7	21.3	17.8
2	39.3	34.1	32.3	31.6	18.2	18
3	35.2	35	21	31.8	18.3	18.1
4	35.4	34	22.1	21.4	18.9	18.5
Rata-rata	**36.0**	**34.3**	**27.3**	**26.6**	**19.2**	**18.1**

Pada tabel III merupakan hasil pengukuran *response time* yang diperoleh dari hasil pengujian algoritma dengan *httperf*. Nilai *response time* merupakan lamanya transfer tiap *request* data yang berhasil dijalankan. Nilai tersebut dilihat pada bagian *request rate*. Dari tabel tersebut dapat dijelaskan sebagai berikut:

1. Pada *rate* 30 nilai rata-rata *response time* algoritma *round robin* sebesar 36,0 *ms/req*, *least connection* sebesar 34,3 *ms/req.*

2. Pada *rate* 60 nilai rata-rata *response time* algoritma *round robin* sebesar 27,3 *ms/req*, *least connection* sebesar 26,6 *ms/req.*

3. Pada *rate* 120 nilai rata-rata *response time* algoritma *round robin* sebesar 19,2 *ms/req*, *least connection* sebesar 18,1 *ms/req.*



Gambar 8. Perbandingan response time round robin dan least connection

Pada gambar 8 merupakan perbandingan nilai *response time* dari algoritma *round robin* dan *least connection*. Terdapat perbedaan dari kedua algoritma, nilai yang dihasilkan tidak jauh berbeda. Algoritma *least connection* memiliki *response time* lebih rendah dari *round robin* dari ketiga *rate* yang diuji yaitu *rate* 30, *rate* 60, dan *rate* 120. Semakin rendah *response time* pada *server* maka semakin bagus *server* tersebut.

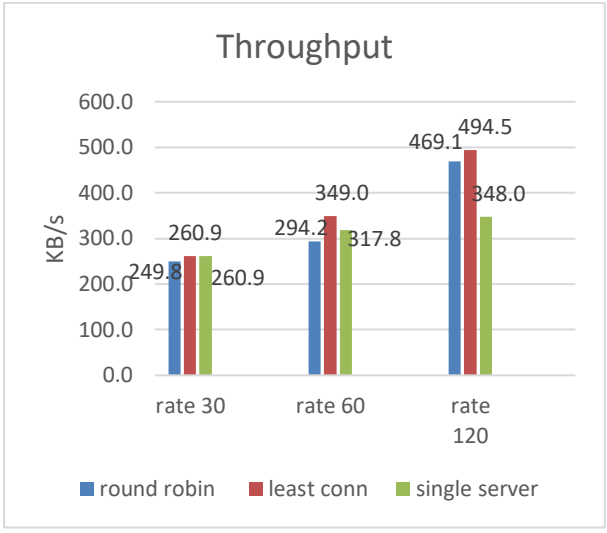
c. *Server* tunggal dengan sistem *load balancing*

Pada bagian ini akan dibandingkan kinerja dari *server* tunggal dan sistem *load balancing*.

Tabel IV Hasil pengujian *single server*

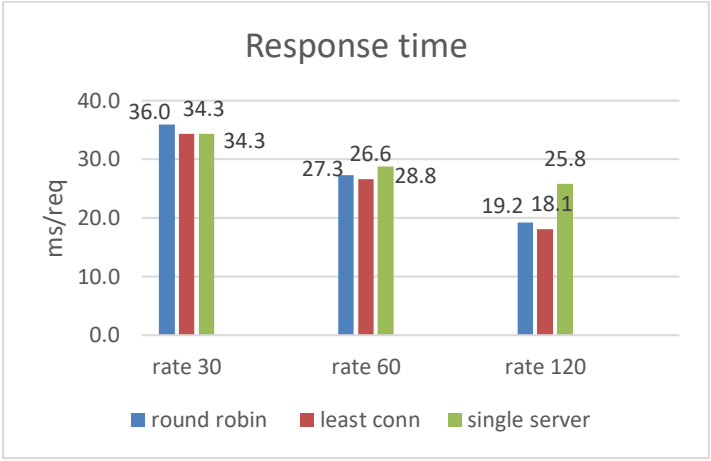
rate	no.	*Throughput (KB/s)*	*Response Time (ms/req)*
30	1	261	34.3
	2	255.2	35.1
	3	264.3	33.9
	4	263.2	34
	Rata-rata	**260.9**	**34.3**
60	1	279.9	32
	2	363.4	24.6
	3	364.7	24.6
	4	263.2	34
	Rata-rata	**317.8**	**28.8**
120	1	319	28.1
	2	354.8	25.2
	3	353.8	25.3
	4	364.3	24.6
	Rata-rata	**348.0**	**25.8**

Pada tabel IV merupakan hasil pengukuran pada single server dengan alamat IP 192.168.1.12 (*server* 1). Dengan hasil nilai *throughput* dan *response time* pada tabel tersebut diugunakan untuk membandingkan dengan sistem *load balancing*.



