IMPLEMENTASI LOAD BALANCING PADA WEB SERVER DENGAN MENGGUNAKAN APACHE

Supramana
D3 Manajemen Informatika, Fakultas Teknik, Universitas Negeri Surabaya, phram96@gmail.com

I Gusti Lanang Putra Eka Prismana
Jurusan Teknik Informatika, Fakultas Teknik, Universitas Negeri Surabaya, glan.putra@gmail.com

Abstrak

Perkembangan teknologi informasi yang semakin pesat, penggunaan web untuk mengakses informasi semakin banyak digunakan karena dengan mudah mendapatkan informasi yang dibutuhkan dalam waktu yang relatif cepat. Peningkatan permintaan tersebut menyebabkan *web server* sibuk melayani permintaan dan sangat memungkinkan *web server* tidak mampu melayani permintaan klien. Hal tersebut dapat mengakibatkan *web server* menjadi *overload* dan akhirnya *web server* menjadi *down*.

Untuk meminimalkan terjadinya *overload*, salah satu mekanisme untuk lebih mengoptimalkan sumber daya *web server* yang ada adalah dengan menggunakan *load balancing* yang akan menyeimbangkan beban pada suatu *web server* yang sibuk sehingga dapat mempercepat waktu respon dari web.

Berdasarkan hasil pengujian setelah diimplementasikan *apache* sebagai *load balancer* diperoleh bahwa setiap *web server* mendapatkan beban yang berurutan serta sama tanpa memiliki beban lebih, dan *web server* memiliki waktu tanggap lebih tinggi.

Kata Kunci : *Web Server, Load Balancing, Apache*.

Abstract

The development of information technology is rapidly increasing, the use of the web to access information more widely used because it is easy to get the information needed in a relatively quick time. The increase in demand led to a busy web server requests and so allow the web server is not able to serve client requests. This can lead to a web server becomes overloaded and eventually web server to be down.

To minimize the occurrence of overload, one mechanism to further optimize resources existing web server is to use load balancing which would balance the load on a busy web server so that it can speed up the response time of the web. Based on the results of the test after it is implemented as a load balancer apache obtained that every web server and get a load of the same sequence without a load, and the web server has a higher response time.

Key Words: *Web Server, Load Balancing, Apache*.

PENDAHULUAN

Seiring dengan berkembangnya teknologi informasi yang semakin pesat, penggunaan web untuk mengakses informasi pun semakin banyak digunakan karena dengan mudah pengguna mendapatkan informasi yang dibutuhkan dalam waktu yang relatif singkat. Peningkatan permintaan pada situs, menyebabkan *web server* sibuk menjawab permintaan dari klien dan sangat memungkinkan *web server* tidak mampu melayani permintaan dari *client*. Hal tersebut dapat mengakibatkan *web server* menjadi *overload*, lambat, dan akhirnya server menjadi *down*. Hal ini akan merugikan pihak yang mempercayakan situsnya pada suatu *web server*, karena situs-situs tersebut tidak dapat diakses untuk sementara waktu. Untuk mengatasi *overload* penyaji layanan web perlu mengupgrade *hardware server* ke performa yang lebih tinggi, Namun untuk solusi ini sepertinya hanya akan mengatasi masalah jangka pendek. Salah satu mekanisme untuk lebih mengoptimalkan sumber daya yang ada adalah dengan menggunakan *load balancing* yang akan menyeimbangkan beban pada *web server* yang sibuk, sehingga mempercepat waktu respon dari web.

Tujuan penelitian ini yaitu mengimplementasikan *Apache* sebagai *load balancing* pada *web server*. Manfaat dari penelitian ini yaitu mengetahui dan memahami cara kerja *load balancing* pada *web server* dengan menggunakan *apache*, sebagai salah satu solusi untuk meningkatkan kinerja *web server* dan menyeimbangkan beban web server yang sibuk.

KAJIAN PUSTAKA

Load Balancing

Load balancing adalah proses untuk mendistribusikan beban trafik pada dua atau lebih jalur

koneksi secara merata antara dua atau lebih computer, link jaringan, *CPU*, *hard drive*, atau sumber daya dengan tujuan untuk mendapatkan pemanfaatan sumber daya yang optimal, memaksimalkan *throughput*, memperkecil waktu tanggap dan menghindari *overload*. (Abdullah, dkk, 2010).

Load balancing digunakan pada saat sebuah *server* telah memiliki jumlah pengguna yang telah melebihi kapasitas maksimal dari *server* tersebut sesuai dengan kriteria dasar proses *load balancing* mampu mengurangi beban kerja setiap *server* sehingga tidak ada *server* yang memiliki kelebihan beban.

Cara Kerja *Load Balancing*

Pada dasarnya disaat *load balancer* menerima permintaan layanan dari pengguna, maka permintaan tersebut akan diteruskan ke *server* utama. Cara kerja pada *apache* sebagai *load balancing* sendiri adalah pada saat pengguna meminta halaman web dari *server*, *apache load balancing* akan menerima permintaan tersebut dan akan meneruskan permintaan dari pengguna menuju *server-server* yang menjadi anggota dari *apache load balancer*.

