Implementasi *Load Balancing* Dan *Virtual Machine* Dengan Algoritma *Round Robin* Pada Sistem Informasi Penerimaan Pegawai Bppt

Annisa Andarrachmi, S.Kom

Balai IPTEKnet, Badan Pengkajian dan Penerapan Teknologi (BPPT)
E-mail : ichazami@gmail.com, nisa@iptek.net.id, annisaandarrachmi@bppt.go.id

ABSTRAK

Peningkatan jumlah pelamar di Badan Pengkajian dan Penerapan Teknologi (BPPT) berpengaruh terhadap kinerja *server* terutama *web server* yang mengalami penurunan penyajian halaman *website*. Salah satu teknologi yang digunakan untuk optimalisasi kinerja *server* atau *web server* ialah menggunakan *load balancing* dan *virtual machine*. *Web server* berbasis *Linux* dan *virtual machine* dapat meningkatkan kinerja *web server* melalui pembagian beban kerja pada sekelompok *web server* yang dikembangkan pada sistem operasi *Linux* distro *Ubuntu*. NDLC atau *Network Development Life Cycle* merupakan metode yang digunakan dalam pembangunan sistem jaringan ini. Algoritma *round robin* dipilih karena dapat membagi beban secara bergiliran dan berurutan dari satu *server* ke *server* lain sehingga membentuk putaran. Sistem jaringan yang dibangun, diuji dan dibandingkan sebelum dan sesudah pengimplementasian *load balancing* dan *virtual machine*. Berdasarkan *response time* hasil pengujian sesudah pengimplementasian *load balancing* dan *virtual machine* menunjukkan bahwa kinerja *server* atau *web server* lebih tinggi untuk beban akses besar.

Kata kunci : *web server, load balancing, virtual machine, round robin algorithm, NDLC*

1. Latar Belakang

Penerapan teknologi jaringan komputer yang disertai dengan peningkatan jumlah pelamar untuk menjadi pegawai di Badan Pengkajian dan Penerapan Teknologi (BPPT) mengakibatkan peningkatan terhadap jumlah data pelamar. Aplikasi penerimaan pegawai yang mulai digunakan tahun 2009 berbasis *web* dengan menggunakan *web server* sebagai penyimpanan dan membuat ketersediaan suatu *website*. Seiring bertambahnya pengguna dan permintaan yang terus-menerus, kinerja *web server* menurun dan lambat laun mengalami kelebihan beban (*overload*) dalam waktu yang relatif singkat. Masalah kelebihan beban akses pada *web server* perlu diatasi dengan melakukan optimalisasi kinerja *server*.

Jumlah pengguna jaringan komputer berjumlah lebih dari 1000 orang yang akan menggunakan fasilitas aplikasi layanan berbasis *web* maka dibutuhkan sebuah konfigurasi *server* yang handal. Selain sisi konfigurasi perangkat keras yang menjadi pertimbangan agar *server* nanti handal maka terdapat layanan-layanan yang ada pada *server* harus bisa mengantisipasi pengaksesan aplikasi berbasis *web* tersebut secara simultan dan mempunyai frekuensi yang sangat tinggi.[12]

1.2. Tujuan

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1) Mendistribusikan beban akses dengan pengimplementasian *load balancer* dan *virtual machine* dengan algoritma *round robin* pada studi kasus aplikasi penerimaan pegawai BPPT.
2) Membangun fasilitas manajemen dengan pengimplementasian *load balancer* dan *virtual machine* dengan algoritma *round robin* pada studi kasus aplikasi penerimaan pegawai BPPT.
3) Mendukung penyelenggaraan aplikasi penerimaan pegawai BPPT.

