ANALISIS PERBANDINGAN PERFORMANSI LOAD BALANCING MENGGUNAKAN ALGORITMA ROUND ROBIN DAN ALGORITMA LEAST CONNECTION PADA WEB SERVER DI JARINGAN UNIVERSITAS BRAWIJAYA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

Memperoleh gelar Sarjana Teknik

Disusun oleh:

LASTONO RISMAN H.

NIM.0510630057 - 63

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2012

PENGANTAR

Dengan mengucapkan Alhamdulillahi Rabbil'alamin, puji syukur kepada Allah SWT yang telah melimpahkan rahmat, ridho, taufik dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "Analisis Perbandingan Performansi Menggunakan Algoritma Round Robin dan Least Connection Pada Web Server di Jaringan Universitas Brawijaya" sebagai salah satu syarat yang diajukan untuk memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya Malang.

Penulis menyadari bahwa penulisan skripsi ini tidak akan terselesaikan tanpa adanya bantuan dari berbagai pihak karena itu ucapan terima kasih penulis sampaikan kepada semua pihak yang telah dengan rela meluangkan waktunya untuk membantu penulis dalam proses penulisan skripsi dan juga dengan ikhlas memberikan bimbingan, arahan, masukan, dan semangat sehingga penulis dapat menyelesaikan skripsi ini. Oleh karena itu pada kesempatan ini, penulis mengucapkan terima kasih kepada:

1. Bapak Dr. Ir. Sholeh Hadi Pramono., MS. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
2. Bapak M. Azis Muslim, S.T., M.T., Ph.D selaku Sekretaris Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
3. Bapak Moch. Rif'an. ST., MT. selaku Ketua Program Studi Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
4. Bapak Waru Djuriatno ST., MT. selaku Ketua Kelompok Dosen Keahlian Rekayasa Komputer Jurusan Teknik Elektro Universitas Brawijaya.
5. Bapak Raden Arief Setyawan, ST., MT dan Adharul Muttaqin ST., MT. selaku dosen pembimbing I dan dosen pembimbing II yang telah banyak memberikan bimbingan, masukan dan arahan dalam penyusunan skripsi ini.
6. Bapak Rudy Yuwono, ST., MSc. selaku dosen penasehat akademik yang telah memberikan nasehat dan arahan dalam proses akademik penulis.
7. Keluarga besar R. Pamudji H.yang telah mendorong untuk menyelesaikan skripsi.

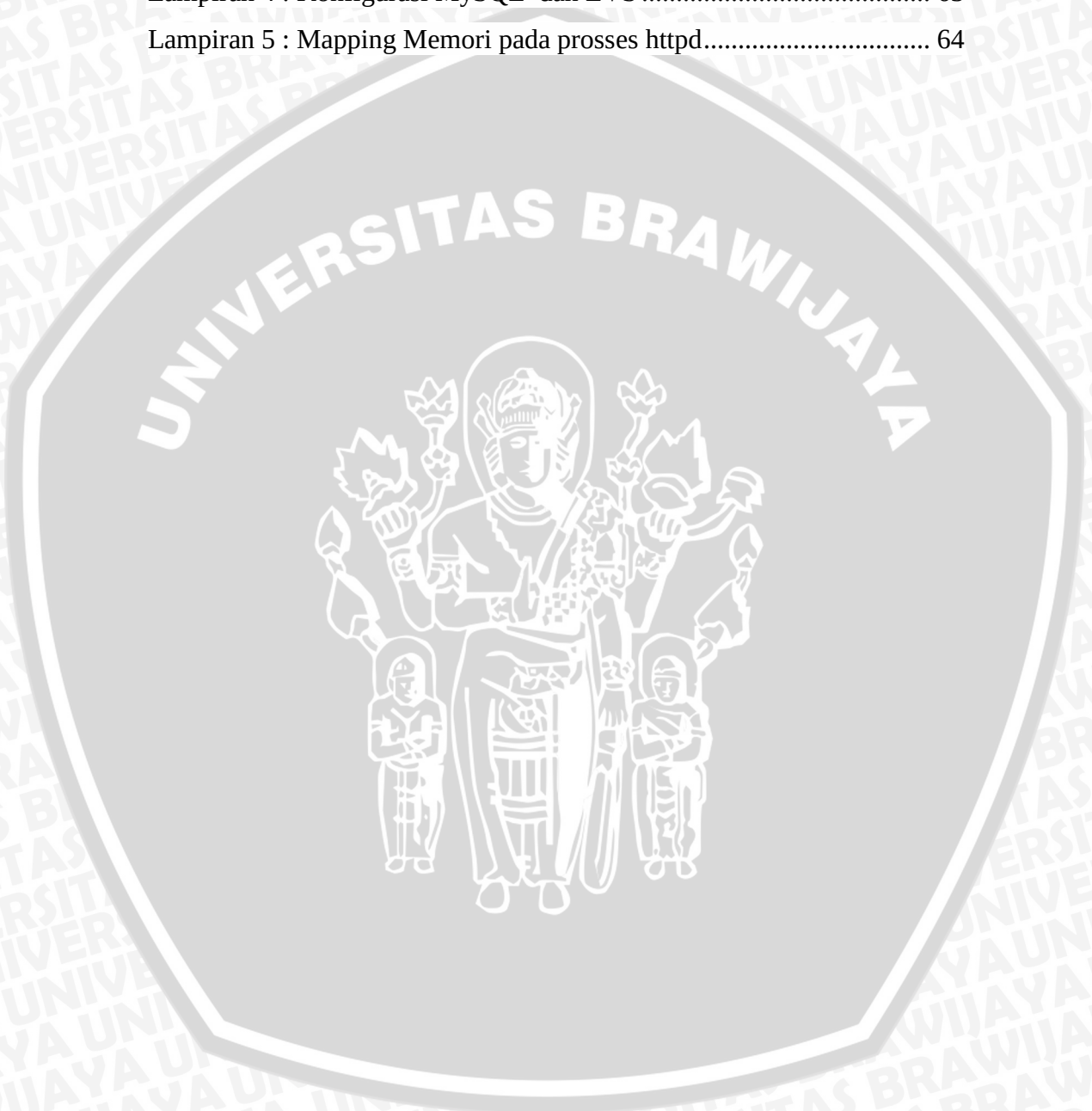
8. Keluarga besar PPTI yang memberikan support dan ilmu untuk penyelesaian skripsi ini.
9. Ferdian Adipta, Gilang Putra, Eryc ”cabul” Tri Juni, Rio Mareta beserta para pengunggu kafet lainnya telah membantu
10. Semua pihak yang telah membantu penyelesaian skripsi ini baik secara langsung dan tidak langsung yang tidak dapat penulis sebutkan satu per satu.

Penulis berharap semoga skripsi ini bermanfaat bagi pembaca dan dapat memberikan sumbangan pikiran bagi pihak-pihak lain khususnya mahasiswa jurusan Teknik Elektro. Segala kritik dan saran yang penulis terima demi kesempurnaan skripsi ini karena penulis menyadari bahwa penulisan skripsi ini masih jauh dari sempurna dan masih banyak memiliki kekurangan.

Malang, Juli 2012

Penyusun

DAFTAR ISI



DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

ABSTRAK

LASTONO RISMAN H, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Juli, 2012. *Analisis Perbandingan Performansi Load Balancing Mrnggunakan Algoritma Round Robin dan Algoritma Least Connection Pada Web Server Di Jaringan Universitas Brawijaya*.Dosen Pembimbing: Raden Arief Setyawan, ST., MT. dan Adharul Muttaqin, ST., MT.

Universitas Brawijaya sebagai salah satu universitas terbesar di Indonesia mempunyai sejumlah portal layanan informasi yang diakses oleh orang. Pengakses web di Universitas Brawijaya dalam bulan Juli 2010 mencapai rata-rata 7266 pengunjung dalam satu hari, dan mencapai 618 orang pada jam tertentu.

Untuk mengatasi beban berlebih pada layanan web server ketika beban puncak maka dilakukan load balancing. Load balancing dapat menggunakan metode Linux Virtual Server (LVS). Pada load balancing ini terdiri atas dua bagian antara lain director dan real server. Jadi ketika ada request maka semua request akan melewati director untuk didistribusikan kepada real server sesuai dengan algoritma yang digunakan.

Untuk single server pada saat beban 300 user diberikan untuk prosentase error, waktu respon dan troughput bernilai 4.89% dari total *request*, 8723 ms dan 4.73 kbps, sedangkan untuk round robin 2,99 % dari total *request*, 8582 ms dan 4.98 kbps, sedangkan untuk least connection 0,9% dari total *request*, 8023 ms dan 5.38. Sehingga penggunaan load balancing lebih handal jika dibandingkan dengan menggunakan single server, dan dari hasil pengujian didapatkan bahwa least connection lebih stabil jika dibandingkan algoritma round robin.

Kata kunci: *web server, load balancing, LVS, round robin, least connection*

BAB I
PENDAHULUAN

1.1. Latar Belakang

Pada dewasa ini perkembangan internet sangat cepat. Kebutuhan akan internet seakan bukan hal yang aneh lagi. Kebutuhan akan informasi memang sangat tinggi. Bahkan tidak sedikit orang yang memanfaatkan peralatan lain selain komputer untuk dapat berinteraksi di dunia maya.

Internet sebagai portal informasi sangat luas cakupannya. Mulai paling kecil sampai paling besar disediakan di internet. Mulai harga paku sampai harga rumah ada semua, dari hal sepele hingga berita internasional juga ada di internet.

Sebagai penyedia layanan informasi, situs web tidak boleh mati atau overload. *web server* merupakan mesin dimana tempat aplikasi atau software beroperasi dalam medistribusikan web page ke *client*, tentu saja sesuai dengan permintaan *client*. [1]

Apache merupakan *web server* yang terpopuler saat ini. Menurut survei Netcraft lebih dari 50% situs di Internet menggunakan Apache sebagai *web server*. Apache memenuhi standar *HTTP*/1.1, mengimplementasikan protokol terbaru, sangat fleksibel dikonfigurasi dan dapat dengan mudah ditambahkan modul lainnya dalam bentuk *module*, serta tersedia untuk berbagai sistem operasi [2]

Universitas Brawijaya sebagai salah satu universitas terbesar di Indonesia dengan jumlah mahasiswa 30.278 orang, mempunyai portal layanan informasi yang banyak diakses oleh orang. Web Universitas Brawijaya selama bulan Juli 2010 dalam satu hari rata-rata mempunyai 7266 pengunjung dalam satu hari. Pengguna internet di Universitas Brawijaya pada jam sibuk mencapai 618 orang pada jam tertentu. [3]

Dengan semakin bertambahnya pengguna Internet, maka *web server* akan memiliki lalu lintas data semakin besar. Hal ini dapat mengakibatkan ketidakmampuan *web server* (dalam hal ini adalah Apache) untuk melayani *request* dari *client*. Hal ini dikenal dengan istilah kelebihan beban (*overloading*).

Salah satu solusi yang dapat dilaksanakan adalah melakukan pendistribusian beban *request HTTP* kepada beberapa *web server* (*Load balancing*).

Pada implementasinya, *client* yang mengakses tidak perlu mengetahui bahwa *request* yang dikirimkan sebenarnya tidak hanya diproses oleh satu buah *web server* saja. Hal ini terjadi karena semua *request HTTP* yang datang selalu diarahkan menuju satu buah *web server* saja yang berfungsi sebagai *director*, kemudian oleh *director* tersebut bertugas untuk mendistribusikan *request* menuju *web server* lainnya (*real server*). Respon dari *director* terhadap sebuah *request HTTP* akan dikirimkan kembali menuju *director* untuk kemudian di-*forward* kepada *client* sebagai respon terhadap *request* yang dikirimkan

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan diatas, maka dapat dirumuskan masalah sebagai berikut :

1. Perancangan dan penerapan metode *Load balancing* pada *web server* dengan menggunakan algoritma *Round robin*
2. Perancangan dan penerapan metode *Load balancing* pada *web server* dengan menggunakan algoritma *Least connection (LC)*
3. Membandingkan performansi *Least connection (LC)* dengan *Round robin* pada jaringan Universitas Brawijaya.
4. Menganalisa unjuk kerja dengan menggunakan *Load balancing* dengan algoritma *Round robin* dengan algoritma *Least connection (LC).*

1.3. Batasan Masalah

Mengacu pada permasalahan yang sudah dipaparkan pada rumusan masalah maka pembatasan masalah pada tugas akhir ini antara lain

1. Parameter yang diuji adalah keberhasilan akses website, error *request*, dan troughput.
2. Algoritma *Load balancing* yang digunakan adalah *Round robin* dan *Least connection (LC).*
3. Jaringan yang digunakan adalah jaringan Universitas Brawijaya.

1.4. Tujuan Penulisan

Tujuan dalam penelitian ini adalah menerapkan metode *Load balancing* yang terbaik pada *web server* untuk meningkatkan unjuk kerja layanan aplikasi yang berbasis *web* di Universitas Brawijaya, serta menganalisa hasil peningkatan unjuk kerja yang didapatkan.

1.5. Manfaat

Manfaat penelitian ini antara lain :

1. Sebagai bahan pertimbangan dalam penerapan *Load balancing web server* di Universitas Brawijaya.
2. Meningkatkan performa *web server* dalam menangani *request* dari *client*.
3. Menjaga stabilitas *web server* ketika menangani *request* dalam jumlah banyak..

1.6. Sistematika Penulisan

Sistematika penulisan dalam skripsi ini sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi pembahasan, dan sistematika pembahasan.

BAB II Dasar Teori

Membahas teori-teori yang mendukung dalam perancangan dan pembuatan aplikasi.