Algoritma *Load Balancing*

1. Round Robin
   Algoritma ini membagi beban secara berurutan dan merata dari satu *server* ke *server* lain.
2. Ratio
   Algoritma dengan parameter diberikan untuk masing-masing *server* yang akan dimasukkan kedalam sistem *load balancing*. *Server* dengan ratio terbesar akan diberi beban besar sedangkan *server* dengan ratio kecil akan diberi beban kecil.
3. Fastest
   Algoritma dengan mengutamakan *server-server* yang memiliki respon yang paling cepat pada saat permintaan masuk.
4. Least connection
   Algoritma membagi beban berdasarkan banyaknya koneksi yang sedang dilayani oleh *server*. *Server* yang memiliki koneksi paling sedikit akan melayani permintaan yang masuk.

Tipe *Load Balancing*

Dalam perancangan *load balancing* terdapat dua pilihan tipe *load balancer* sebagai berikut:

1. Software Load Balancing
   Load balancer yang sering digunakan berbasis perangkat lunak dimana *load balancing* berjalan disebuah *server* dan aplikasi *load balancing* diinstall dan perlu dikonfigurasi sebelum dapat berfungsi. Proses *load balancing* dipengaruhi oleh perangkat yang digunakan seperti kartu jaringan *(Network Interface Card)*, besarnya RAM *(Random Access Memory)* dan juga media penyimpanan.

2. Hardware Load Balancing
   Load balancer yang berjalan di sebuah alat yang siap digunakan. Tipe *load balancing* ini banyak digunakan karena kemudahannya.

Reverse Proxy

Terdapat beberapa jenis sistem penyeimbang beban *(load balancing)* dan salah satu jenisnya merupakan *proxy*. *Proxy* adalah sebuah sistem komputer atau program aplikasi yang melayani permintaan dari klien dengan meminta layanan ke *server* lain. (Ardhian, dkk, 2013).

Proxy server memiliki 3 fungsi utama yaitu :

1. *Connection sharing* : perantara *client* dan *server*.
2. *Filtering* : bekerja pada layer aplikasi yang dapat memblokir paket-paket tertentu.
3. *Caching* : mampu menyimpan informasi yang pernah diakses dari *server-server*.

Proxy di bagi menjadi 2 yaitu *forward proxy* dan *reverse proxy*. *Forward proxy* adalah *proxy* yang meneruskan data ke *host* tujuan. *Reverse proxy* adalah sebuah *proxy* yang berada di depan dari *web server*, digunakan sebagai *cache* atau bisa juga sebagai *load balancer*. *Reverse proxy* menjadi perantara user-user di internet terhadap akses ke web-web *server* yang berada pada *local area network*, sehingga seolah-olah user di internet mengakses langsung *web server* yang dimaksud padahal sesungguhnya user di internet mengakses web-web server yang terdapat di *local area network* melalui *reverse proxy* tersebut.

Web Server

Web server merupakan perangkat lunak yang melayani permintaan HTTP dari web browser dan mengirimkan kode-kode dinamis ke *server* aplikasi. Server inilah yang menerjemahkan dan memproses kode-kode dinamis menjadi kode-kode statis dalam suatu halaman statis yang kemudian dikirimkan ke browser oleh *web server*. *Web server* biasanya disebut juga dengan HTTP *server* karena menggunakan *protocol* HTTP. (Abdullah, dkk, 2010).

Penggunaan paling umum *web server* adalah untuk menempatkan situs web. Fungsi utama sebuah *web server* untuk mentransfer berkas atas permintaan pengguna melalui protokol komunikasi yang telah ditentukan dan mentransfer kedalam sebuah halaman web. Kriteria dasar web server berjalan dengan baik adalah dengan terjalinnya komunikasi misalnya

protokol HTTP/HTTPS dari *server* ke *client* atau sebaliknya tanpa ada data yang hilang.

Cara Kerja *Web Server*

Cara kerja dari *web server* adalah ketika klien meminta halaman web kepada *web server*, maka permintaan klien tersebut akan membuat sinyal sambungan *syncronize* dan dikirim ke alamat *web server*. Selanjutnya *web server* akan membalas dan mengirimkan sinyal *syncronize* dan *acknowledge* kepada klien dan kemudian klien akan membalas dengan sinyal *acknowledge* yang artinya sudah siap untuk mengirimkan data. Halaman web yang diminta dari klien ke *web server* disebut dengan *HTTP request* dan halaman web tersebut dikirim dari *web server* ke klien yang dikenal dengan *HTTP response*.

Apache

Apache merupakan program untuk menjalankan, melayani, dan memfungsikan situs web dalam sebuah komputer. (Emanuel, 2006). *Apache* dapat dijalankan di banyak sistem operasi *(BSD, Linux, Microsoft Windows* dan *Novell Netware* serta platform lainnya) karena merupakan perangkat lunak sumber terbuka dikembangkan oleh komunitas terbuka yang terdiri dari pengembang dibawah naungan *Apache Software Foundation*.

Apache memiliki fitur-fitur canggih seperti pesan kesalahan yang dapat dikonfigurasi, autentikasi berbasis basis data dan lain-lain. Selain itu *apache* juga bisa digunakan sebagai *load balancer* dengan menambahkan beberapa modul modul konfigurasi tertentu. Pada penelitian ini *apache* akan di gunakan sebagai *load balancer* adalah *apache* versi 2.4 karena *apache* pada versi 2.4 sudah mendukung modul untuk *load balancing* serta *space memory* yang digunakan lebih sedikit daripada versi sebelumnya.