1.3. Ruang Lingkup

a. Pembangunan *load balancer* dan *virtual machine* mengikuti tahap *Network Development Life Cycle* (NDLC).

b. Penerapan *load balancing* pada *web server* terdiri dari dua buah komputer dengan dengan spesifikasi perangkat keras yang tidak identik.
c. Pembangunan sistem jaringan *load balancing* dan *virtual machine* berbasis *open source* dengan menggunakan Sistem Operasi *Linux* Ubuntu.
d. Penggunaan perangkat lunak untuk mengkonfigurasi *load balancing* berbasis pada algoritma *round robin* karena pemakaian memori pada CPU (*Control Processing Unit*) yang sedikit namun berkemampuan tinggi.
e. Sistem *load balancing* ini akan diuji untuk beban koneksi dari 100 *request user* hingga 5.000 *request user* dalam rentang waktu yang berdekatan.
f. Analisa pengujian kinerja web *server* yang terimplementasi *load balancing* dilihat pada permintaan koneksi per detik secara iteratif terhadap keberhasilan pelayanan dan kemerataan beban pada masing-masing web *server*.
g. Penelitian dilakukan langsung pada jaringan komputer yang sedang berjalan.

I. Tinjauan Pustaka

2.1. *Server*

Server adalah sebuah komputer yang berisi program baik sistem operasi maupun program aplikasi yang menyediakan pelayanan kepada komputer atau program lain yang sama ataupun berbeda [14].

2.2. *Web Server*

2.2.1 *Nginx Web Server*

Nginx webserver merupakan perangkat lunak gratis dan *open source* yang dapat berjalan pada sistem operasi berbasis *Linux, Mac OS, Windows*. Aplikasi ini sekarang digunakan untuk melayani beberapa situs paling populer di web seperti *Wordpress, Hulu, SourceForge*, dan banyak lagi. *Nginx* telah terbukti menjadi *webserver* yang sangat efisien, ringan namun kuat. Aplikasi ini berjalan baik pada arsitektur 32 dan 64 bit. [11]

2.3. *Load Balancing*

Load balancing atau penyeimbangan beban dalam jaringan sangat penting bila skala dalam jaringan komputer makin besar demikian juga *traffic* data yang ada dalam jaringan komputer makin lama semakin tinggi [12]. Beban yang seimbang pada *server* terhitung sesuai dengan arus beban dari mesin tersebut [7].

Fungsi *load balancer* adalah untuk memintas lalu-lintas jaringan yang mengarah pada *server* dan membagi lalu-lintas ke dalam permintaan tunggal dan memilih *server* yang menerima permintaan tersebut [3].

Metode yang digunakan untuk implementasi *load balancing* ini menggunakan perpaduan antara perangkat keras dan perangkat lunak dimana perangkat keras yang dioptimasi dan diisi oleh *platform* berbasis *linux* untuk menjalankan perangkat lunak utama pada *load balancer*. Fleksibilitas yang luar biasa didapatkan mulai dari menggunakan perangkat keras yang selalu *up to date* sampai dengan menggunakan sistem operasi dengan *patch* terbaru. Sehingga waktu guna dari perangkat ini dapat lebih panjang daripada sebuah switch khusus yang tidak fleksibel. Solusi ini tentunya jauh lebih murah dibandingkan dengan solusi perangkat keras khusus atau solusi perangkat lunak saja.[12]

2.4. *Virtual Machine* (Mesin Virtual)

Mesin virtual hanya dapat berjalan pada *monitor-mode* jika berupa sistem operasi, sedangkan mesin virtual itu sendiri berjalan dalam bentuk *user-mode*. Konsekuensinya, baik virtual *monitor-mode* maupun virtual *user-mode* harus dijalankan melalui *physical user mode*. Hal ini menyebabkan adanya *transfer* dari *user-mode* ke *monitor-mode* pada mesin nyata, yang juga akan menyebabkan adanya transfer dari virtual

user-mode ke virtual *monitor-mode* pada mesin virtual.[9]