BAB III Metodologi

Berisi tentang metode penelitian yang digunakan dalam perancangan dan pengujian sistem.

BAB IV Analisis Kebutuhan dan Perancangan

Membahas tentang analisa kebutuhan dari sistem dan kemudian merancang hal-hal yang berhubungan dengan analisa tersebut.

BAB V Implementasi

Bagian ini berisi penjelasan tentang implementasi yang telah dilakukan (sistem operasi, perangkat keras dan perangkat lunak) dan batasan-batasan implementasi.

BAB VI Pengujian dan Analisis

Bagian ini berisi penjelasan tentang strategi pengujian dan teknik pengujian yang dilakukan. Dijelaskan juga seluruh kasus uji beserta hasil pengujiannya.

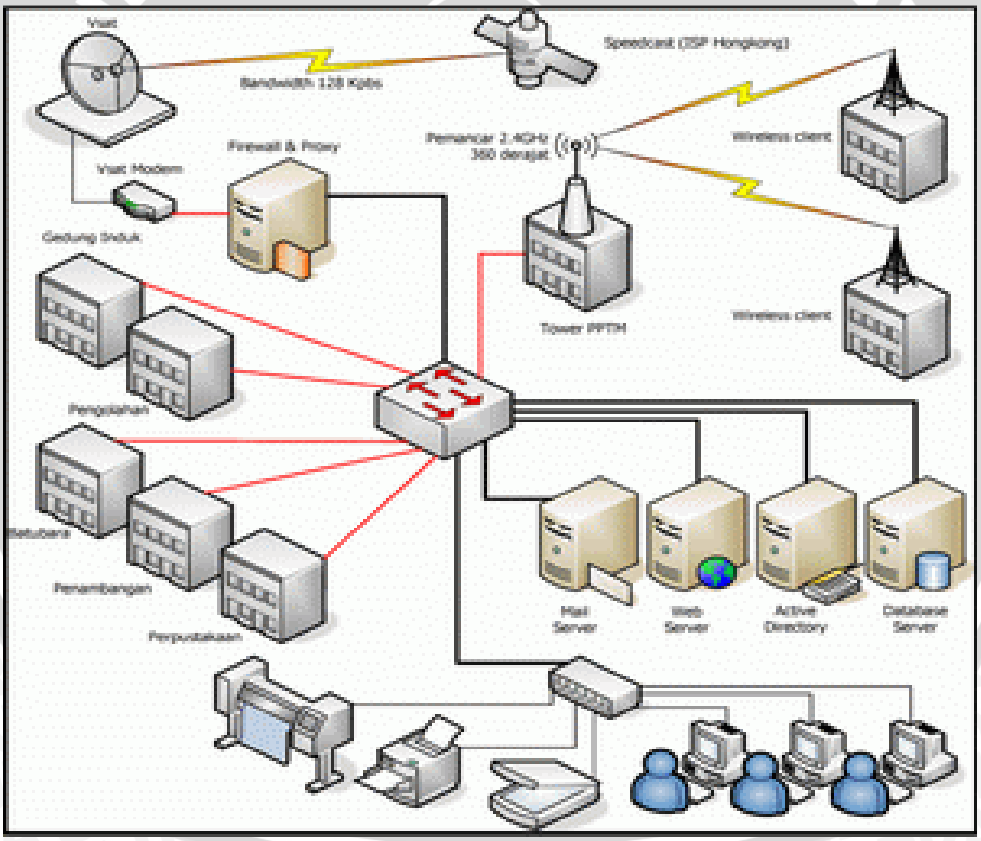
BAB VII Penutup

Bagian ini berisi kesimpulan dan saran. Kesimpulan didasarkan atas pengujian dan analisis yang dilakukan di dalam proses penelitian.

BAB II
TINJAUAN PUSTAKA

2.1. Kajian Pustaka

Jaringan Komputer adalah sekelompok komputer otonom yang saling berhubungan antara satu dengan lainnya menggunakan protokol komunikasi melalui media komunikasi sehingga dapat saling berbagi informasi, program – program, penggunaan bersama perangkat keras seperti *printer*, *harddisk*, dan sebagainya. Selain itu jaringan komputer bisa diartikan sebagai kumpulan sejumlah terminal komunikasi yang berada diberbagai lokasi yang terdiri dari lebih satu komputer yang saling berhubungan. [4]

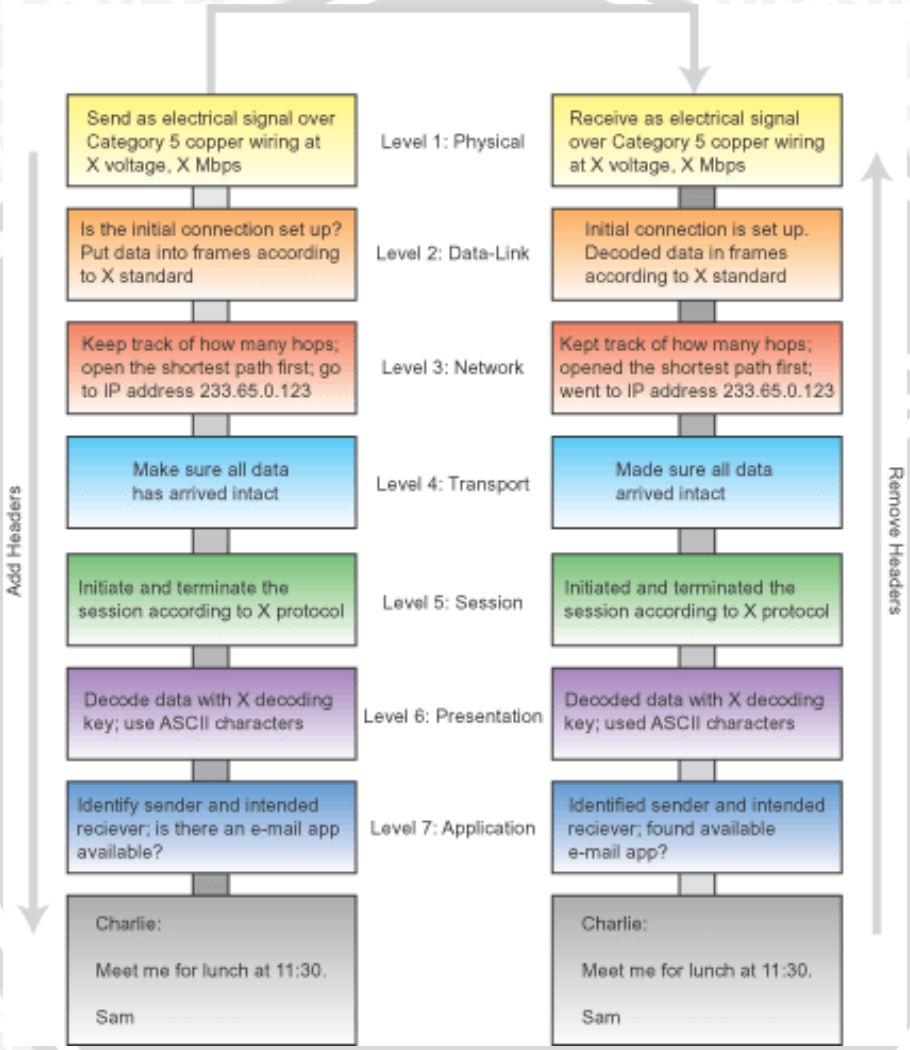


Gambar 2.1 Jaringan Komputer

Sumber: [EDW-10]

2.1.1. Protokol Komunikasi

Protokol adalah sebuah aturan atau standar yang mengatur atau mengijinkan terjadinya hubungan, komunikasi, dan perpindahan data antara dua atau lebih titik komputer. Protokol dapat diterapkan pada perangkat keras, perangkat lunak atau kombinasi dari keduanya. Pada tingkatan yang terendah, protokol mendefinisikan koneksi perangkat keras [5].



Gambar 2.2 Protokol OSI Layer
Sumber: Christodonte (2008)

Protokol perlu diutamakan pada penggunaan standar teknis, untuk menspesifikasi bagaimana membangun komputer atau menghubungkan peralatan perangkat keras. Protokol secara umum digunakan pada komunikasi *real-time* dimana standar digunakan untuk mengatur struktur dari informasi untuk penyimpanan jangka panjang. [5]

2.1.2. Model OSI

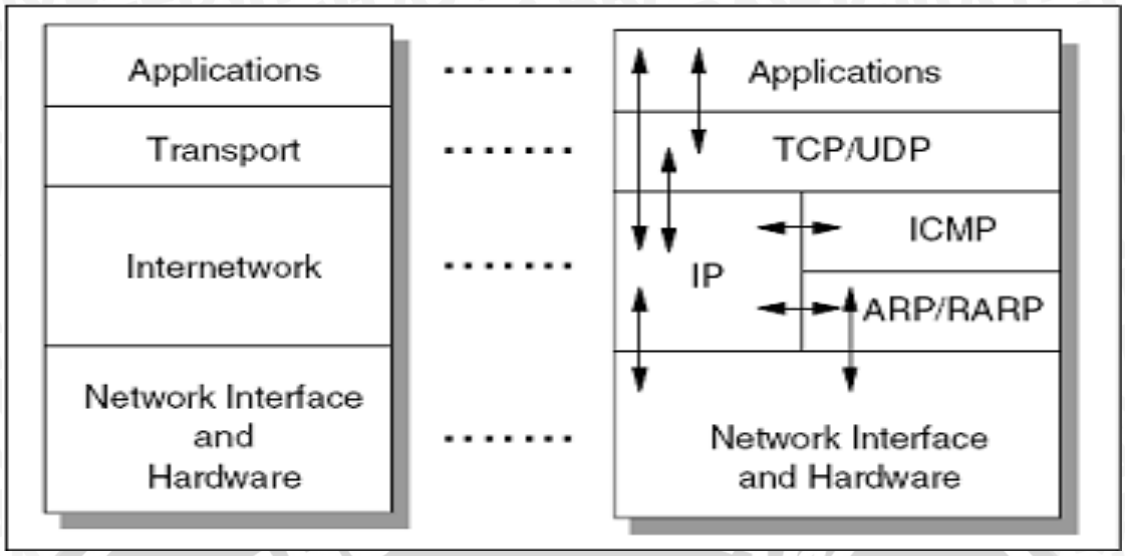
Model referensi jaringan terbuka OSI atau *OSI Reference Model for open networking* adalah sebuah model arsitektural jaringan yang dikembangkan oleh badan International Organization for Standardization (ISO) di Eropa pada tahun 1977. OSI sendiri merupakan singkatan dari *Open System Interconnection*. Model ini disebut juga dengan model "Model tujuh lapis OSI" (*OSI seven layer model*) [5].

OSI Reference Model merupakan model ideal dari koneksi logis yang harus terjadi agar komunikasi data dalam jaringan dapat berlangsung. Beberapa protokol yang digunakan dalam dunia nyata, semacam *TCP/IP, DECnet dan IBM Systems Network Architecture (SNA)* memetakan tumpukan protokol (*protocol stack*) mereka ke *OSI Reference Model*. *OSI Reference Model* pun digunakan sebagai titik awal untuk mempelajari bagaimana beberapa protokol jaringan di dalam sebuah kumpulan protokol dapat berfungsi dan berinteraksi [5].

2.1.3. TCP/IP (Transmission Control Protocol/Internet Protocol)

TCP/IP adalah serangkaian protokol yang memungkinkan komunikasi antara beberapa komputer. Dulu TCP/IP ini tidak begitu penting bagi komputer-komputer untuk saling berkomunikasi karena bukan protokol yang umum. Tapi saat komputer saling terkoneksi dalam jaringan, kebutuhan itu muncul saat komputer-komputer tersebut sepakat untuk menggunakan protokol-protokol tertentu [6]

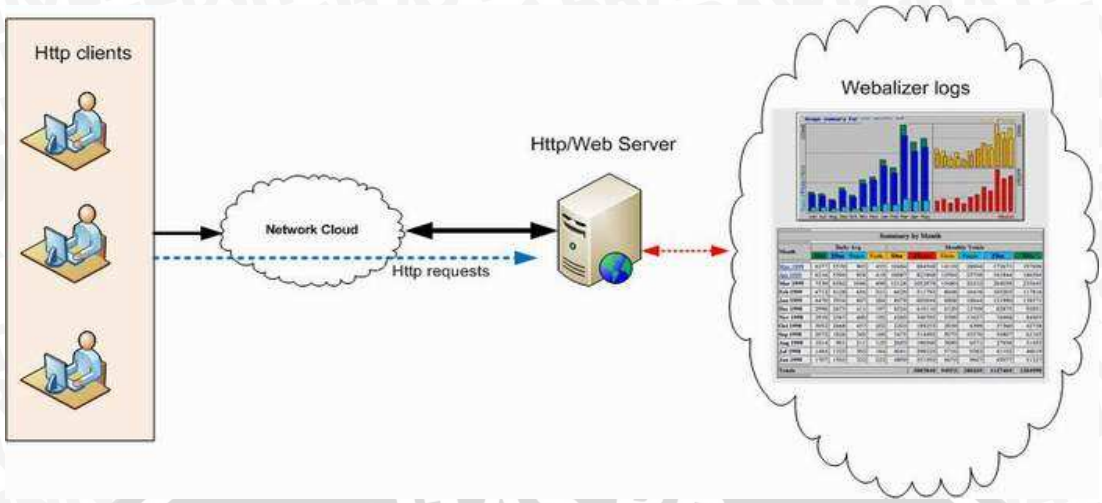
Seperti pada perangkat lunak, TCP/IP dibentuk dalam beberapa lapisan (layer). Dengan dibentuk dalam layer, akan mempermudah untuk pengembangan dan pengimplementasian. Antar layer dapat berkomunikasi ke atas maupun ke bawah dengan suatu penghubung interface. Tiap-tiap layer memiliki fungsi dan kegunaan yang berbeda dan saling mendukung layer diatasnya. Pada protokol TCP/IP dibagi menjadi 4 layer, tampak pada gambar berikut : [7]



Gambar 2.3 Protokol TCP/IP
Sumber : Dhoto (2007)

2.2. Website

Secara terminologi website adalah kumpulan dari halaman-halaman situs, yang biasanya terangkum dalam sebuah domain atau subdomain, yang tempatnya berada di dalam World Wide Web (WWW) di Internet. WWW terdiri dari seluruh situs web yang tersedia kepada publik. Halaman-halaman sebuah situs web (web page) diakses dari sebuah URL yang menjadi "akar" (root), yang disebut homepage (halaman induk; sering diterjemahkan menjadi "beranda", "halaman muka"), URL ini mengatur web page untuk menjadi sebuah hirarki, meskipun hyperlink-hyperlink yang ada di halaman tersebut mengatur para pembaca dan memberitahu mereka susunan keseluruhan dan bagaimana arus informasi ini berjalan [8].