GNS3 (*Graphical Network Simulator 3*)

GNS3 merupakan *software* simulasi jaringan berbasis GUI yang digunakan untuk merancang dan membangun jaringan komputer. Pada GNS3 memungkinkan simulasi jaringan yang kompleks, karena menggunakan *operating system* asli dari perangkat jaringan seperti *cisco* dan *juniper* sehingga kita berada kondisi lebih nyata dalam mengkonfigurasi secara langsung. (Dewannanta, 2013).

GNS3 adalah alat pelengkap yang sangat baik untuk laboratorium nyata bagi *network engineer*, karena GNS3 mendukung beberapa jenis aplikasi virtual lainnya seperti *Pemu, Qemu, Virtual Box* dan dapat mendukung virtual yang lainnya. Performa proses virtualisai menggunakan virtual box di GNS3 dipengaruhi oleh perangkat komputer yang digunakan. Untuk virtualisasi atau simulasi yang lebih kompleks, dibutuhkan spesifikasi perangkat *hardware* yang tinggi seperti besarnya RAM *(Random Access Memory)* dan media penyimpanan yang besar dan cepat untuk mendukungnya. *Virtual machine* yang akan digunakan sebagai program untuk penerapan *load balancing web server* adalah *Virtual Box*.

VirtualBox

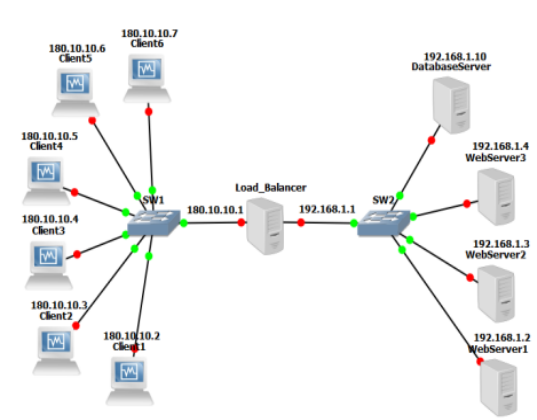
Virtual Box adalah sebuah perangkat lunak *open source* yang berfungsi untuk membuat virtualisasi ataupun simulasi dari *operating system* komputer di dalam sebuah *operating system* yang sedang berjalan tanpa mengganggu jalannya aktivitas *operating system* yang sebenarnya. (Pranantyo, 2012).

Virtual Box merupakan salah program yang memiliki kestabilan dan *interface* dalam proses instalasi sama dengan instalasi sistem operasi secara nyata, selain itu juga *virtual box* tidak membutuhkan partisi sendiri untuk melakukan sebuah instalasi tersebut. *VirtualBox* juga dapat digunakan sebagai media pembelajaran karena sifatnya yang praktis, aman, dan memiliki simulasi yang real.

METODE

Analisis Sistem

Dalam penelitian ini akan di implementasikan penggunaan *apache* sebagai *load balancing* pada *web server*. Berikut adalah arsitektur jaringan yang akan digunakan.



Gambar 1. Web Server Dengan Reverse Proxy

Berikut penjelasan gambar :

1. Web server
   Web server merupakan perangkat keras ataupun perangkat lunak. *Web server* memberikan *service* (layanan) kepada pengguna yang meminta informasi berkaitan dengan web melalui protokol komunikasi HTTP atau HTTPS.
2. Database server

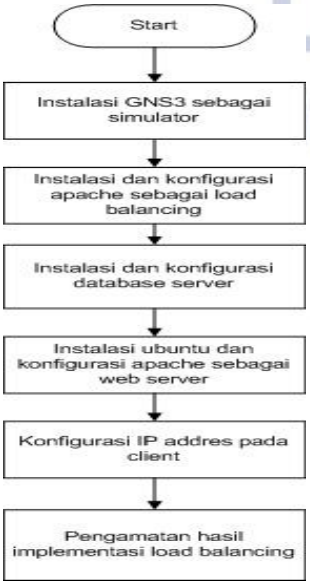
Database server merupakan sebuah program komputer yang menyediakan layanan pengelolaan basis data dan melayani komputer ataupun program aplikasi basis data yang menggunakan model *client/server*.

3. Reverse proxy (*Load Balancer*)
   Reverse proxy adalah sebuah *proxy* yang berada di depan dari *web server*, digunakan sebagai *cache* atau bisa juga sebagai *load balancer*. *Reverse proxy* menjadi perantara user-user di internet terhadap akses ke *web-web server* yang berada pada *local area network.*
4. Client
   Merupakan pengguna yang dapat mengakses sebuah sistem layanan yang berada di sistem atau komputer lain yang dikenal dengan *server* melalui jaringan komputer.

Perancangan Sistem

Penerepan sistem menggunakan permodelan jaringan untuk mensimulasikan sistem *load balancing* yang terdiri dari tiga komputer untuk *web server* serta satu komputer berfungsi sebagai *reverse proxy* atau *balancer*, serta beberapa komputer sebagai klien. Sistem yang diterapkan ini, diharapkan dapat menyeimbangkan beban pada *web server*.