2.5. Algoritma *Round Robin*

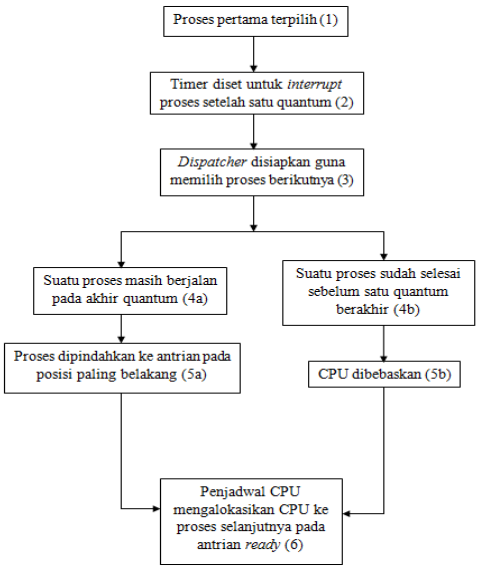
Merupakan algoritma yang paling sederhana dan banyak digunakan oleh perangkat *load balancing*. Algoritma ini membagi beban secara bergiliran dan berurutan dari satu *server* ke *server* lain sehingga membentuk putaran.[12]

Algoritma *round robin* digunakan pada *server* untuk menyeimbangkan beban pada beberapa *server*. Penjadwalan algoritma *round robin* memberikan tugas-tugas ke semua bagian dalam bentuk matematika. *Round robin* memilih *server* dengan i = (i + 1) mod n setiap kali, dimana n adalah jumlah *server*.[1]. Prosedur formal penjadwalan *round robin* seperti tabel 2.1.

Tabel 2.1. Algoritma *Round robin* [1]

```
Algoritma round_robin
S = {S0, S1, .... , Sn-1} ialah server dengan jumlah n
    dimana n>0
j ialah digunakan untuk banyaknya proses yang komplit
i ialah mengindikasikan server yang terseleksi saat akhir
dan i juga diinisialisasikan dengan -1 sehingga j = i
 do { j = (j + 1) mod n;
 if (Available (Sj))
 {
          {
           i = j;
           return Si;
          }
 }
 while (j != i);
 return NULL;
```

Urutan kejadian algoritma *round robin* dapat dilihat seperti pada gambar 2.4 yang mengacu pada [10] :



Gambar 2.4. Urutan Kejadian Algoritma *Round Robin*

Berdasarkan gambar 2.4 jika suatu proses memiliki CPU *burst* lebih kecil dibandingkan dengan *quantum-time*, maka proses tersebut akan melepaskan CPU jika telah selesai bekerja, sehingga CPU dapat segera digunakan oleh proses selanjutnya. Sebaliknya, jika suatu proses memiliki CPU *burst* yang lebih besar dibandingkan dengan *quantum-time*, maka proses tersebut akan dihentikan sementara jika sudah mencapai *quantum-time*, dan selanjutnya mengantri kembali pada posisi ekor dari *ready queue*, CPU kemudian menjalankan proses berikutnya.[9]. Misalkan ada 3 proses: P1, P2, dan P3 yang meminta pelayanan CPU, jika *quantum-time* sebesar 4ms, maka:

Process	Burst Time
P_1	24
P_2	3
P_3	3

Gambar 2.5. Proses P1,P2, dan P3 [9]

Gambar 2.5 merupakan tiap-tiap proses dengan *burst time* yang meminta pelayanan CPU.

Pi	P2	P3	P1	P1	P1	P1	P1

Gambar 2.6. *Gant Chart* dari Tiap-Tiap Proses [9]

Gambar 2.6 merupakan tiap-tiap proses yang berjalan berdasarkan ketentuan algoritma *round robin* dengan membandingkan *CPU burst* dan *quantum-time*.

Tabel 2.2. Waktu untuk Tiap-Tiap Proses [9]

Proses	Waiting Time (ms)
P1	0 + (10-4) = 6
P2	4
P3	7

Dari tabel 2.2 didapatkan rata-rata waktu tunggu = (6+4+7)/3= 5.66 ms. Algoritma *round robin* memiliki keuntungan, yaitu adanya keseragaman waktu.