Gambar 2.4 *WebSserver*

Sumber: http://www.otakudang.org (2010)

2.2.1. Hypertext Transfer Protocol (*HTTP*)

Hypertext Transfer Protocol (*HTTP*) merupakan protokol yang digunakan untuk jenis layanan *World Wide Web* (WWW) pada jaringan TCP/IP. Pengembangan *HTTP* dikoordinasi oleh konsorsium WWW dan IETF (*Internet Engineering Task Force*) dan dipublikasikan melalui kumpulan RFC (*Request For Comments*). RCF 2616 mendefinisikan *HTTP*/1.1 yang merupakan versi *HTTP* yang saat ini umum digunakan.

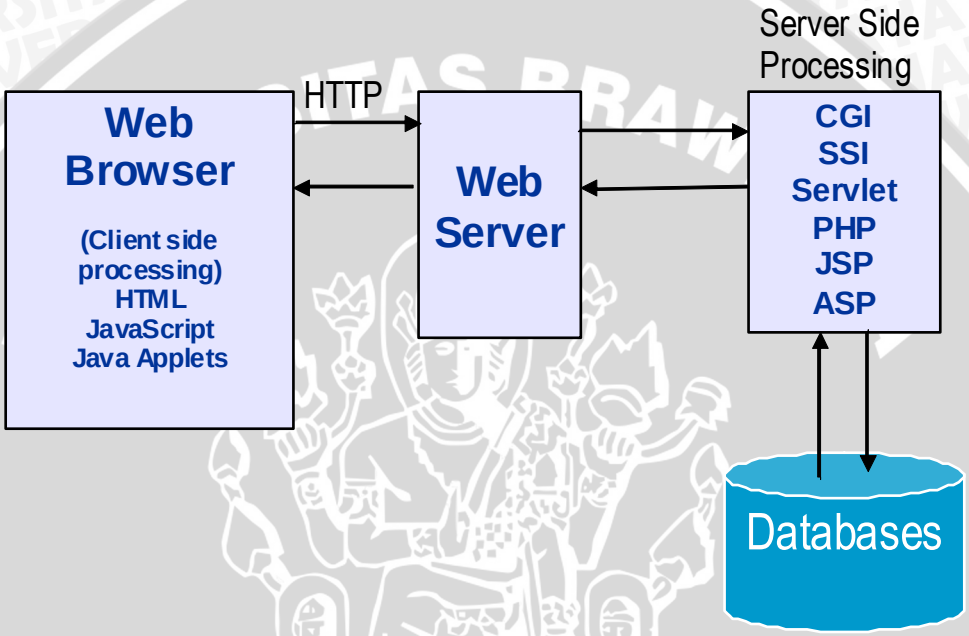
Sebuah *HTTP client* menginisiasi *request* dengan membuat koneksi TCP (*Transmission Control Protocol*) menuju *server* (umumnya adalah port 80). Sedangkan *HTTP server* menunggu adanya pesan *request* pada *port* yang telah ditentukan. Setelah menerima *request* dari *client*, *server* kemudian mengirimkan *status line* antara lain "*HTTP*/1.1 200 OK". Setelah itu dilanjutkan dengan mengirimkan *file* yang diinginkan *client* beserta pesan kesalahan atau informasi lainnya. *HTTP* diidentifikasi menggunakan *Uniform Resource Identifier* (URI) dengan format penulisan tertentu.

Protokol *HTTP* bersifat *request*-respon, yaitu dalam protokol ini *client* menyampaikan pesan *request* ke *web server*, dan *web server* kemudian memberikan respon yang sesuai dengan *request* tersebut. *Request* dan respon dalam protokol *HTTP* disebut sebagai *request chain* dan respon *chain*. Hubungan

HTTP yang paling sederhana adalah terdiri atas hubungan langsung antara *client agent* dengan *server* asal.

2.2.2. Website Script

Script dalam web mempunyai tujuan agar web yang dibuat menjadi lebih menarik dan atraktif. Penggunaan script dalam web akan mempunyai kelebihan dan kekurangan tersendiri.



Gambar 2.5 Pemrosesan Script Web

Pemrosesan script web ada yang dip roses pada server side dan ada yang pada *client* side. Eksekusi script ini mempunyai beberapa kelebihan dan kekurangan. Untuk eksekusi di *client* side, script yang ada akan dieksekusi langsung oleh browser sehingga waktu untuk memproses script ini lebih cepat dibandingkan dengan server side. Untuk kelemahan dari *client* side adalah script yang ada dapat langsung dilihat oleh *client* dan dapat dikopi oleh *client* sehingga jika dari segi keamanan kurang baik. Berbeda dengan server side, script yang ada akan dieksekusi pada server sehingga membutuhkan waktu yang relative lebih lama disbanding dengan *client* side tetapi jika dari segi keamanan server side lebih ama karena script ada pada server sehingga *client* tidak dapat mengetahui script yang ada dan tidak dapat menyalinnya.

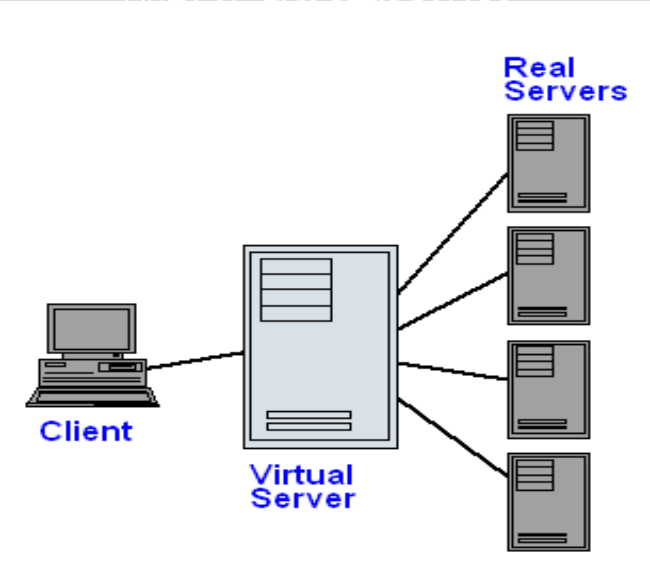
2.3. Load Balancing

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 makin tinggi. Layanan Load Balancing dimungkinkan pengaksesan sumber daya dalam jaringan didistribusikan ke beberapa host lainnya agar tidak terpusat sehingga unjuk kerja jaringan komputer secara keseluruhan bisa stabil.

Ketika sebuah sebuah *server* sedang diakses oleh para pengguna, maka sebenarnya *server* tersebut sebenarnya sedang terbebani karena harus melakukan proses permintaan kepada para penggunanya. Jika penggunanya banyak maka prosesnyapun banyak.

Session-session komunikasi dibuka oleh *server* tersebut untuk memungkinkan para pengguna menerima servis dari *server* tersebut. Jika satu *server* saja terbebani, tentu *server* tersebut tidak bisa banyak melayani para penggunanya karena kemampuan melakukan proses ada batasnya.

Solusi yang paling ideal adalah dengan membagi-bagi beban yang datang ke beberapa *server*. Jadi yang melayani pengguna tidak hanya terpusat pada satu perangkat saja. Teknik ini disebut Teknik *Load balancing* [9].



Gambar 2.6 *Load balancing*

Sumber: http://encyclopedia2.thefreedictionary.com/virtual+server (2010)

2.3.1. Parameter *Load balancing*

2.3.1.1 Paket *Loss*

Paket loss adalah kegagalan salah satu atau lebih paket yang ditransmisikan untuk tiba di tempat tujuan. Hal ini merupakan efek nyata dari semua jenis komunikasi dijital. Paket loss terjadi peak load dan congestion (kemacetan transmisi paket akibat padatnya traffic yang harus dilayani) dalam batas waktu tertentu.

Paket yang hilang ini harus ditransmisi ulang, yang akan membutuhkan waktu tambahan. Prosentase *packet loss* maksimum yang diperbolehkan oleh ISO untuk aplikasi mul*time*dia adalah 5 %. Prosentase *packet loss* ditentukan dengan Persamaan 1 [10].

$$packet\ loss(\%) = \frac{N_{packet\ loss}}{N_{paket} + N_{packet\ loss}} x100\ \% \qquad \textbf{(2.1)}$$

dengan :

$N_{packet\ loss}$ = jumlah paket yang hilang

N_{paket} = jumlah paket yang diterima dengan benar

2.3.1.2 *Throughput*

Throughput adalah bandwidth aktual yang terukur pada suatu ukuran waktu tertentu dalam suatu hari menggunakan rute internet yang spesifik ketika sedang mendownload suatu file [11].

Throughput definisikan sebagai rata-rata paket data yang diterima benar yang dapat ditransmisikan pada satu waktu yang sama. *Throughput* pada jaringan komputer diukur dengan melihat jumlah paket data per detik .

Persamaan *throughput* [12] menurut rumus, dapat ditulis:

$$\lambda = \frac{1}{t_v} \qquad \textbf{(2.2)}$$

Dengan;

λ = *throughput* (packet/s)

t_v= waktu rata-rata untuk mentransmisikan 1 paket yang benar (s)

dimana; $$t_v = \frac{T_{total}}{N_{Paket}} \quad \textbf{(2.3)}$$

T_{total} = Total waktu transmisi

N_{paket} = Jumlah paket yang diterima

2.3.1.3 Waktu Respon

Waktu respon adalah waktu antara *client* memberikan input dengan sistem memberikan output atau umpan balik ke *client*.

$$T = Ta - Td \quad \textbf{(2.4)}$$

Dimana

T = waktu respon

Ta = waktu akses

Td = waktu delay

2.3.1.4 Jumlah *Client* Maksimal

Jumlah *client* maksimal adalah jumlah *client* yang dapat dilayani oleh sebuah server. Jumlah *client* ini dipengaruhi oleh jumlah memori yang digunakan untuk melayani seorang *client*.

$$C = \frac{M_{total} - M_{os} - M_{mysqld} - M_{httpd} - M_{alosql}}{M_{child}} \quad \textbf{(2.5)}$$

Dimana

C = jumlah *client* maksimal

M_{total} = jumlah total memori

M_{os} = memori yang digunakan untuk os

M_{mysqld} = memori yang digunakan untuk menjalankan mysqld

M_{httpd} = memori yang digunakan untuk menjalankan httpd

M_{alosql} = memori yang dialokasikan untuk mysql

M_{child} = memori rata-rata yang digunakan oleh setiap child httpd

2.4. Algoritma Penjadwalan

Dalam *Load balancing* terdapat beberapa algoritma penjadwalan diantaranya :

a. *Least-Connection*

Least-connection mengalokasikan koneksi ke *real-Server* dengan jumlah koneksi paling sedikit **[13]**.

b. *Weighted Least-Connection (wlc)*

Weighted Least-Connection (wlc) hampir sama dengan agoritma *least-connection*, namun ini algoritma yang mempertimbangkan bobot. Bobot diberikan kepada masinng-masing *real-Server*. Untuk koneksi baru akan dipilih *Server* yang paling sedikit kapasitasnya. Kapasitas dihitung dengan membagi bobot suatu *real-Server* dibagi dengan bobot seluruh *real-Server* yang terhubung dengan *virtualServer* **[13]**

c. *Round-Robin (rr)*

Round-Robin (rr) menempatkan semua *real-Server* pada antrian yang melingkar dan mengalokasikan koneksi bergantian untuk setiap *turn* **[13]**.

d. *Weighted Round-Robin (wrr)*

Penjadwalan ini memperlakukan *real-Server* dengan kapasitas proses yang berbeda. Masing-masing *realServer*dapat diberi bobot bilangan integer yang menunjukkan kapasitas proses, dimana bobot awal adalah 1 **[13]**.

e. *Locality-Based Least-Connection (lblc)*

Mendistribusikan permintaan lebih ke *Server* dengan koneksi yang aktif lebih sedikit dibandingkan dengan IP tujuan mereka.Algoritma ini akanmeneruskan semua *request* kepada *real-Server* yang memilikikoneksi kurang aktif tersebut sampai kapasitasnya terpenuhi **[13]**.

f. *Locality-Based Least-Connection Scheduling with Replication*

Mendistribusikan permintaan lebih ke *Server* dengan koneksi yang aktif lebih sedikit dibandingkan dengan IP tujuan. Algoritma

ini juga dirancang untuk digunakan dalam sebuah *clusterServer* proxy-cache. Ini berbeda denganpenjadwalan *Locality-Based Least-Connection,*dengan pemetaan alamat IP target untuk subset dari *Nodereal-Server*. Permintaan ini kemudian diteruskan ke *Server* dalam subset dengan jumlah koneksi paling sedikit. Jika semua *Node* untuk IP tujuan di atas kapasitas, dilakukan replikasi *Server* baru untuk alamat IP tujuan dengan menambahkan *realServer* dengan koneksi yang sedikit dari keseluruhan *real-Server pool* untuk subset dari *real-Server* sebagai IP tujuan *request*. *Node* yang bermuatan lebihdikeluarkan dari subset *real-Server* untuk mencegah over-replikasi **[14]**.