Cara kerja *reverse proxy* sebagai *load balancing* adalah dengan memilih *web server* mana yang dipilih untuk melayani permintaan *user/client* ditentukan oleh konfigurasi aplikasi pada *server* penyeimbang beban. Ada beberapa algoritma dan parameter yang dapat menentukan pembagian beban/pemilihan web server tersebut. Dalam penerapan *load balancing* pada *web server* menggunakan *apache*, akan di bagi menjadi beberapa bagian perancangan sistem. Berikut adalah perancangan yang akan diterapkan :



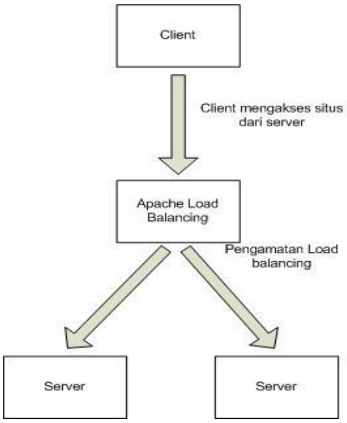
Gambar 2. Alur Perancangan Sistem

Berikut adalah penjelasan alur perancangan :

1. Instalasi GNS3
   Instalasi GNS3 adalah tahap awal dimana aplikasi GNS3 tersebut akan digunakan sebagai media simulasi untuk *implementasi load balancing web server* menggunakan *apache*.
2. Instalasi Ubuntu dan konfigurasi apache sebagai load balancer
   Instalasi Ubuntu dan konfigurasi *apache* merupakan tahap konfigurasi *apache* yang akan digunakan sebagai *load balancing* dengan konfigurasi modul-modul, algoritma dan parameter tertentu.
3. Instalasi dan konfigurasi database server
   Instalasi dan konfigurasi *database server* merupakan tahap untuk membangun pengelolahan basis data dari *web server*.
4. Instalasi Ubuntu dan konfigurasi apache sebagai web server
   Instalasi dan konfigurasi *apache* sebagai *web server* merupakan tahap untuk membangun sebuah web sederhana yang akan dipergunakan untuk pengujian *load balancing*.
5. Konfigurasi IP address
   Konfigurasi *IP address* adalah Pemberian alamat IP address akan disesuaikan sesuai pengalamatan IP pada jaringan.
6. Pengamatan hasil implementasi load balancing
   Proses pengujian dan pengamatan hasil *implementasi load balancing* menggunakan *apache* akan menentukan keberhasilan konfigurasi yang telah dilakukan. Pengamatan ini juga dilakukan untuk membuktikan *load balancing* dengan *apache* berhasil atau tidak.

Rencana Pengujian

Rencana proses pengujian yang akan dilakukan adalah dalam komputer *server* akan dibuat sebuah contoh situs web sederhana dan klien akan mencoba mengakses situs web yang ada pada *server* tersebut. Pengamatan dilakukan berdasarkan pada pembagian beban *web server* secara merata. Adapun gambaran pengujiannya sebagai berikut.



Gambar 3. Flowchart Cara Pengujian

Pertama-tama klien akan mengakses situs *web server* yang telah ditentukan. Permintaan dari klien akan masuk ke *apache load balancer*, dari *apache load balancer* permintaan klien akan diteruskan ke beberapa *server* dengan melakukan pembagian beban terhadap *server* lain. Pada tahap selanjutnya akan dilakukan pengamatan apakah pembagian beban di *apache load balancer* bekerja atau tidak.

Skenario Pengujian

Skenario Pengujian yang akan dilakukan sebagai tahap menganalisa hasil dari ujicoba. Pada *apache load balancing* akan diberikan konfigurasi *method by requests* dan juga *method by busyness* dengan masing-masing klien mengakses halaman web secara berurutan, secara acak dan juga secara *connect-disconnect*. Skenario pengujian penelitian adalah sebagai berikut:

1. Tes koneksi antara *web server* dengan *database server*.
2. Tes berjalannya *load balancer method by requests*.
3. Tes berjalannya *load balancer method by busyness*.

HASIL DAN PEMBAHASAN

Konfigurasi Apache Load Balancer

Install Apache2 pada Ubuntu menggunakan perintah *sudo apt-get install apache2*. Tambahkan modul *proxy* untuk mendukung *load balance*.

```
a2enmod proxy
a2enmod proxy_http
a2enmod proxy_balancer
a2enmod lbmethod_byrequests
a2enmod lbmethod_bybusyness
a2enmod headers
```

Tambahkan anggota *balancer* yang akan ditangani oleh *load balancer*. Setiap anggota dituliskan sebagai sebuah *url end point* yang nantinya akan diakses oleh klien.

```
ProxyRequests Off
ProxyPreserveHost On

Header add set-Cookie
"ROUTEID=.%{BALANCER_WORKER_ROU
TE}e; path=/"
env=BALANCER_ROUTE_CHANGED

<Proxy balancer://webta>
BalancerMember http://192.168.1.2 route=1
BalancerMember http://192.168.1.3 route=2
BalancerMember http://192.168.1.4 route=3

ProxySet stickysession=ROUTEID
lbmethod=byrequests

</Proxy>

ProxyPass / balancer://webta/
ProxyPassReverse / balancer://webta/
```