2.6. *Open System Interconnect* (OSI)

OSI adalah suatu standar komunikasi antar mesin yang terdiri atas 7 lapisan. Ketujuh lapisan tersebut mempunyai peran dan fungsi yang berbeda satu terhadap yang lain. Setiap *layer* bertanggung jawab secara khusus pada proses komunikasi data [14].

2.7. *Transmission Control Protocol/Internet Protocol* (TCP/IP)

TCP/IP adalah sekumpulan protokol yang didesain untuk melakukan fungsi-fungsi komunikasi data pada *W ide Area Network* (WAN).

III. Analisa dan Perancangnan Sistem

3.1 Analisa Sistem

3.1.1. Kebutuhan Akan Sistem

Penelitian menggunakan pemodelan jaringan untuk mengimplementasikan sistem jaringan load *balancing* dan *virtual machine* terdiri dari satu komputer dengan virtualisasi *cluster* menjadi dua *web server* serta satu komputer sebagai *server* untuk mengimplementasikan sistem terhadap algoritma *round robin* pada *load balancer*.

3.1.2. Tujuan Sistem

Sistem jaringan *load balancing* dan *virtual machine* ini diharapkan dapat menyeimbangkan beban pada *web server* atau *server* untuk mencegah terjadinya *overload* atas beban akses permintaan *user*.

3.1.3. Analisa Kebutuhan

3.1.3.1. Kebutuhan Fungsional

Kebutuhan fungsional meliputi fungsi-fungsi yang harus dipenuhi oleh sistem.

Tabel 3.1. Kebutuhan Fungsional Sistem

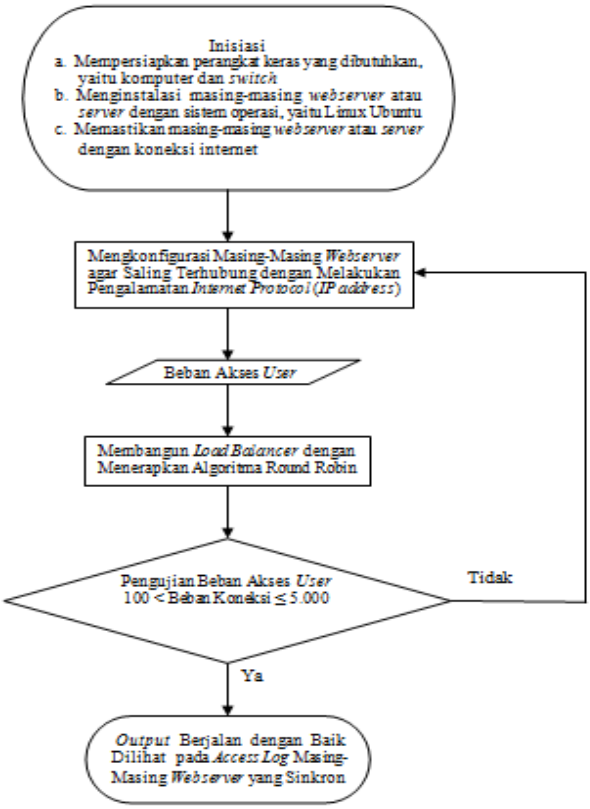
Subsistem	Kebutuhan Fungsional
Load balancer	1. Menerima permintaan *user* 2. Menentukan *web server* yang melayani permintaan *user* 3. Menjadwalkan *rute* permintaan *user* sesuai dengan algoritma penjadwalan *round robin* 4. Memastikan *web server* merespon permintaan *user*
Web server	1. Menerima permintaan *user* dari *load balancer* 2. Menjawab permintaan *user* 3. Mengakses *database server* sesuai permintaan *user*
Database server	1. Menyimpan data dari hasil masukkan *user*

3.1.3.2. Kebutuhan Nonfungsional

Kebutuhan nonfungsional meliputi kebutuhan sistem operasi dan kebutuhan perangkat lunak pendukungnya. Sistem operasi yang digunakan ialah Ubuntu *Server* 10.04. Perangkat lunak pendukungnya yaitu *Apache web server*.