g. Destination Hash Scheduling

Menggunakan hash statis dari alamat IP tujuan untuk mengalokasikan koneksi **[13]**.

h. Source Hash Scheduling

Mendistribusikan permintaan ke kumpulan*real-Server* dengan melihat sumber IP dalam tabel hash statis.

i. *Shortest Expected Delay*

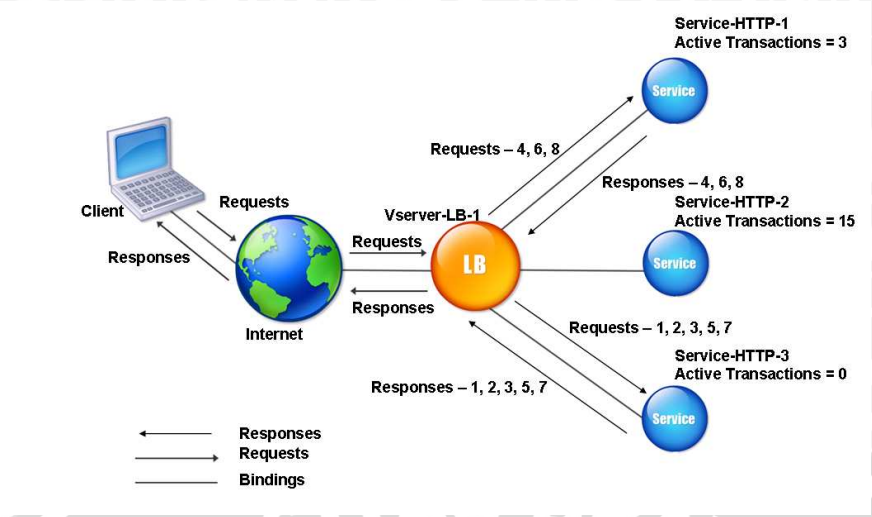
Mengalokasikan koneksi ke *Server* yang akan melayani permintaan dengan delay terpendek.

j. Never Queue

Mengalokasikan koneksi ke *idle real-Server* jika ada, yang lain menggunakan algoritma *Shortest Expected Delay* [13]

2.4.1. Algoritma *Least connection*

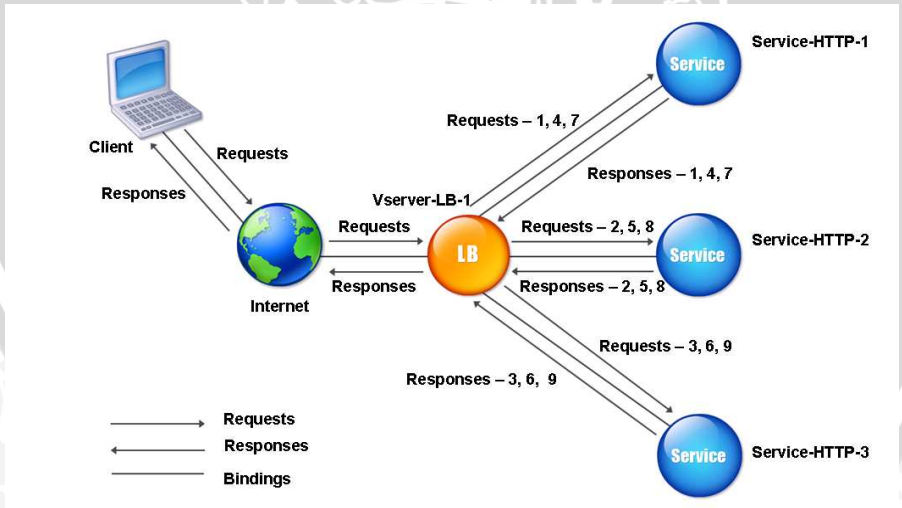
Algoritma *Least Connection* adalah algoritma yang langsung mengarahkan *request* dari *client* ke server yang memiliki jumlah koneksi yang paling sedikit. *Least Connection* adalah salah satu algoritma yang dinamis, karena algoritma ini memerlukan pengecekan secara berkala kepada masing-masing *real server* yang ada. Algoritma *least connection* baik jika digunakan untuk melayani jumlah *request* yang banyak.



Gambar 2.7 Algoritma Least Connection
Sumber: http://support.citrix.com(2010)

2.4.2. Algoritma *Round robin*

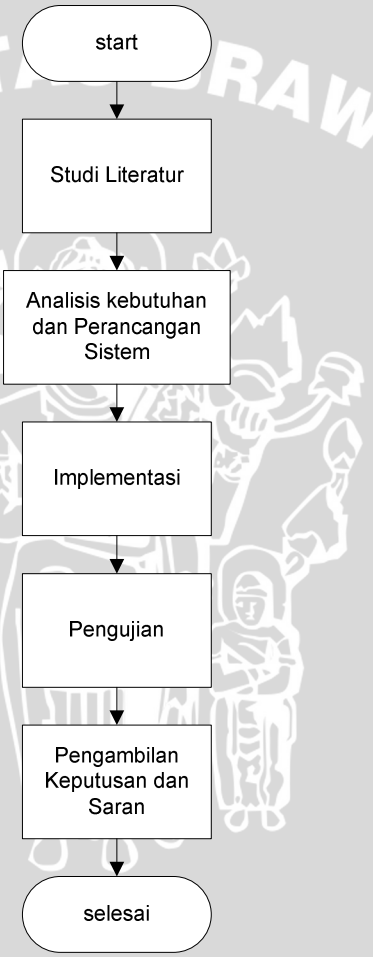
Algoritma *Round Robin* adalah algoritma penjadwalan dimana setiap *request* dari *client* akan langsung diarahkan ke masing-masing *real server*. Algoritma ini membagi sama rata tanpa memandang dari jumlah koneksi yang terbentuk. Hal ini akan menyebabkan ketidakseimbangan yang dinamis dari masing-masing *real server*.



Gambar 2.8 Algoritma Least Connection
Sumber: http://support.citrix.com(2010)

BAB III
METODOLOGI PENELITIAN

Metodelogi dalam penelitian ini membahas langkah-langkah studi literatur, analisis kebutuhan, implementasi, pengujian hingga penarikan kesimpulan dari sistem *Load balancing* baik menggunakan algoritma *Round robin* ataupun *Least connection*. Dari langkah-langkah yang ada dapat digambarkan dalam bentuk diagram seperti pada gambar 3.1



Gambar 3.1 Diagram Alir Metodologi Penelitian

3.1. Studi Literatur

Studi literatur yang dilakukan bertujuan untuk mengkaji hal-hal yang berhubungan dengan teori-teori yang mendukung dalam perancangan dan perealisasisan aplikasi. Studi literaturyang dilakukan mengenai:

1. Jaringan computer
2. *web server*
3. *Load balancing*
4. Algoritma *Round robin*
5. Algoritma *Least connection*
6. *Linux Virtual Server*
7. Pengujian *web server*

3.2. Analisis Kebutuhan dan Perancangan Sistem

Topologi jaringan komputer merupakan representasi dari interkoneksi antara *director*, *real server,* dan *client*. Perancangan topologi dilakukan agar *director* dapat menerapkan fungsinya sebagai *load balancer* dengan baik. Berdasarkan rancangan topologi tersebut, dapat ditentukan kebutuhan perangkat keras dan pengalokasian alamat IP untuk masing-masing komputer yang akan terhubung dengan jaringan komputer *server*.

3.3. Implementasi

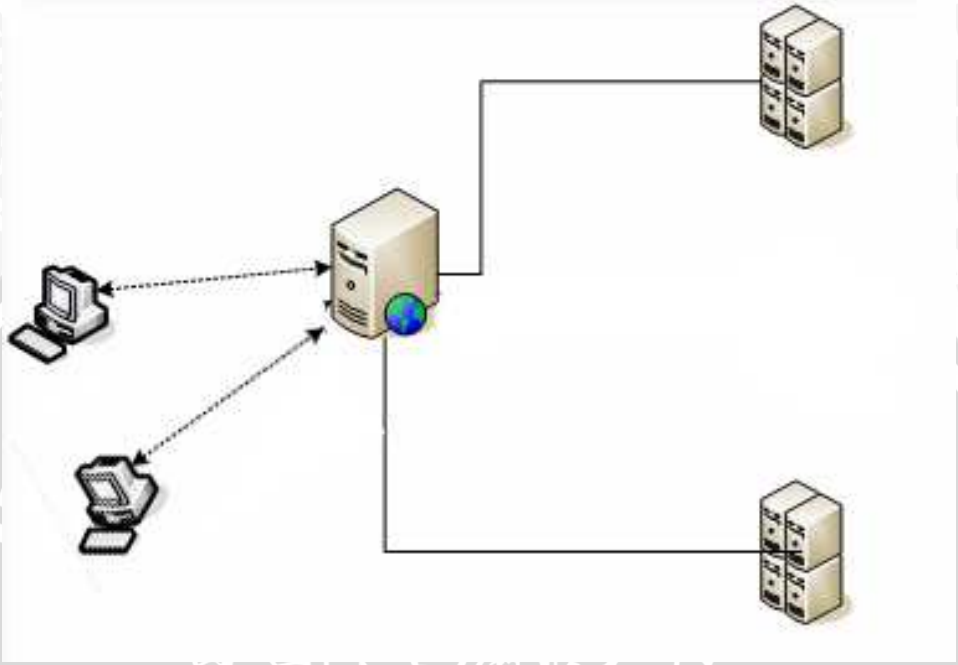
Untuk mengimplementasikan metode *Load balancing* pada *web server* Apache, komputer *server* dihubungkan sesuai hasil rancangan topologi jaringan komputer.

Setelah jaringan komputer *server* terhubung secara fisik, berikutnya adalah melakukan konfigurasi jaringan pada masing-masing komputer *server*. Langkah selanjutnya adalah melakukan instalasi dan konfigurasi *web server* Apache pada *director* dan masing-masing *real server*.

3.4. Pengujian dan Analisis Hasil

Pengujian sistem dilakukan untuk mengetahui proses serta keberhasilan skenario dan konfigurasi sistem *Load balancing* serta ke berhasilan sistem untuk mendistribusikan beban pada *server* yang diujikan. Parameter keberhasilan adalah keberhasilan penyampaian paket, paket loss, waktu tunda (delay) dan throughput serta hubungan antara paket loss dan waktu tunda.

Pengujian akan dilakukan dengan menggunakan simulasi sehingga akan didapatkan hasil yang mendekati dengan kondisi nyata *web server*. Skenario pengujian yang dilakukan antara *Round robin* dengan *Least Connetcion* harus sama.



Gambar 3.2 *Load balancing*

Skenario pengujian dilakukan pada *single web server* dan *Load balancing web server*. Pengujian yang dilakukan meliputi:

1. Penentuan jumlah *client* yang dapat dilayani oleh masing-masing web server.
2. Menentukan *respond time* dari *server* dalam menangani sebuah *request.*
3. Penanganan *request* ketika menggunakan *Load balancing,* baik menggunakan algoritma *Round robin atau Least connection.*
4. Pengujian parameter-parameter yang diamati sebagai bahan untuk dianalisis.

Analisis yang dilakukan adalah dari data yang telah didapatkan dapat dibandingkan performansi dari

1. Error *request* antara single server, algoritma *Round robin* dan *Least connection*.
2. Troughput antara single server, algoritma *Round robin* dan *Least connection*.
3. Respon *time* antara single server, algoritma *Round robin* dan *Least connection*.

3.5. Pengambilan Keputusan dan Saran

Setelah mendapatkan hasil dan analisis dari tahap pengujian, maka langkah berikutnya adalah penarikan kesimpulan dari hasil pengujian dan memberi saran untuk pengembangan lebih lanjut.

BAB IV
PERANCANGAN

Proses analisis kebutuhan dilakukan dengan pendekatan terstruktur. Bagian perancangan meliputi analisis kebutuhan, kebutuhan sistem, kebutuhan perangkat lunak, dan kebutuhan perangkat keras**.**

4.1. Analisis Kebutuhan

Analisis kebutuhan adalah aktifitas rekayasa perangkat lunak yang menjembatani antara kebutuhan ditingkat sistem dan perancangan perangkat lunak. Dengan analisis kebutuhan aka dihasilkan spesifikasi dan kerakteristik sistem yang diharapkan [15]

Proses analisis kebutuhan *Load balancing* dengan menggunakan algoritma *Round robin* serta *Least connection* meliputi spesifikasi sistem dan permodelan analisis sistem.

4.1.1. Kebutuhan Sistem

Dalam kebutuhan sistem dapat dibagi menjadi 2 bagian, yaitu kebutuhan sistem dari sisi perangkat lunak dan kebutuhan sistem dari sisi perangkat keras.

4.1.2. Analisis Kebutuhan Perangkat Lunak

Kebutuhan untuk membangun sistem *Load balancing* pada *web server* membutuhkan perangkat lunak sebagai berikut:

1. Sistem Operasi

   Sistem operasi yang digunakan dalam pembuatan sistem *Load balancing* pada *web server* ini adalah Centos 5.7 dengan kernel 2.6.18. Sistem operasi ini digunakan pada *director* dan *real server*. Sistem operaasi linux yang mempunyai kernel di bawah 2.4.28 belum mendukung virtual *server* sehingga tidak dapat digunakan dalam membuat sistem *Load balancing*.