Gambar 4. Kofigurasi Apache Load Balancer

Konfigurasi Database server

Pada *database server* berikan alamat *ip address* dan *install mysql*.

```
# This file describes the network interfaces available
on your system
# and how to active them. For more information, see
interfaces(5)

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.1.10
netmask 255.255.255.0
```

Gambar 5. Konfigurasi Ip Address Database Server

Agar *database* bisa diakses *webserver* dalam satu jaringan maka perlu mengkonfigurasi file *my.cnf* yang berisi *bind-address* pada *directory /etc/mysql/my.cnf* menjadi *ip address server*.

```
# Instead of skip-networking the default is now to
listen only on
# localhost which is more compatible and is not less
secure
bind-address            = 192.168.1.10
#
```

Gambar 6. Konfigurasi Bind Address

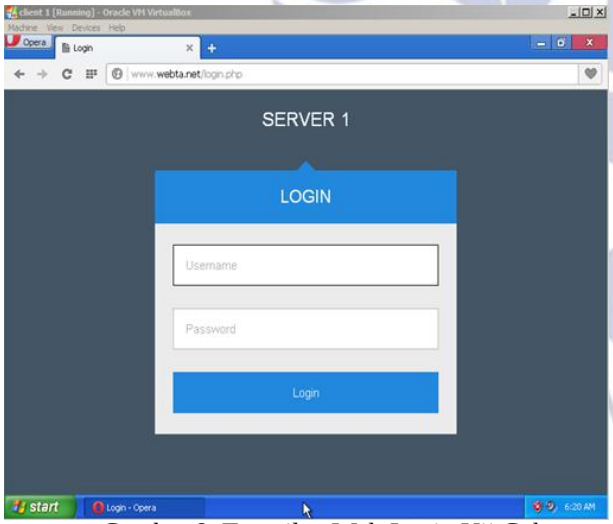
Konfigurasi Web Server

Pada ketiga *real server* di *install apache web server* dan dibuat sebuah halaman web sederhana yang bertugas untuk menerima dan membalas *request* yang datang dari klien. Instalasi *apache* dapat dilakukan dengan *apt-get install apache2*, beri *ip address* pada masing-masing *web server*, untuk *webserver* 1 = 192.168.1.2, *web server* 2 = 192.168.1.3, dan untuk *web server* 3 = 192.168.1.4

```
# interfaces(5) file used by ifup(8) and  ifdown(8)
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.1.2
netmask 255.255.255.0
```

Gambar 7. Konfigurasi Interface Eth0 Web Server

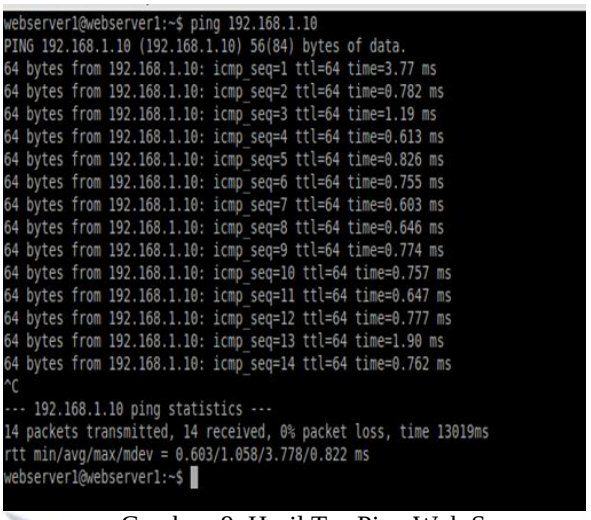


Gambar 8. Tampilan Web Login Uji Coba

Pengujian dan Pembahasan

1. Tes Koneksi Web Server dan Database server

Pengujian koneksi antara *web server* dan *database server* dilakukan untuk mengetahui hasil dari konfigurasi yang telah dilakukan sebelumnya pada kedua *server* apakah kedua *server* terhubung tau tidak dengan melakukan tes *ping* dari *web server* ke *database server*. Berikut hasil dari tes *ping*:

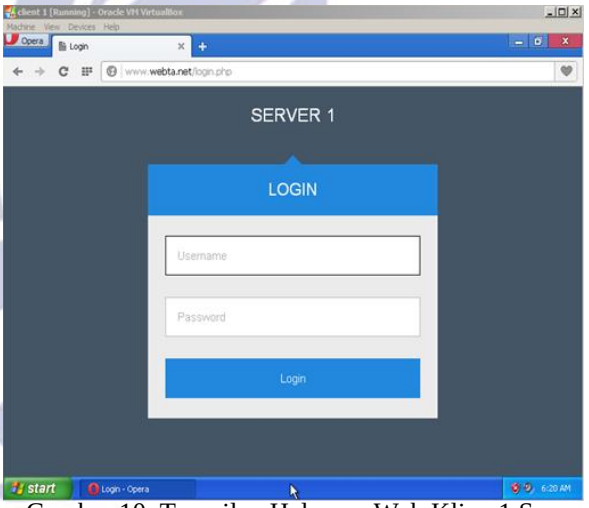


Gambar 9. Hasil Tes Ping Web Server

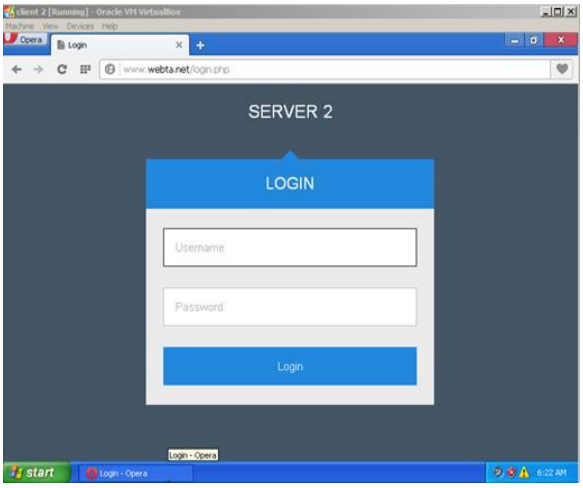
2. Tes Berjalannya Load Balancer By Requests

a. Akses Klien Secara Berurutan

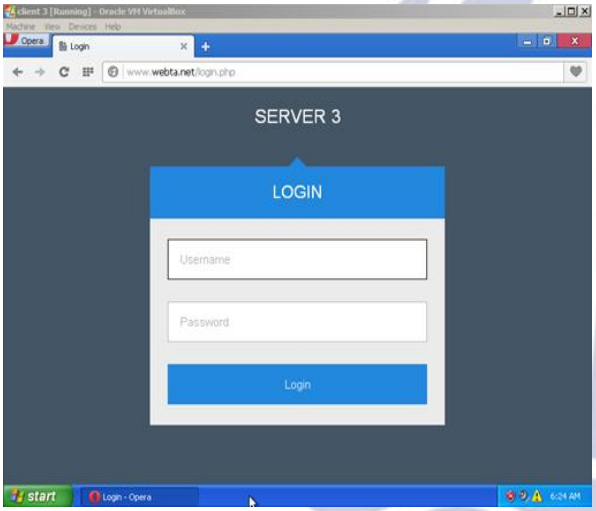
Pengujian berjalannya *load balancer* dilakukan dengan skenario beberapa komputer klien mencoba mengakses halaman web yang telah sebelumnya dibuat pada *web server* secara berurutan.



Gambar 10. Tampilan Halaman Web Klien 1 Secara Berurutan



Gambar 11. Tampilan Halaman Web Klien 2 Secara Berurutan



Gambar 12. Tampilan Halaman Web Klien 3 Secara Berurutan

Untuk memudahkan melihat hasil dari pengujian *load balancer* diatas, dibuatlah sebuah tabel sebagai berikut:

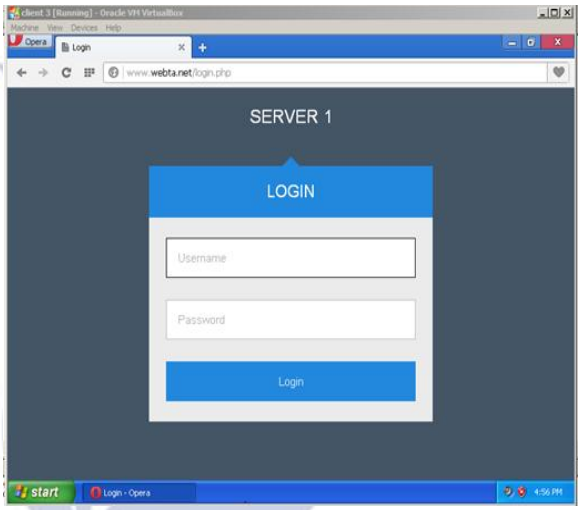
Tabel 1. Hasil Pengujian Load Balancer by Request Secara Berurutan

	Web Server 1	**Web Server 2**	**Web server 3**
Klien 1			
Klien 2			
Klien 3			
Klien 4			
Klien 5			
Klien 6			

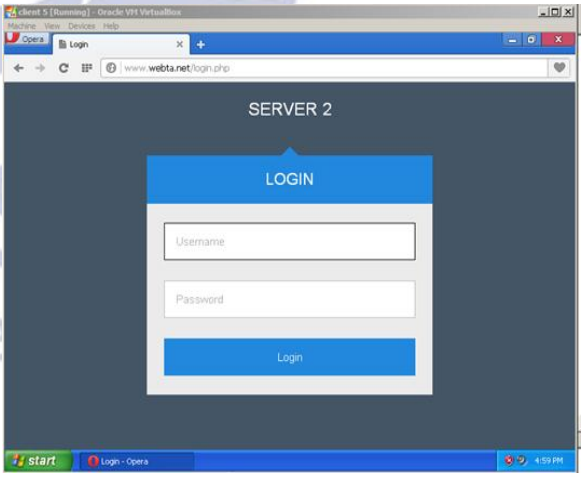
Dari tabel hasil pengujian *load balancer* diatas menunjukan bahwa *web server* 1 melayani *request* dari klien 1 dan klien 4, *web server* 2 melayani *request* dari klien 2 dan klien 5,dan *web server* 3 melayani *request* dari klien 3 dan klien 6. Ketiga *web server* tersebut mendapatkan *request* yang sama dan seimbang.