3.2. Perancangan Sistem

Perancangan arsitektur sistem ialah korelasi fisik antara masing-masing subsistem dengan perangkat jaringan yang digunakan.



Gambar 3.1. Tahapan Pembangunan Sistem Jaringan *Load Balancing*

IV. Implementasi, Pengujian, dan Analisa Hasil

4.1. Implementasi

Instalasi merupakan aktivitas yang dilakukan pada masing-masing subsistem sesuai dengan kebutuhan fungsional dan kebutuhan nonfungsional.

Tabel 4.1. Lingkungan Instalasi Sistem

Subsistem	Kapasitas		
	Processor/ *Front Side Bus*	*Hard Disk*	*Memory*
Web Server 1	8 x Intel(R) Xeon(R) CPU E5620 @2.40GHz	34,4GB	2048MB
Web Server 2	8 x Intel(R) Xeon(R) CPU E5620 @2.40GHz	34,4GB	2048MB
Database Server	8 x Intel(R) Xeon(R) CPU E5620 @2.40GHz	20GB	2048MB
Load Balancer	8 x Intel(R) Xeon(R) CPU E5620 @2.40GHz	34,4GB	2048MB

4.1.1. Implementasi Kebutuhan Nonfungsional

Aktivitas implementasi kebutuhan nonfungsional terdiri atas instalasi sistem operasi dan instalasi perangkat lunak. Instalasi sistem operasi yang dilakukan pada tiap subsistem menggunakan Ubuntu *Server* 10.04 LTS-*Long-Term Support* 32 *Bit*. Kemudian menginstalasi perangkat lunak *Nginx* dan *Apache web server* pada subsistem.

```
http{
upstream annisa {
                        server 202.46.3.[redacted]
                        server 202.46.3.[redacted]
                        keepalive 1024;
                }

server {
            listen 80;
            #server_name ip
            server_name 202.46.3.[redacted]
            access_log
/var/log/nginx/202.46.3.[redacted].access.log;
            error_log
/var/log/nginx/202.46.3.[redacted].error.log;

location / {
            proxy_pass http://annisa;
          }
      }
   }
```

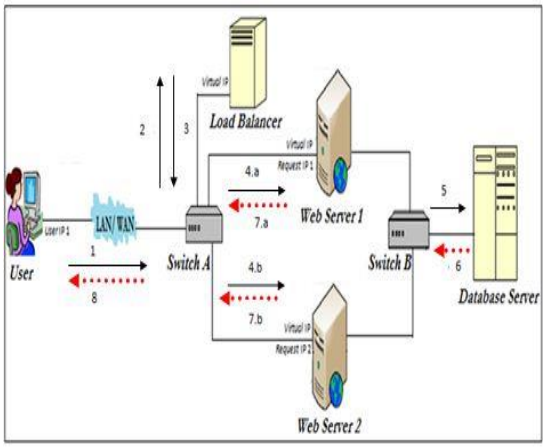
Gambar 4.1. *Load Balancer* dengan Algoritma *Round Robin*

4.1.2. Implementasi Kebutuhan Fungsional

Aktivitas pada pengimplementasian kebutuhan fungsional dengan melakukan konfigurasi dan pengaturan perangkat lunak yang diinstall. Implementasi kebutuhan fungsional pada subsistem menunjukkan masing-masing peran sebagai anggota dari sistem jaringan *load balancing* dan *virtual machine*.

4.2. *Simulation Prototyping*

Aktivitas *simulation prototyping* dimaksudkan untuk menjelaskan proses permintaan dari *user* sampai dijawab oleh *web server* atau *server* hingga kembali lagi ke *user*. Proses permintaan tersebut dapat dilihat pada gambar 4.2.