2. *Load balancing* (Piranha)

   Sistem *Load balancing* yang dibuat menggunakan perangkat *Linux Virtual Server* (*LVS*) dari RedHat yaitu Piranha. Piranha adalah

perangkat *monitoring cluster* dalam *LVS* dan juga dapat digunakan untuk konfigurasi *director* dan *real server* dalam sistem *LVS*.

Dalam Piranha terdapat ipvsadm (IP Virtual Server Administration) yang mengatur kerja dari *director*. Dalam ipvsadm dapat diterapkam fungsi-fungsi yang mengatur kerja *director* termasuk juga algoritma penjadwalan. Ketika *director* mendapat sebuah *request*, *director* akan meneruskan *request* ke *real server* sesuai dengan algoritma yang digunakan.

3. *Web server*

   Untuk *web server* digunakan Apache *web server*. Apache *web server* memiliki program pendukung yang cukup banyak yang dapat memberikan pelayanan bagi penggunanya. Program-program yang didukung apache *web server* seperti *PHP, java script, flash*

4. Database Server

   Halaman web yang dinamis membutuhkan database. *MySQL*, diucapkan ”my ess que el” adalah open source, enterprise-level, multi threaded, relational database management system. *MySQL* menggunakan bahasa standar *Structure Query Language (SQL).* Dalam implementasi menggunakan mysql-*server* dan mysql-*client* versi 5.0.95. Apache membutuhkan modul tambahan berupa php-mysql.

5. Stress Tool

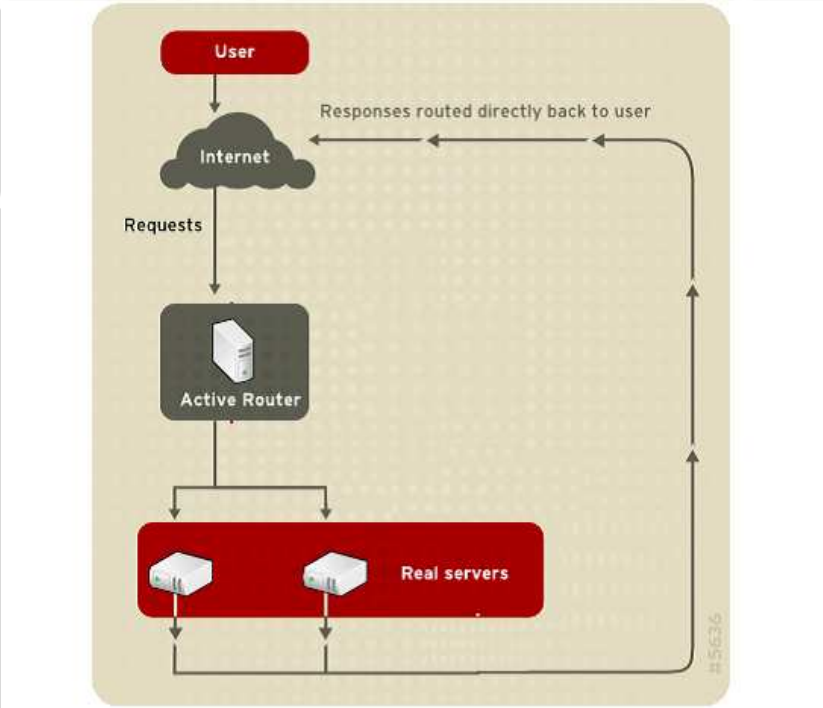
   Stress tool berfungsi sebagai simulator *client* sehingga dapat dibuat *client* dengan jumlah yang ingin diujikan. *Stress tool* yang digunkan berupa Peassler Web Stress Tool 7.

4.1.2.1 Sistem Linux Virtual Server (LVS)

Linux Virtual Server atau disingkat *LVS* merupakan suatu teknologi clustering yang dapat digunakan untuk membangun suatu *server* dengan menggunakan kumpulan dari beberapa *real server*. *Load balancing* menggunakan *LVS* dapat menerapkan algoritma *Round robin* ataupun *Least connection*. Sistem *Load balancing* yang dibuat memiliki karakteristik dan fungsi sebagai berikut:

1. Fungsi *Load balancing* pada web*server* adalah menciptakan sebuah *web server* dalam sebuah cluster komputer. Web dapat diakses melalui IP *director*.
2. Sistem *Load balancing* digunkana pada sistem jaringan yang menggunakan IPv4.
3. Reliability sistem *Load balancing* pada *web server* akan bekerja lebih baik jika ditinjau dari parameter-parameter seperti troughput, *request* loss, CPU utilization, dan waktu repson.

Pada gambar 4.1 merupakan beberapa node dalam sistem *LVS*. Setiap node memiliki fungsi yang berbeda.



Gambar 4.1 Topologi *LVS* Direct Routing

1. *Client*

Dalam sebuah sistem *web server client* adalah *client* yang megakses website dalam *web server* tersebut. *Client* dapat berasal dari *network* dan lokasi yang berbeda. Hal ini menyebabkan *client* sebuah *service web server* sangat banyak. Untuk itu dibutuhkan sistem pembagian beban dalam *cluster* komputer yang dapat melayani *request* yang sangat banyak dari *client*

2. *Director*

Dalam *Linux Virtual Server director* disebut *virtual Server*. *Client* yang melakukan *request website* akan mengakses IP dari *director. Director* dalam *LVS* akan melanjutkan *request* dari *client* ke *real server* yang memiliki data *website* yang diminta oleh *client. Director* tidak memiliki data *website* dan berfungsi sama seperti *router* yaitu meneruskan paket data ke alamat yang di tuju. Oleh karena itu *director* disebut dengan *virtual server.*

3. *Real server*

Real server adalah *Server* yang mempunyai data website yang sebenarnya. *Real server* terdiri lebih dari satu mesin. Beban *request* dari *client* akan dibagikan kepada seluruh *real server* yang aktif. Jumlah *real server* yang lebih dari satu menyebabkan sistem lebih *high availability* karena apabila ada satu *real server* yang mati *request* dari *client* tetap dapat dilayani. Pembagian beban kepada *real server* ditentukan dengan penggunaan algoritma antrian.

LVS memiliki komponen-komponen yang mempunyai fungsi sendiri-sendiri seperti tampak pada tabel 4.1

Tabel 4-1 Komponen *LVS* dan Fungsinya

Komponen	Fungsi
Pulse	Merupakan komponen yang bertugas untuk mengontrol proses jalannya daemon yang sesuai dengan kebutuhan proses. Pulse bekerja pada *LVS director* dan dapat dijalankan dengan perintah /etc/rc.d/init.d/pulse start, atau dapat dijalankan setiap kali boot. Pulse merupakan heartbeat yang mengirimkan sinyal pada aktif *director* maupun untuk file-over sistem
LVS	*LVS* daemon ini bertugas untuk membaca file konfigurasi dan memanggil program IPVSADM dan menampilkan tabel routing dari IPVS
Nanny	Merupakan komponen yang bertugas untuk memonitor daemon yang bekerja pada aktif *director*. Dengan nanny,

	director dapat mengetahui bahwa *real server* mana yang aktif, dan dapat memonitor beban kerja dan jumlah *client* yang ditangani oleh tiap *real server LVS*.
/etc/lvs.cf	Merupakan file konfigurasi dari *LVS* cluster. Baik secara langsung maupun tidak, semua daemon mengambil informasi konfigurasi dari file ini
Piranha Web Interface	Merupakan tool yang berbasiskan web untuk memonitoring, mengatur, dan mengkonfigurasi *LVS* cluster. Secara normal, tool ini adalah untuk memonitor file lvs.cf melakukan restart daemon yang bekerja dan momonitor *LVS* cluster
Send_arp	Merupakan program yang bertugas mengirimkan arp broadcast ketika alamat virtual berubah dari satu node ke node yang lain.

4.1.3. Analisis Kebutuhan Perangkat Keras

Hardware yang digunakan untuk membuat sistem *Load balancing* pada *web server* antara lain, *Central Processing Unit* atau CPU (*director* dan *real server)*, ethernet *card*, kabel UTP dan switch.

1. CPU untuk *Director*

   Untuk spesifikasi 1 buah *director* yang digunakan adalah sama dengan rincian sebagai berikut :

   - Prosesor : Intel(R) Pentium(R) III CPU 1266MHz
   - Memori : 644 MB
   - Kapasitas Hardisk : 18 GB

   Untuk spesifikasi 2 buah *real server* yang digunakan dengan rincian sebagai berikut :

2. CPU untuk *real server* 1

   - Prosesor : Intel(R) Core(TM) i3 CPU 540 @3.07GHz
   - Memori : 2 GB
   - Kapasitas Hardisk : 500 GB

3. CPU untuk *real server* 2
   - Prosesor : AMD Athlon(tm) 64 X2 Dual Core Processor 5000+
   - Memori : 2 GB
   - Kapasitas Hardisk : 8 GB

4. Switch

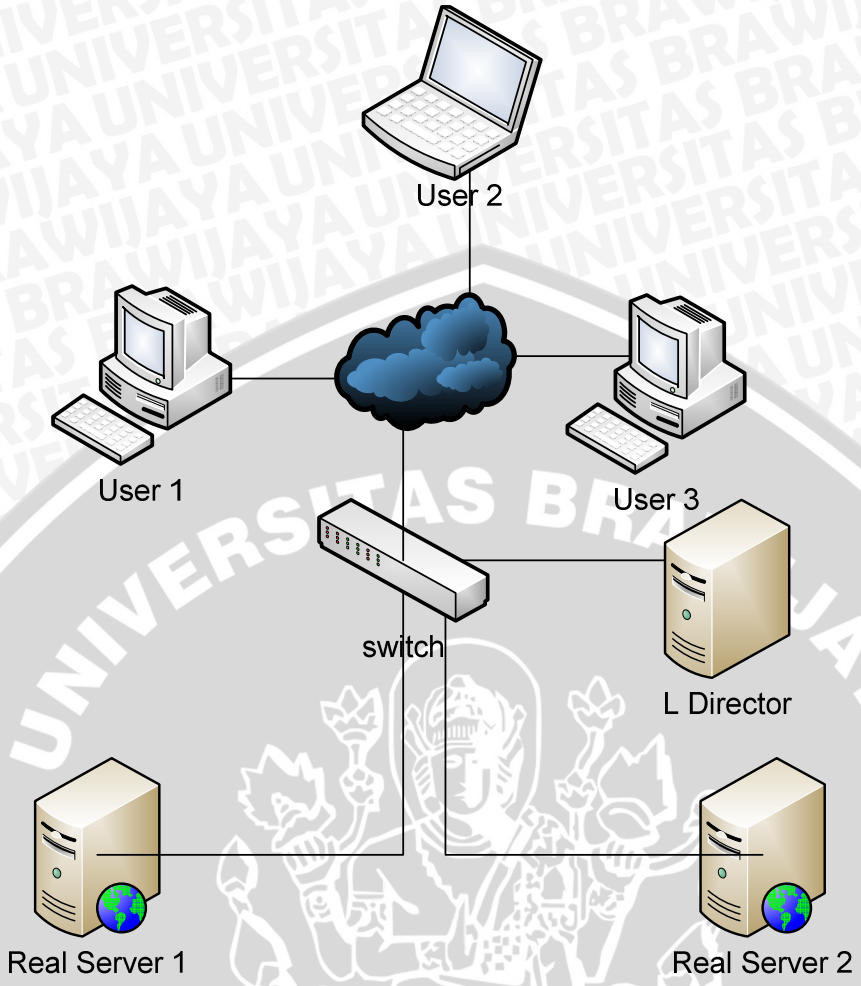
   Untuk spesifikasi switch yang digunakan adalah *manageable* switch HP ProCurve2510G-24

4.2. Perancangan

4.2.1. Perancangan Topologi Jaringan

Dalam perancangan topologi jaringan komputer, digunakan topologi bintang (*star*) dengan menggunakan sebuah *switch* sebagai penghubung antara *Director* dengan *real server*. Pemilihan topologi ini berdasarkan kelebihannya dibandingkan dengan topologi lainnya, yaitu kemudahan dalam perawatan dan *troubleshooting*. Topologi jaringan komputer yang digunakan dirancang sedemikian rupa sehingga seluruh *HTTP request* yang dikirimkan oleh *client* diarahkan secara terpusat menuju *director*.

HTTP request dari *client* diterima oleh *director*. *Request* tersebut kemudian di-*forward* menuju salah satu *real server*. *Real server* yang menerima *request* hasil *forwarding* dari *director server* akan memproses *HTTP request* tersebut, kemudian mengirimkan *HTTP* respon menuju *director*. *Director* menerima *HTTP* respon tersebut, untuk selanjutnya dikirimkan menuju *client* sebagai respon dari *request* yang dikirimkan sebelumnya oleh *client*. Perancangan topologi yang digunakan ditunjukkan dalam Gambar 4.2



Gambar 4.2 Topologi *Load balancing*

Pada sistem *Load balancing* dengan *LVS*, *request* dari *client* akan diteruskan kepada *real server* lalu dari *real server* akan membalas *request* langsung ke *client* tanpa melalui *director*. Terlihat pada tabel 4.1 tipe *forwarding* pada teknik *Load balancing*.