b. Akses Klien Secara Acak

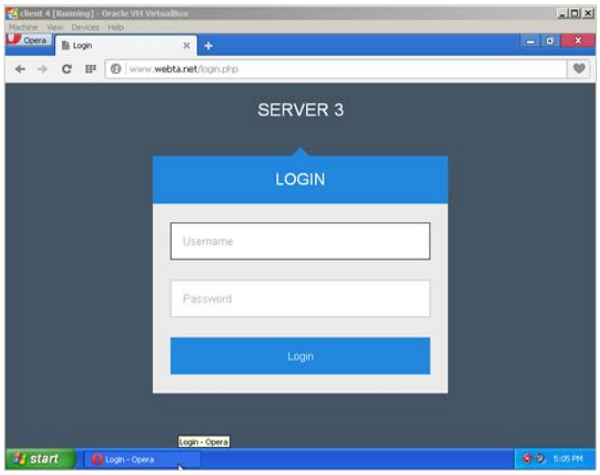
Pengujian berjalannya *load balancer* dilakukan dengan skenario beberapa komputer klien mencoba mengakses halaman web yang telah sebelumnya dibuat pada *web server* secara acak.



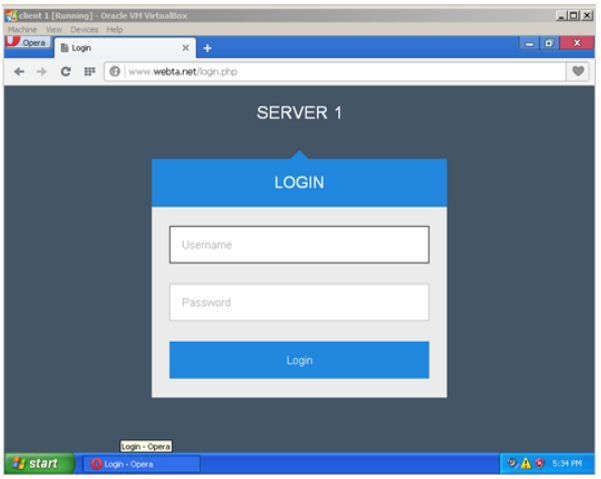
Gambar 13. Tampilan Halaman Web Klien 1 Secara Acak



Gambar 14. Tampilan Halaman Web Klien 2 Secara Acak



Gambar 15. Tampilan Halaman Web Klien 3 Secara Acak

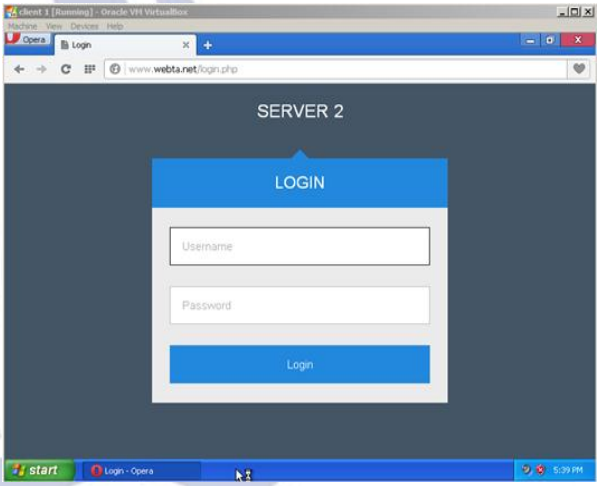


Gambar 16. Tampilan Halaman Web Klien 1 Secara Terputus

Untuk memudahkan melihat hasil dari pengujian *load balancer* dengan skenario klien mengakses secara acak diatas, dibuatlah sebuah tabel sebagai berikut:

Tabel 2. Hasil Pengujian Load Balancer by Request Secara Acak

	Web Server 1	**Web Server 2**	**Web server 3**
Klien 1			
Klien 2			
Klien 3			
Klien 4			
Klien 5			
Klien 6			