Gambar 4.2. Permintaan *User* pada *Web Server* atau *Server*

Gambar 4.2 dapat dijelaskan, yaitu proses pertama (1) permintaan *user* diawali saat *user* mengirimkan paket permintaan untuk layanan yang berjalan pada sistem operasi. *User* dengan alamat IP (UIP) melakukan permintaan terhadap halaman *website* (aplikasi penerimaan pegawai BPPT) menggunakan alamat tujuan *web server* virtual IP (VIP). Alamat resolusi protokol (ARP) *broadcast* pada *user* untuk mencari alamat VIP.

Pada proses kedua (2) *load balancer* menjawab ARP *broadcast* melalui alamat MAC (*Media Access Control*) dan memberitahukan bahwa VIP tersebut ialah milik *load balancer*. Lalu komputer *user* membuat paket yang memiliki alamat IP dan alamat tujuan VIP serta menyertakan alamat MAC milik *load balancer* yang kemudian dikirimkan ke *load balancer*. Paket tersebut akan diterima oleh *load balancer*.

Proses ketiga (3) ialah *load balancer* membuat dan mengirimkan paket menuju *web server* (4.a) dan (4.b) yang terpilih berdasarkan penjadwalan dari algoritma *round robin*. Alamat asal dan alamat tujuan pada paket tidak berubah hanya saja alamat MAC yang berubah dari alamat MAC *load balancer* menjadi alamat MAC *web server* yang dituju.

Proses 4.a dan 4.b, paket tersebut diterima oleh *web server* yang terpilih (misal *web server* 2) sesuai penerapan dari algoritma penjadwalan *round robin* yang kemudian pada proses kelima (5) *web server* 2 mengakses *database server* untuk melakukan permintaan data dalam memenuhi permintaan *user*.

Proses 6, 7(a) , 7(b), dan 8 alamat asal pada paket ialah VIP dengan alamat tujuannya UIP dengan *web server* 2 (*web server* yang terpilih) membuat paket baru sebagai jawaban atas paket permintaan dari *user*. Paket tersebut disalurkan dengan menggunakan *loopback device untuk* dikirimkan melalui *interface real server*.

4.3. Pengujian

Pengujian dilaksanakan untuk memastikan bahwa sistem jaringan yang dibangun telah memenuhi tujuan awal sistem jaringan.

Tabel 4.2. Spesifikasi Mesin Penguji

Spesifikasi	Nilai
Sistem Operasi	Ubuntu 11.10
Processor	Intel(R)Pentium(R) core"i5
Kapasitas Memori	2048Mb
Perangkat Lunak	*Apache 2 Webserver, Firefox Web Browser*

Pengujian kinerja *load balancer* dan *web server* dengan virtualisasi dilakukan dengan menggunakan akses beban koneksi terhadap permintaan dengan membandingkan *response time (mili second).* Perbandingan tersebut akan dilaksanakan pada saat sebelum menggunakan *load balancer* dengan setelah menggunakan *load balancer* guna mendapatkan data pembanding untuk kemudian dilakukan analisa. Akses data yang akan diuji, yaitu:

1) Akses Permintaan 100 beban koneksi.
2) Akses Permintaan 1.000 beban koneksi.
3) Akses Permintaan 5.000 beban koneksi.

Masing-masing pengujian diuji melalui program pengujian sederhana dengan menggunakan bahasa pemrograman *PHP* guna mendapatakn hasil pembanding sebelum dengan setelah terimplementasinya teknologi *load balancing* dan *virtual machine*.