Tabel 4-2 Tipe Forwarding Pada *Load balancing*

	Layer-2 Forwarding	Layer-3 Forwarding
Two-Ways	-	NAT
One-Ways	Direct Routing	IP Tunneling

Sumber: Linux VirtualServer (LVS) for Red Hat Enterprise Linux 5.1. (2007)

Teradapat arsitektur *Two-Ways* dan *One-Ways forwarding* pada *Load balancing*. Arsitektur *Two-Ways* adalah respon dari *server* ke *client* dapat melalui *director*. Pada arsitektur *One-Ways* respon dari *server* mencari jalur alternatif langsung menuju *client* tanpa melewati *director*.[SIE-09]

Layer-2 *forwarding* adalah *forwarding* pada layer 2 OSI (Open Systems Interconnection) yaitu pada data link layer. Layer-3 *forwarding* adalah *forwarding* pada OSI yaitu pada network layer. Sistem *Load balancing* yang dibuat menggunakan layer-2 *forwarding* karena jaringan menggunakan direct routing.

4.2.2. Perancangan Alamat IP

Dalam perancangan topologi dapat dilihat bahwa topologi yang digunakan semua *real server* dan *director* ada pada satu jaringan yang sama, sehingga pemberian IP juga menyesuaikan menjadi satu jaringan.

Untuk alokasi IP dapat dilihat pada tabel 4.2

Tabel 4-3 Alokasi Alamat IP

Nama	Alokasi IP	Virtual IP
Director	175.45.185.42/25	-
Real server 1	175.45.185.43/25	175.45.185.42/25
Real server 2	175.45.185.44/25	175.45.185.42/25

IP yang digunakan adalah alamat IP public, agar *server* dapat diakses dari manapun. Alamat IP virtual yang digunakan sebagai alamat IP yang diakses oleh *client*, lalu alamat tersebut akan didistribusikan ke alamat *real server* yang sebenarnya.

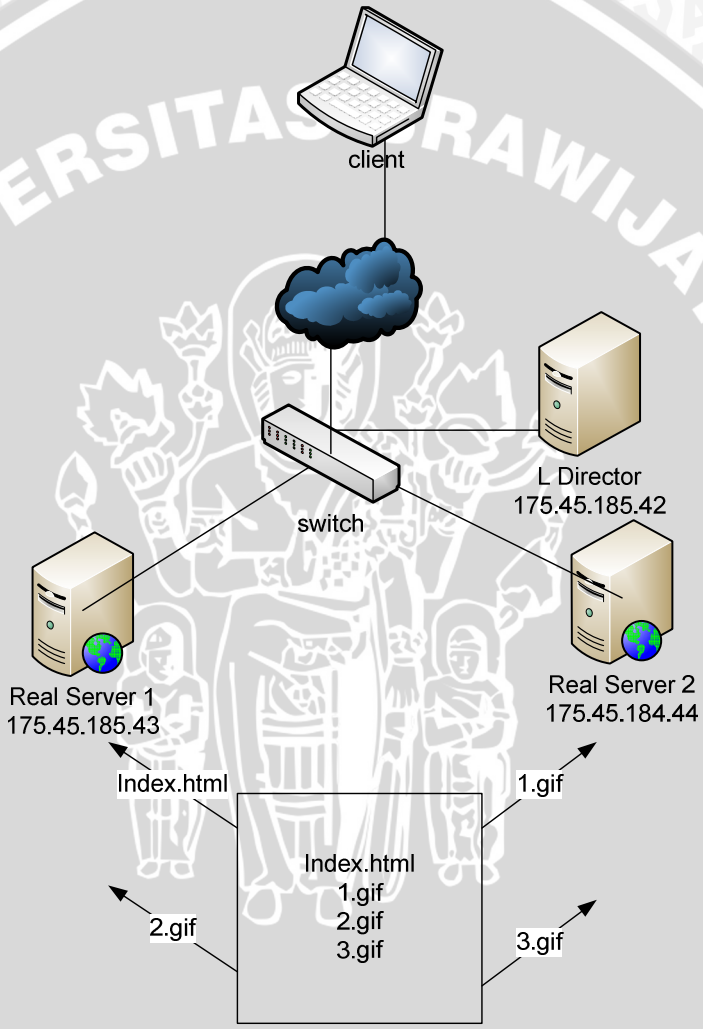
4.3. Cara Kerja Sistem

Algoritma yang digunakan adalah *Round robin* dan *Least connection* karena sistem yang dibuat menggunakan protokol TCP yang bersifat *connection oriented*. Algoritam *Round robin* membagi beban berdasarkan dengan jumlah *request* secara bergantian sedangkan algoritma scheduling *Least connection* membagi

beban berdasarkan dengan jumlah koneksi ke masing-masing *real server* dengan kapasitas yang sama [14].

4.3.1. Algoritma *Round robin*

Algoritma *Round robin* adalah algoritma yang membagi dengan rata setiap *request* yang masuk secara bergantian. Tampak pada gambar 4.3 cara kerja algoritma *Round robin*.



Gambar 4.3 Cara Kerja Algoritma *Round robin*

Algoritma ini membagi *request* secara bergantian antara *request* satu dengan yang lain untuk dilayani oleh masing-masing *real server* tanpa melihat *server* yang diberikan *request* sedang melayani *request* yang lain atau tidak.

4.3.2. Algoritma *Least connection*

Least connection mengarahkan *request* dengan jumlah koneksi paling sedikit. Ini adalah salah satu algoritma penjadwalan dinamik, karena dalam pendistribusian *request* akan menghitung jumlah koneksi yang terbentuk pada *real server* terlebih dahulu.

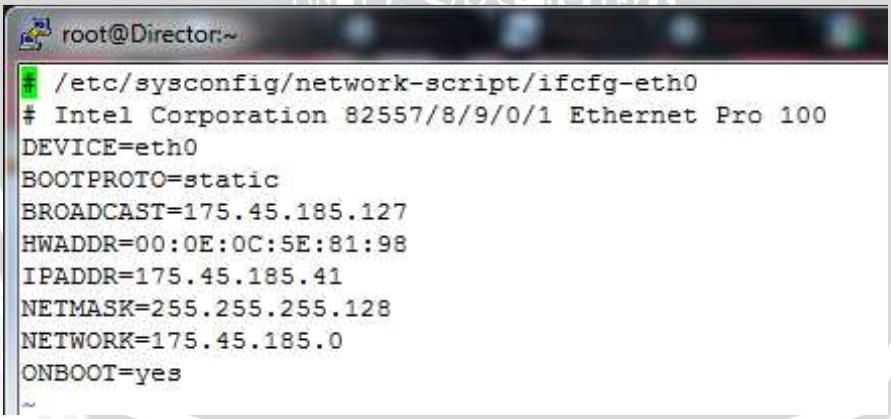
BAB V
IMPLEMENTASI

Berdasarkan perancangan yang telah dibuat pada bab IV, dapat dilakukan implementasi. Implementasi yang dilakukan dapat dibagi menjadi beberapa bagian yaitu:

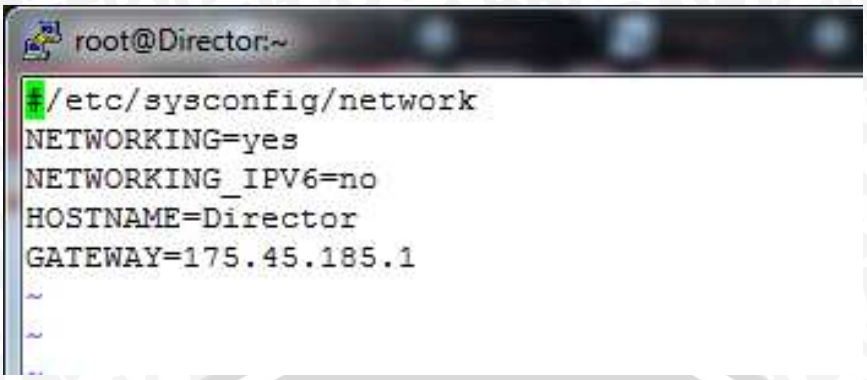
1. Konfigurasi jaringan pada *director*
2. Konfigurasi jaringan pada masing-masing *real server*
3. Installasi LVS pada *director*
4. Installasi web server pada *director*

5.1. Konfigurasi Jaringan Pada *Director*

Konfigurasi jaringan pada *director* meliputi konfigurasi alamat ip, gateway serta DNS. Untuk dapat mengkonfigurasi harus sebagai root dari sistem. File yang perlu diubah dari konfigurasi adalah file-file yang berada di */etc/sysconfig/network-script/ifcfg-eth0* untuk konfigurasi alamat ip, */etc/sysconfig/network* untuk konfigurasi default gateway, dan */etc/resolve.conf* untuk konfigurasi DNS seperti tampak pada gambar 5.1, gambar 5.2, gambar 5.3 dan gambar 5.4.

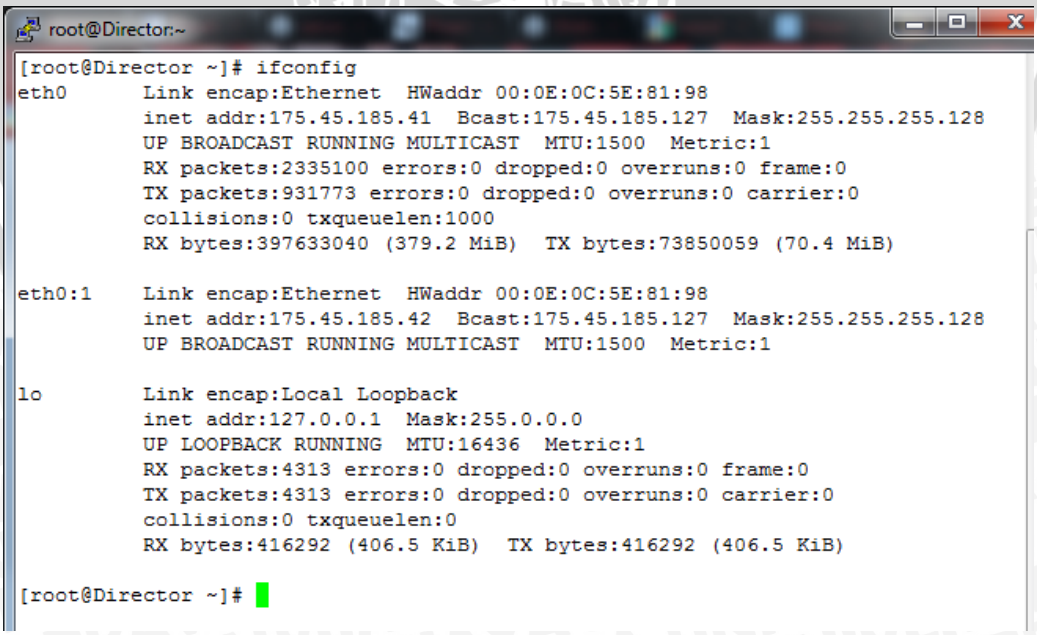


Gambar 5.1 Konfigurasi Alamat IP



Gambar 5.2 Konfigurasi Default Gateway

Gambar 5.3 Konfigurasi Alamat DNS



Gambar 5.4 Hasil Konfigurasi Jaringan Pada *Director*

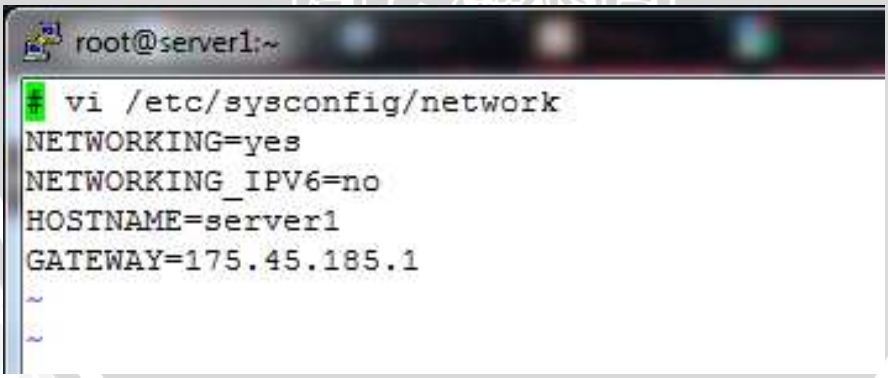
5.2. Konfigurasi Jaringan Pada *Real server*

Konfigurasi jaringan pada *director* meliputi konfigurasi alamat ip, gateway serta DNS. Untuk dapat mengkonfigurasi harus sebagai root dari sistem. File yang perlu diubah dari konfigurasi adalah file-file yang berada di /etc/sysconfig/network-script/ifcfg-eth0 untuk konfigurasi alamat ip, /etc/sysconfig/network untuk konfigurasi default gateway, /etc/resolve.conf untuk konfigurasi DNS.