Gambar 9. Perbandingan throughput server tunggal dengan load balancing

Gambar 9 merupakan perbandingan antara *server* tunggal sebelum dilakukan *load balancing* dan sesudah menggunakan sistem *load balancing*. Pengukuran menggunakan aplikasi httperf dengan kondisi *server* tetap dalam keadaan *idle*. Pada pengujian *rate* 30, nilai *throughput single server* sama dengan nilai dari *least algoritma connection*. Pada *rate* 60, *throughput least connection* lebih unggul dan berlanjut hingga *rate* 120 dimana *single server* memiliki nilai yang lebih rendah. Sementara kedua algoritma *load balancing* mampu meningkatkan *throughput*. *Single server* pada *rate* 120 mengalami kenaikan *throughput* namun lebih rendah. Dengan semakin meningkatnya koneksi yang diberikan, maka sistem *load balancing* lebih unggul dalam performa *throughput*. Sehingga load balancing efektif untuk di implementasikan pada jalur koneksi yang padat.



Gambar 10. Perbandingan response time single server dengan load balancing

Pada gambar 10 menunjukan perbandingan nilai *response time* pada *single server* sebelum diterapkan *load balancing* dan setelah menggunakan sistem *load balancing*. Pada pengujian *rate* 30, nilai *response time single server* dan *least connection* lebih rendah dari *round robin*. *Single server* masih mampu mengatasi waktu respon dalam koneksi sedikit. Pada *rate* 60, sistem *load balancing* memiliki waktu respon yang lebih rendah dari *single server*. Begitu juga pada *rate* 120, nilai waktu respon sistem *load balancing* lebih rendah dari *single server*. sehingga pada pengujian *response time*, sistem *load balancing* lebih baik pada koneksi yang semakin banyak.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Pada bagian ini merupakan bagian akhir dalam melakukan penelitian setelah melakukan pengujian dan analisa dari sistem yang telah dirancang. Berikut beberapa kesimpulan dari penulis dari hasil penelitian:

a. Penerapan algoritma *load balancing round robin* dan *least connection* dapat berjalan dengan baik dan memberikan nilai pengukuran yang baik. Apabila dibandingkan dengan *single server*, sistem *load balancing* menghasilkan nilai *throughput* dan *response time* lebih baik pada koneksi yang lebih banyak.
b. Pada penelitian ini perbandingan antara algoritma *load balancing*, performa dari algoritma *least connection* lebih baik dan stabil pada semua pengujian *rate*.

5.2 Saran

Terdapat beberapa saran yang ingin penulis berikan kepada para peniliti lain yang ingin mengembangkan penelitian ini antara lain:

1. Mengembangkan algoritma *least connection* agar lebih maksimal.
2. Mengimplementasikan sistem *load balancing* pada *server* fisik.
3. Melakukan pengujian dengan parameter yang bervariasi untuk menghasilkan hasil yang maksimal.

DAFTAR PUSTAKA

[1] I. G. L. P. E. Supramana, Prismana, "Implementasi Load Balancing Pada Web Server Dengan Menggunakan Apache," *J. Manaj. Inform.*, vol. 5, no. 2, pp. 117–125, 2016.

[2] A. B. M. Moniruzzaman, M. Waliullah, and M. S. Rahman, "A high availability clusters model combined with load balancing and shared storage technologies for web servers," *Int. J. Grid Distrib. Comput.*, vol. 8, no. 1, pp. 109–120, 2015, doi: 10.14257/ijgdc.2015.8.1.11.

[3] K. Pendke, A. Kasrekar, A. Sawarkar, T. Yeddeloo, and R. Bhusari, "The Concept of Load Balancing Server in Secured and Intelligent Network," *Int. J. Adv. Eng. Manag. Sci.*, vol. 3, no. 3, pp. 221–225, 2017, doi: 10.24001/ijaems.3.3.12.

[4] A. Wijaya and A. M. Harjuna, "PERANCANGAN PROGRAM APLIKASI TABUNGAN MENGGUNAKAN SERVER NGINX (Studi Kasus Madrasah Ibtidayah Negeri 01 Kota Bengkulu)," *Pseudocode*, vol. 4, no. 1, pp. 29–36, 2017, doi: 10.33369/pseudocode.4.1.29-36.

[5] T. Sasidhar, V. Havisha, S. Koushik, M. Deep, and V. Krishna Reddy, "Load balancing techniques for efficient traffic management in cloud environment," *Int. J. Electr. Comput. Eng.*, vol. 6, no. 3, pp. 963–973, 2016, doi: 10.11591/ijece.v6i3.7943.

[6] H. Nasser and T. Witono, "Analisis Algoritma Round Robin, Least Connection, Dan Ratio Pada Load Balancing Menggunakan Opnet Modeler," *J. Inform.*, vol. 12, no. 1, pp. 25–32, 2016, doi: 10.21460/inf.2016.121.455.

[7] F. S. Rahmad Dani, "Perancangan dan Pengujian Load Balancing dan Failover Menggunakan Nginx," *J. Ilmu Komput. dan Inform.*, vol. 3, no. 1, pp. 43–50, 2017, doi: https://doi.org/10.23917/khif.v3i1.2939.

[8] I. F. Irza, Zulhendra, and Efrizon, "Analisis Perbandingan Kinerja Web Server Apache dan Nginx Menggunakan Httperf Pada Portal Berita (Studi Kasus beritalinux.com)," *J. Vokasional Tek. Elektron. Inform.*, vol. 5, no. 2, pp. 75–82, 2017, doi: https://doi.org/10.24036/voteteknika.v5i2.8489.

[9] I. Y. Andhica and D. Irwan, "Performa Kinerja Web Server Berbasis Ubuntu Linux Dan Turnkey Linux," *PIKSEL Penelit. Ilmu Komput. Sist. Embed. Log.*, vol. 5, no. 2, pp. 68–78, 2018, doi: 10.33558/piksel.v5i2.269.