4.3.1. Tabel Akses Permintaan Sebelum Pengujian *Load Balancer*

Tabel 4.3. Hasil Pengujian Sebelum *Load Balancing* dan *Virtual Machine*

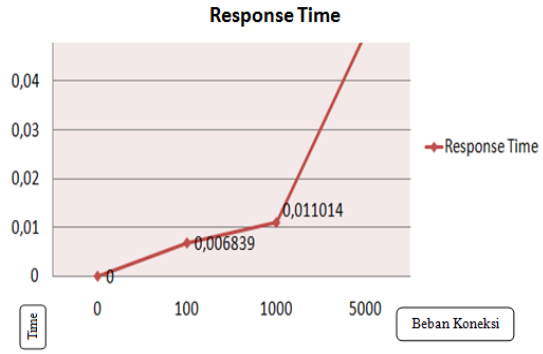
No	*Connection*	*Response Time (mili second)*
1	100	0,006839
2	1000	0,011014
3	5000	*over load*

Tabel 4.4. Hasil Pengujian Setelah *Load Balancing* dan *Virtual Machine*

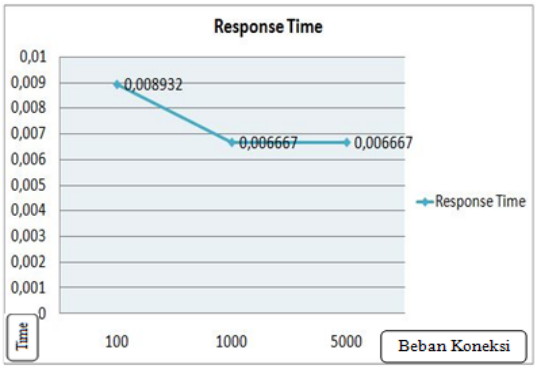
No	*Connection*	*Response Time Web Server 1 (mili second)*	*Response Time Web Server 2 (mili second)*	Rataan *Web Server 1 dan 2 (mili second)*
1	100	0,009110174	0,008754	0,008932
2	1000	0,00651	0,006757	0,006667
3	5000	0,010901	0,094123	0,050182

4.3.2. Analisa Hasil

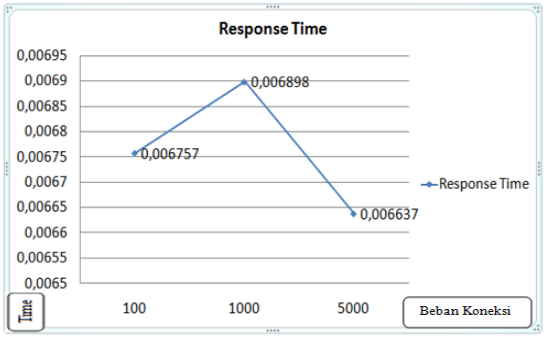
Perbandingan beban akses permintaan sebelum dan setelah menggunakan *load balancer* dari 100 hingga 5.000 untuk dianalisa, dapat dilihat pada gambar 4.3 dan 4.4.



Gambar 4.3. *Response Time* Sebelum *Load Balancer* dan *Virtual Machine*



Gambar 4.4. *Response Time* Sesudah *Load Balancer* dan *Virtual Machine* dengan Algoritma Round Robin



Gambar 4.5. *Response Time* Sesudah *Load Balancer* dan *Virtual Machine* dengan Algoritma *Round Robin Weight*

Analisa terhadap hasil perbandingan akses permintaan pada gambar 4.3, 4.4, dan gambar 4.5 dapat dijabarkan sebagai berikut:

a. Saat jumlah pengguna yang mengakses *web server* semakin banyak maka *response time* yang dihasilkan oleh *server* atau *web server* untuk melayani permintaan dari penggunaan semakin tinggi.
b. Sebelum diimplementasikannya teknologi *load balancing* dan *virtual machine*, pengguna tidak dapat mengakses aplikasi penerimaan pegawai dalam kurun waktu hampir bersamaan yang dimulai dari 753 beban akses dikarenakan *over load*.
c. Sesudah diimplementasikannya teknologi *load balancing* dan *virtual machine*, pengguna dapat mengakses aplikasi penerimaan pegawai dalam kurun waktu hampir bersamaan dengan stabil hingga kurang dari sama dengan 5000 beban akses.
d. Berdasarkan hasil perbandingan terhadap pengujian *response time* algoritma *round robin* dengan algoritma *round robin* weight pada aplikasi penerimaan pegawai BPPT dapat disimpulkan lebih baik menggunakan algoritma *round robin* karena beban koneksi yang diarahkan *load balancer* ke masing-masing *web server* seimbang, yaitu 50%.
e. Implementasi *load balancing* dan *virtual machine* dapat mengoptimalkan kinerja *server* atau web *server* pada sistem jaringan yang dibangun dalam studi kasus aplikasi penerimaan pegawai BBPT.

V. Kesimpulan

Kesimpulan yang didapatkan dari hasil implementasi dan penelitian dari pembangunan sistem jaringan *load balancing* dan *virtual machine* adalah:

a. Pembangunan sistem jaringan *load balancing* dan *virtual machine* dengan algoritma *round robin* merupakan solusi yang tepat dalam mengatasi kinerja *server* hingga kurang dari atau sama dengan 5.000 akses permintaan beban koneksi.
b. *Load balancer* mampu mengatur pembagian beban koneksi secara seimbang terhadap akses ke tiap *web server* sehingga masing-masing *web server* mendapatkan beban koneksi yang seimbang dari *load balancer*.
c. *Virtual machine* membantu mengoptimalkan keterbatasan perangkat keras yang tersedia, sehingga efektif tanpa perlu menambah biaya penyediaan perangkat keras baru.
d. Dengan menggunakan sistem jaringan *load balancing* dan *virtual machine*, aplikasi penerimaan pegawai di BPPT dapat lebih terjamin dalam melayani konektivitas dari para pengguna dalam menggunakan aplikasi penerimaan pegawai dalam jangka waktu akses yang relatif bersamaan.

REFERENSI

[1] Abdullah A., dkk., *Implementasi dan Analisa Load Balancing pada Suatu Web Server Lokal*. www.openstorage.gunadarma.ac.id, , 2003.
[2] Andrew S Tanenbaum., *Computer Networ*. Fourth Edition, Prentice Hall Publisher. 2003.
[3] Anonim, 2012, "Nginx", diakses dari http://trac.nginx.org/nginx/browser/nginx/branches/stable-1.2/src/http/ngx_http_upstream_round_robin.c pada tanggal 6 Agustus 2012, pukul 16.00 WIB.
[4] Bourke and Tony., *Server Load Balancing*. Sebastopol, O'Reilly. 2001.
[5] Dewobroto P., *Load Balance Menggunakan Metode PCC*. www.mikrotik.co.id. 2005.
[6] Hariyanto B., *Sistem Operasi*. Informatika, Bandung. 2007.
[7] James and Phillip., *Applied Data Communications*. John Wiley & Sons, Inc, United States of America. 2004.
[8] Kurniawan W., *Jaringan Komputer*. Andi, Yogyakarta. 2007.

[9] Kusumadewi S., *Sistem Operasi*. Graha Ilmu, Yogyakarta. 2002.
[10] Masyarakat Digital Gotong Royong (MDGR)., *Pengantar Sistem Operasi Komputer*. Masyarakat Digital Gotong Royong. 2007.
[11] Nedelcu Clement., *Nginx HTTP Server*. Packt Publishing, Birmingham-Mumbai. 2010.
[12] Rijayana Iwan., *Teknologi Load Balancing untuk Mengatasi Beban Server*. www.journal.uii.ac.id. 2005.
[13] Sembiring JH., *Jaringan Komputer Berbasis Linux*. Elex Media Komputindo, Jakarta. 2002.
Sopandi D., *Instalasi dan Konfigurasi Jaringan Komputer*. Informatika, Bandung. 2010.