Dengan menggunakan perancangan topologi dan perancangan ip seperti bab perancangan maka file yang harus dikonfigurasi dapat diubah seperti pada berikut

1. Konfigurasi pada server 1

Gambar 5.5 Konfigurasi Alamat IP



Gambar 5.6 Konfigurasi Default Gateway

```
root@server1:~
#vi /etc/resolv.conf
nameserver 175.45.184.164
nameserver 175.45.184.165
~
~
~
~
```

Gambar 5.7 Konfigurasi Alamat DNS

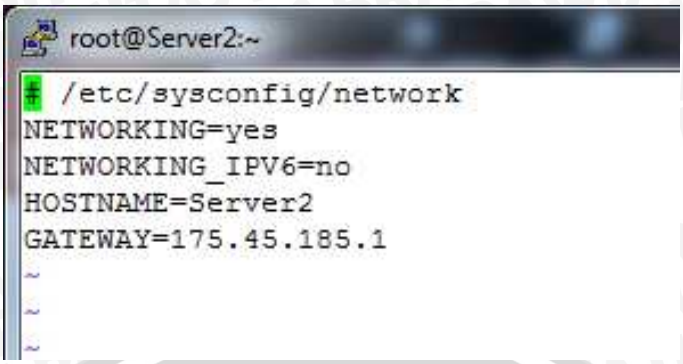
```
root@server1:~
[root@server1 ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 6C:62:6D:51:26:E5
          inet addr:175.45.185.43  Bcast:175.45.185.127  Mask:255.255.255.128
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2261854 errors:0 dropped:0 overruns:0 frame:0
          TX packets:924471 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:201413632 (192.0 MiB)  TX bytes:427828001 (408.0 MiB)
          Interrupt:50 Base address:0xe000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:9210 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9210 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:804512 (785.6 KiB)  TX bytes:804512 (785.6 KiB)

lo:0      Link encap:Local Loopback
          inet addr:175.45.185.42  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:16436  Metric:1

[root@server1 ~]#
```

Gambar 5.8 Hasil Konfigurasi Jaringan Pada *Server 1*

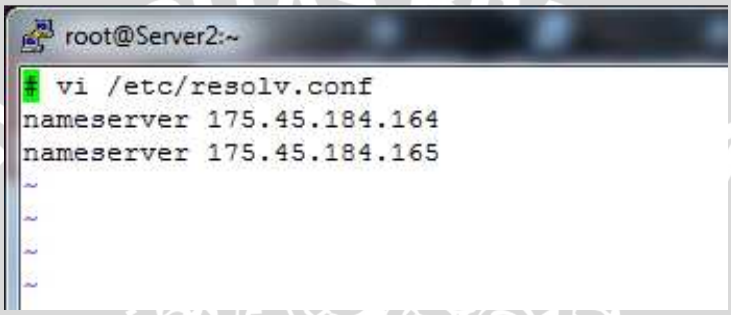
2. Konfigurasi pada server 2

```
root@Server2:~
# nVidia Corporation MCP77 Ethernet
# /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=static
BROADCAST=175.45.185.127
HWADDR=00:18:37:04:C0:09
IPADDR=175.45.185.44
NETMASK=255.255.255.128
NETWORK=175.45.185.0
ONBOOT=yes
~
~
```

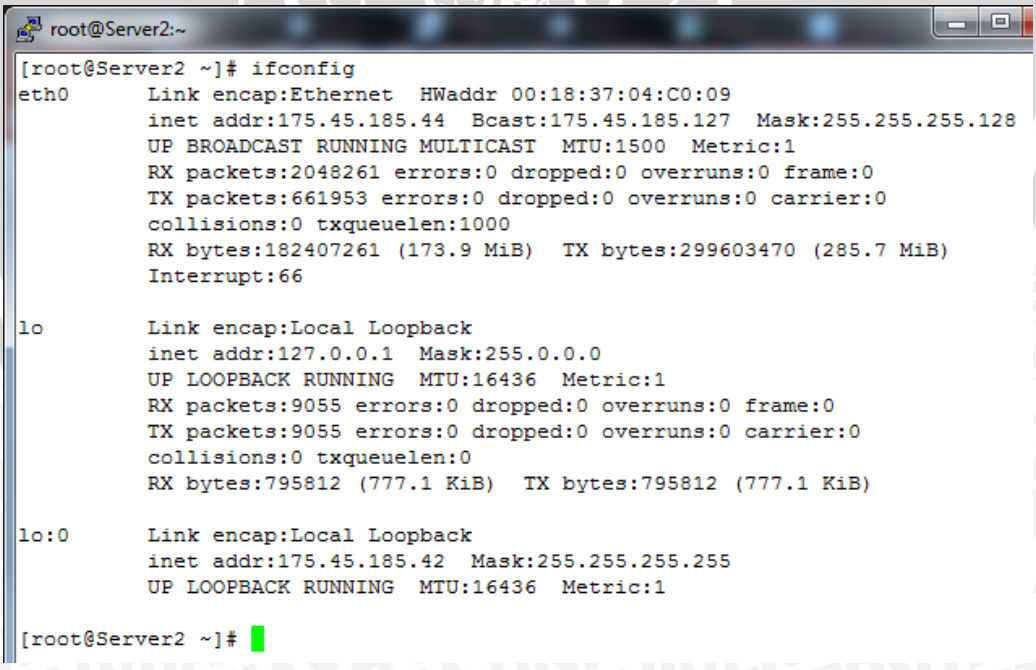
Gambar 5.9 Konfigurasi Alamat IP



Gambar 5.10 Konfigurasi Default Gateway



Gambar 5.11 Konfigurasi Alamat DNS



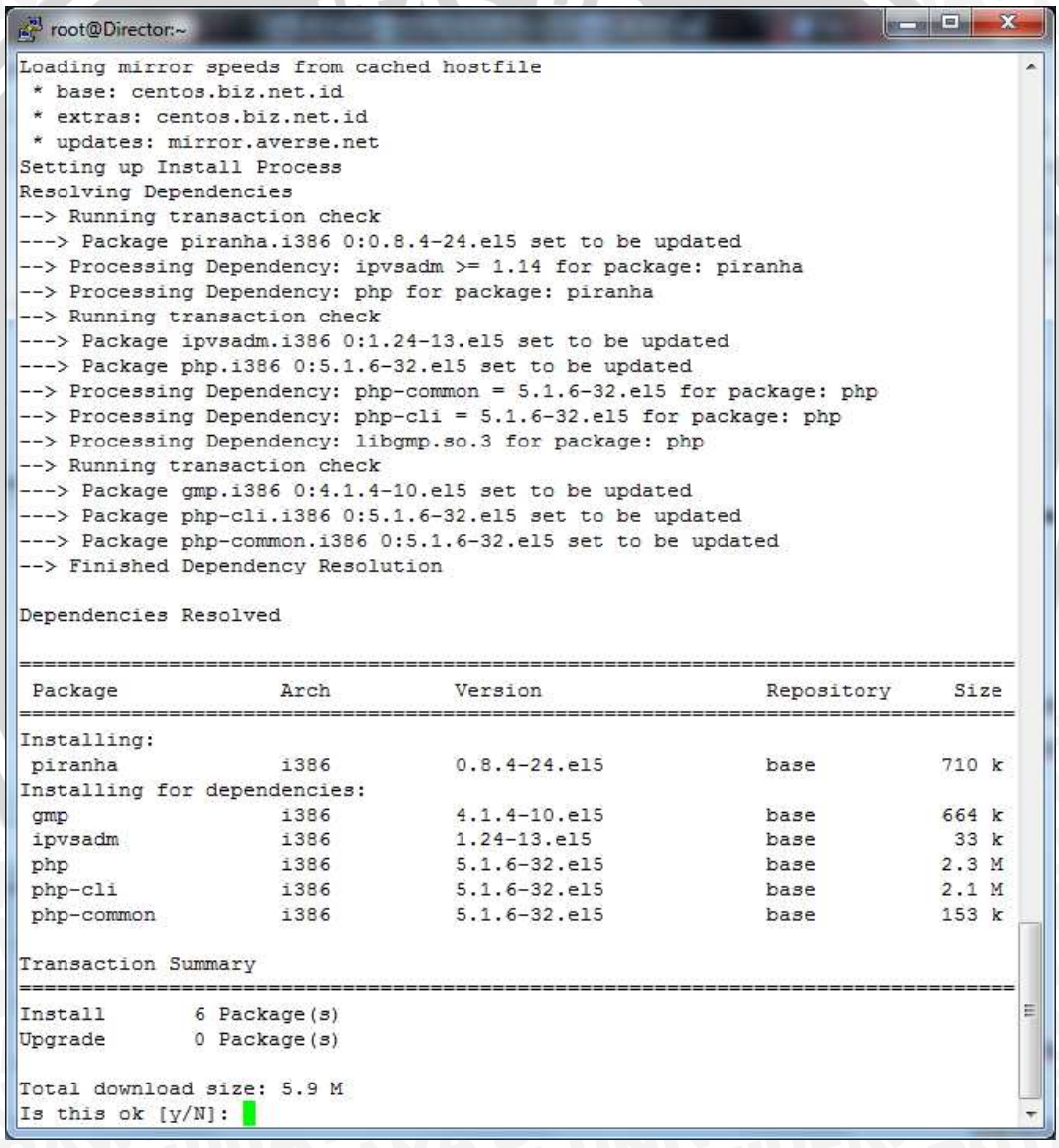
Gambar 5.12 Hasil konfigurasi jaringan pada *Server2*

5.3. Installasi web server pada *Director*

Proses installasi yang dilakukan adalah proses installasi menggunakan paket yang berada pada repo yang ada. Dalam installasi web server ini beberapa paket yang dibutuhkan antara lain

1. Httpd
2. mysql
3. Mysql-server
4. Ipvsadm

Proses installasi paket-paket yang diperlukan tampak pada gambar 5.13

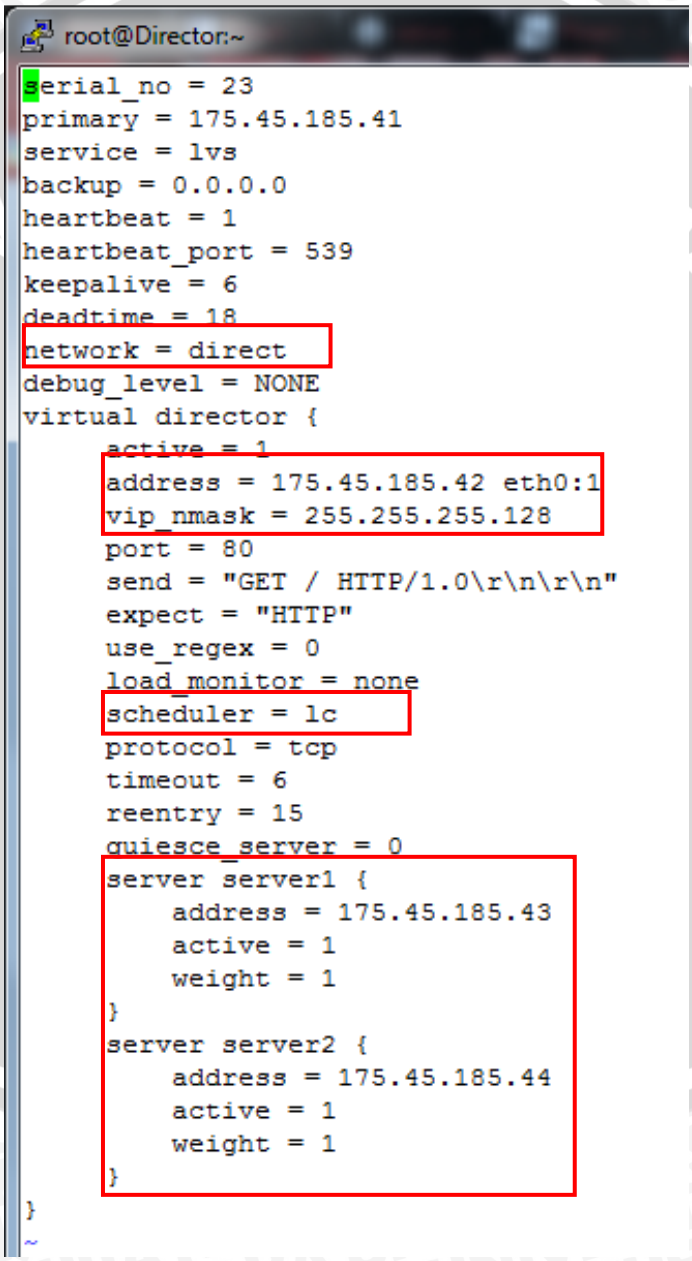


Gambar 5.13 Proses Installasi Paket

5.4. Konfigurasi LVS pada *Director*

LVS pada *director* ini berfungsi sebagai pembagi beban dari *request* yang ada. Beban yang seharusnya diterima oleh sebuah server maka akan diteruskan untuk dibagi ke beberapa server yang ada.

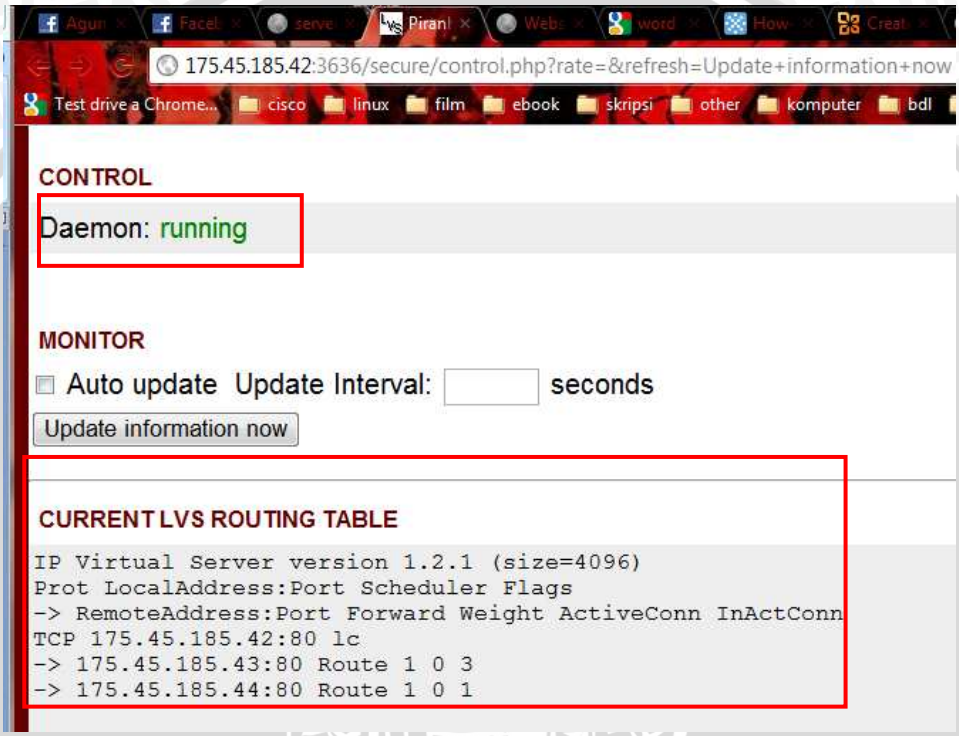
Konfigurasi yang perlu dilakukan agar LVS dapat berjalan ada pada file /etc/sysconfig/ha/lvs.cf



Gambar 5.14 Tampilan Konfigurasi LVS

Pada gambar 5.14 tampak konfigurasi LVS. Pada konfigurasi tampak bahwa metode yang digunakan adalah direct routing dengan virtual IP 175.45.185.42 pada interface eth0:1, algoritma yang digunakan adalah *least connection* dan ada dua buah *real server* yang masing-masing mempunyai alamat IP 175.45.185.43 dan 175.45.185.44 dengan bobot yang sama yaitu 1.