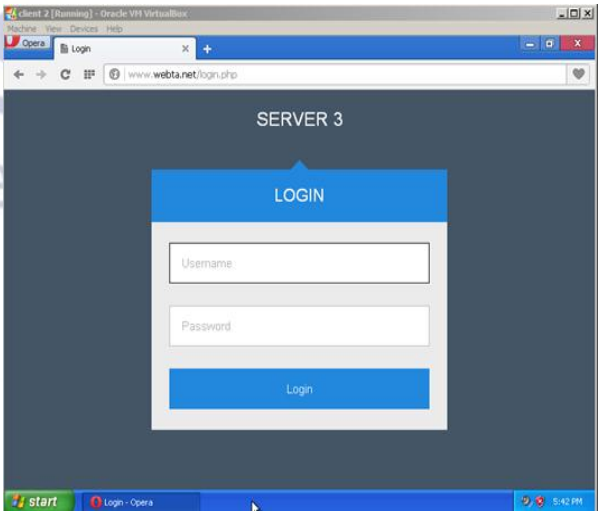


Gambar 17. Tampilan Halaman Web Klien 1 Terhubung Kembali

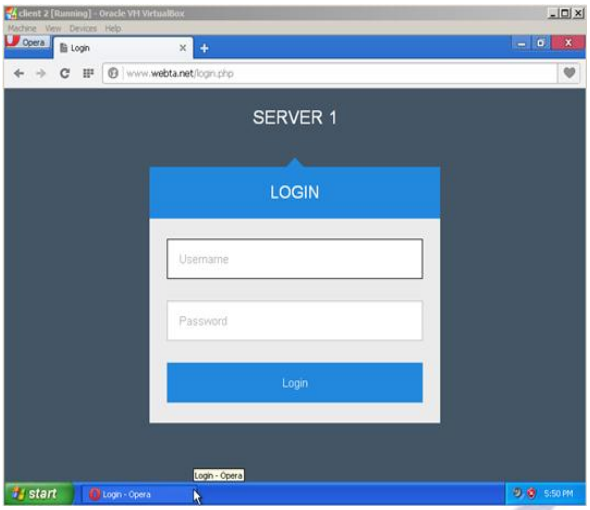
Dari tabel hasil pengujian *load balancer* diatas menunjukan bahwa *web server* 1 melayani *request* dari klien 1 dan klien 3, *web server* 2 melayani *request* dari klien 5 dan klien 6, dan *web server* 3 melayani *request* dari klien 2 dan klien 4. Ketiga *web server* tersebut mendapatkan *request* yang sama.

c. Akses Klien Terputus

Pengujian berjalannya *load balancer* dilakukan dengan skenario beberapa komputer klien mencoba mengakses halaman web yang telah sebelumnya dibuat pada *web server* secara terputus.



Gambar 18. Tampilan Halaman Web Klien 2 Secara terputus



Gambar 19. Tampilan Halaman Web Klien 2 Terhubung Kembali

Untuk memudahkan melihat hasil dari pengujian *load balancer* dengan skenario klien mengakses secara terputus diatas, dibuatlah sebuah tabel sebagai berikut:

Tabel 3. Hasil Pengujian Load Balancer by Request Secara Terputus

	Sebelum koneksi terputus	**Setelah koneksi terhubung kembali**
Klien 1	Server 1	Server 2
Klien 2	Server 3	Server 1
Klien 5	Server 2	Server 3
Klien 3	Server 1	Server 2
Klien 6	Server 3	Server 1
Klien 4	Server 2	Server 3

Dari tabel hasil pengujian *load balancer* diatas menunjukan bahwa *web server* 1 melayani 4 *request*, *web server* 2 melayani 4 *request*, dan *web server* 3 melayani 4 *request*. Ketiga *web server* tersebut mendapatkan *request* yang sama.

PENUTUP

Simpulan

Dari hasil pengujian diperoleh bahwa setelah diimplementasikan *apache* sebagai *load balancer* maka didapatkan *apache load balancer* berdasarkan *method by requests* akan meneruskan setiap *request* dari klien ke beberapa *web server* secara berurutan dan bergantian sehingga setiap *web server* mendapatkan *request* yang sama tanpa ada salah satu *web server* yang memiliki beban lebih. Sedangakan pada *apache load balancer* berdasarkan *method by busyness* apabila salah satu *web server* memiliki waktu respon yang rendah maka *apache load balancer* akan mengarahkan *request* dari klien ke *web server* yang memiliki waktu respon yang tinggi.

Saran

Saran untuk pengembangan dan penelitian selanjutnya adalah perlu dilakukan penambahan sebuah sinkronisasi antara *web server* agar apabila salah satu *web server* terjadi *update data* ataupun *file* maka *web server* yang lain akan terupdate secara otomatis.

DAFTAR PUSTAKA

Abdullah, A, S.N.M.P Simamora, dan H. R. Andrian. 2010. *Implementasi* dan Analisa *Load Balancing* pada suatu *Web Server* Lokal.

Apache Software Foundation. 2016. Apache Http Server (online), (http://httpd.apache.org/docs/2.4/mod/mod_proxy.html, diakses 30 april 2016).

Ardhian, Dite, Adian F. R, dan Eko Didik W. 2013. Analisis perbandingan untuk keja penyeimbang beban *Web Server* dengan *Haproxy* dan *PoundLinks*. *Jurnal Teknologi dan Sistem Informasi* Vol. 1, No. 2.

Dewannanta, Didha. 2013. Mengenal Software Simulator Jariangan Komputer GNS3 (online), (http://ilmukomputer.org/wp-content/uploads/2013/01/gns3.pdf, diakses 30 mei 2016).

Emanuel, Andi Wahyu Rahardjo. 2006. Instalasi Apache Web Server, MySQL Database, dan PHP pada Sistem Operasi Fedora Core 5. *Jurnal Informatika UKM* Vol 2, No. 3.

Pranantyo, Agung. 2012. Penggunaan Media Pembelajaran Virtual Box sebagai Upaya Untuk Meningkatkan Kemampuan Siswa Dalam Melakukan Instalasi Sistem Operasi di SMK Negeri 2 Pengasih. *Jurnal Elektronik Pendidikan Teknik Inormatika* Vol. 1, No.1.

Rijayana, Iwan. 2005. "Teknologi Load Balancing Untuk Mengatasi Beban Server". Disajikan dalam Seminar Nasional Aplikasi Teknologi Informasi 2005 (SNATI 2005), Yogyakarta, 18 juni.