Piranha-gui adalah web interface dari LVS. Fungsi dari Piranha-gui adalah memonitoring serta web interface dari LVS.



Gambar 5.15 Tampilan Piranha-gui

Dari Piranha-gui dapat terlihat bahwa LVS daemon telah berjalan, serta pada table routing di Piranha-gui dapat terlihat table routing dari load balancing serta koneksi yang sedang terjadi.

BAB VI
PENGUJIAN DAN ANALISIS

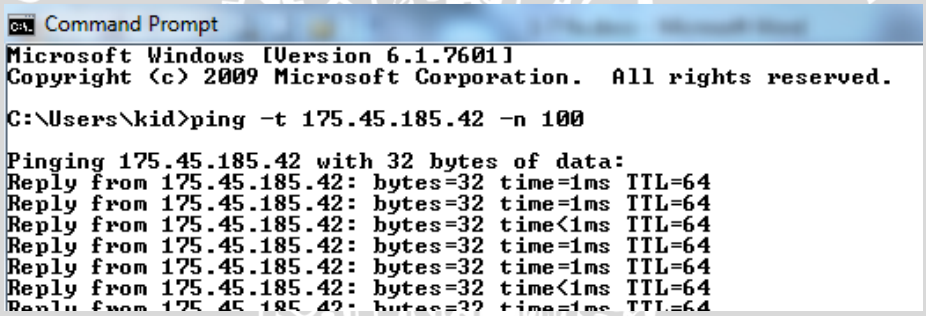
Pengujian sistem adalah untuk memastikan sistem dapat berjalan serta memberikan keluaran sesuai dengan yang diharapkan.

Sistem *Load balancing* diuji dengan memberikan masukan jumlah *client* yang berbeda-beada untuk mendapatkan keluaran sesuai dengan parameter-parameter yang telah ditentukan.

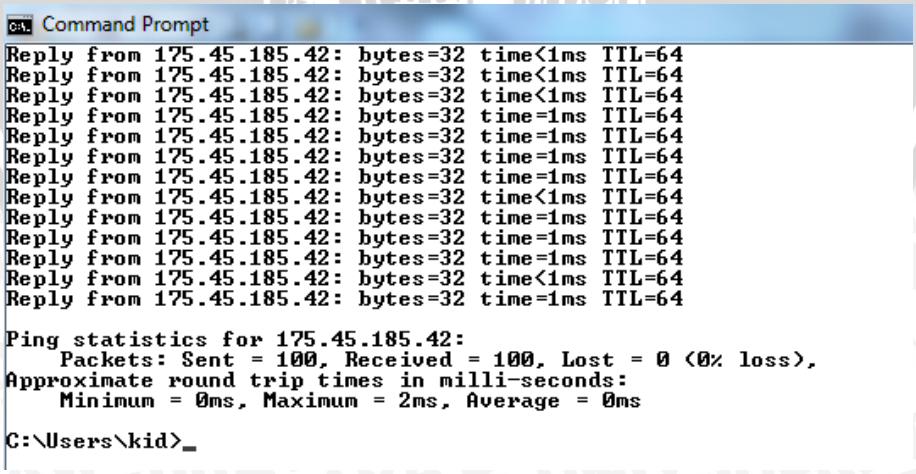
6.1. Pengujian Koneksi *Client* Server

Pengujian ini dilakukan untuk memastikan bahwa koneksi antara *client* server tidak terputus. Pengujian ini dilakukan dengan menggunakan perintah ping pada *client* kepada server. Pengujian ditunjukkan pada gambar 6.1.

Pada *client* dijalankan perintah berikut pada teminal *client* untuk mendapatkan hasil ping sampai dengan 100 kali.



Gambar 6.1 Perintah Ping



Gambar 6.2 Hasil Perintah Ping

6.2. Pengujian Web Server

Pengujian single web server ini adalah sebagai pembanding performa web server tanpa menggunakan *Load balancing*. Pengujian single web server juga dimaksudkan untuk mengetahui jumlah *client* maksimal yang mampu ditangani oleh sebuah web server.

6.2.1. Jumlah *Request* Maksimal

Dalam Apache setiap *request* yang masuk akan didistribusikan ke dalam child proses yang sedang tidak menangani *request*. Child proses ini merupakan duplikasi parent proses, karena child proses mempunyai alamat memori awal yang sama dengan parent proses httpd.

Tabel 6-1 Konsumsi Memori Pada Httpd

nama proses	PID	PPID	Use Memory (KB)
httpd (parent)	2819	0	9000
httpd (child)	2921	2819	5400
httpd (child)	2922	2819	18000
httpd (child)	2923	2819	5400
httpd (child)	2924	2819	18000
httpd (child)	2925	2819	18000
httpd (child)	2926	2819	5400
httpd (child)	2927	2819	5400
httpd (child)	2928	2819	18000
Total penggunaan memori child			93600
Total penggunaan memori setiap child			11700

Jumlah *client* maksimal yang dapat dilayani oleh sebuah server dapat dihitung dari memori yang tersedia pada server dibagi dengan jumlah penggunaan memori setiap child proses dari httpd sesuai dengan persamaan 2.5

$$C = \frac{M_{total} - M_{os} - M_{mysqld} - M_{httpd} - M_{alosql}}{M_{child}}$$

$$= \frac{1806252 - 65532 - 27280 - 21528 - 137516}{11700}$$

$$= 132.85 \text{ } client$$

Dimana

C = jumlah *client* maksimal

M_{total} = jumlah total memori

M_{os} = memori yang digunakan untuk os

M_{mysqld} = memori yang digunakan untuk menjalankan mysqld

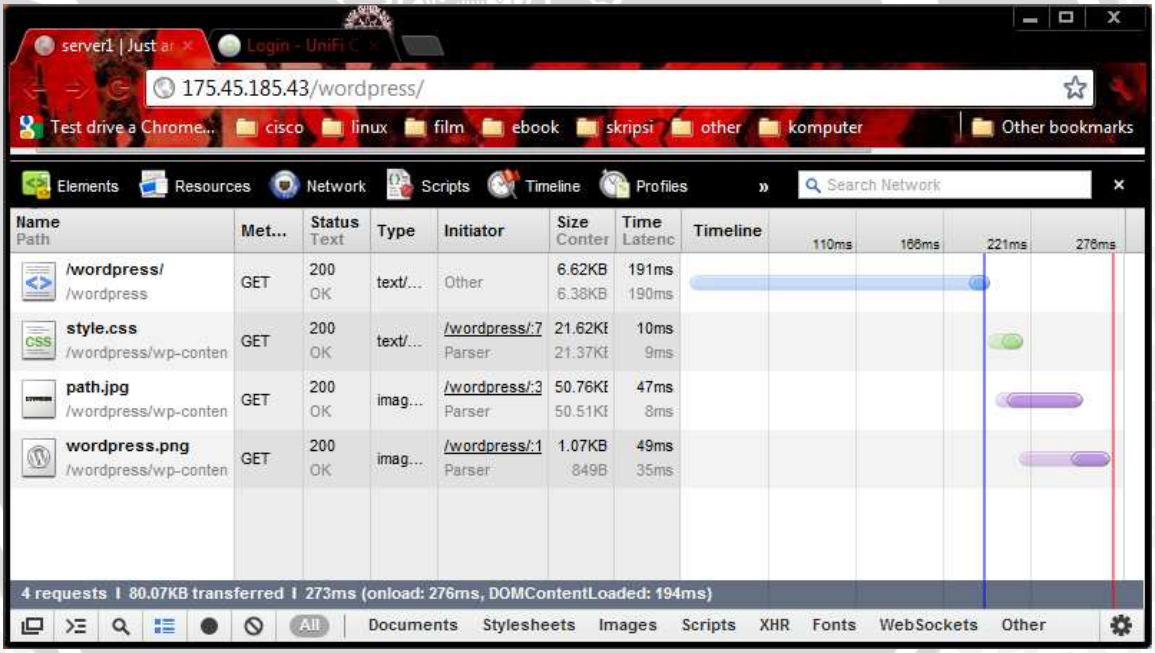
M_{httpd} = memori yang digunakan untuk menjalankan httpd

M_{alosql} = memori yang dialokasikan untuk mysql

M_{child} = memori rata-rata yang digunakan oleh setiap child httpd

6.2.2. Akses Halaman Web

Dalam satu halaman web dapat terdiri beberapa file yang diakses. File-file tersebut ikut terakses ketika halaman index.php terbuka. Pada penelitian ini file-file yang terakses dan waktu yang dibutuhkan untuk mengakses tampak pada gambar 6.3 dan perhitungannya tampak pada table 6.2



Gambar 6.3 Inspect Element Chrome

Tabel 6-2 Waktu Akses File

Nama File	Tipe File	Waktu respon	Waktu Akses (ms)
Index.php	index	190	191
Style.css	Css	9	10
Path.jpg	Image (jpg)	8	47
Wordpress.png	Image (png	35	49
Total Waktu		242	297

Dari waktu respon dan waktu akses ada perbedaan delay. Perhitungan waktu delay didapatkan dengan persamaan 2.5 dan didapatkan

T = Ta-Td

242 = 297-Td

Td = 297-242

Td = 55 ms

Dimana

T = waktu respon

Ta = waktu akses

Td = waktu delay

Jadi waktu delay untuk sekali akses web dengan seorang *client* yang sedang mengakses akan didapatkan 55ms.

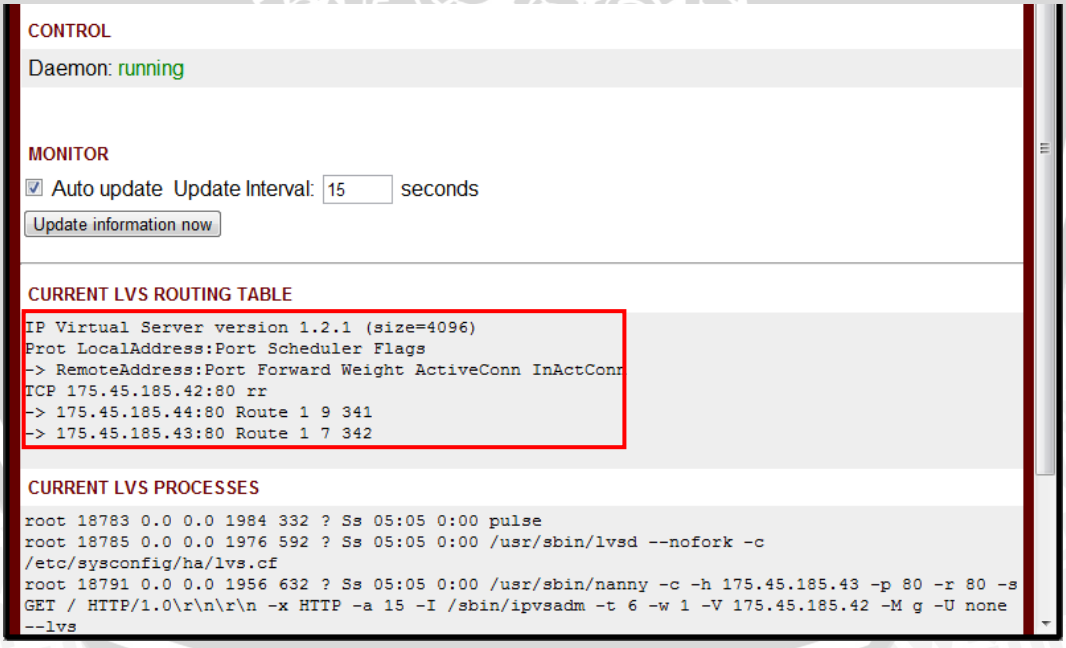
6.3. Pengujian Pendistribusian *Request*

Pengujian distribusi *request* ini dilakukan untuk melihat bagaimana pendistribusian *request* jika menggunakan algoritma *Round robin* dan bagaimana pendistribusian *request* jika menggunakan algoritma *Least connection*.

Skenario pengujian untuk pendistribusian *request* masing-masing algoritma diberi beban 300 *request*. Hasil pengujian untuk pendistribusian untuk algoritma *Least connection* tampak pada gambar 6.4 dan untuk *Round robin* tampak pada gambar 6.5

Gambar 6.4 Pendistribusian *Request* Pada *Least connection*

Pada gambar 5.4 tampak bahwa pada *least connection* untuk koneksi yang aktif adalah 38 dan 40 koneksi sehingga InActConn menjadi 1251 dan 747 koneksi.



Gambar 6.5 Pendistribusian *Request* Pada *Round robin*

Pada gambar 6.5 tampak bahwa dengan menggunakan *round robin* maka koneksi yang dihasilkan sama rata. Pada gambar 6.5 tampak bahwa koneksi yang aktif adalah 9 dan 7 koneksi sedangkan InActConn 341 dan 342 koneksi.

6.3.1. Pengujian Unjuk Kerja Web Server

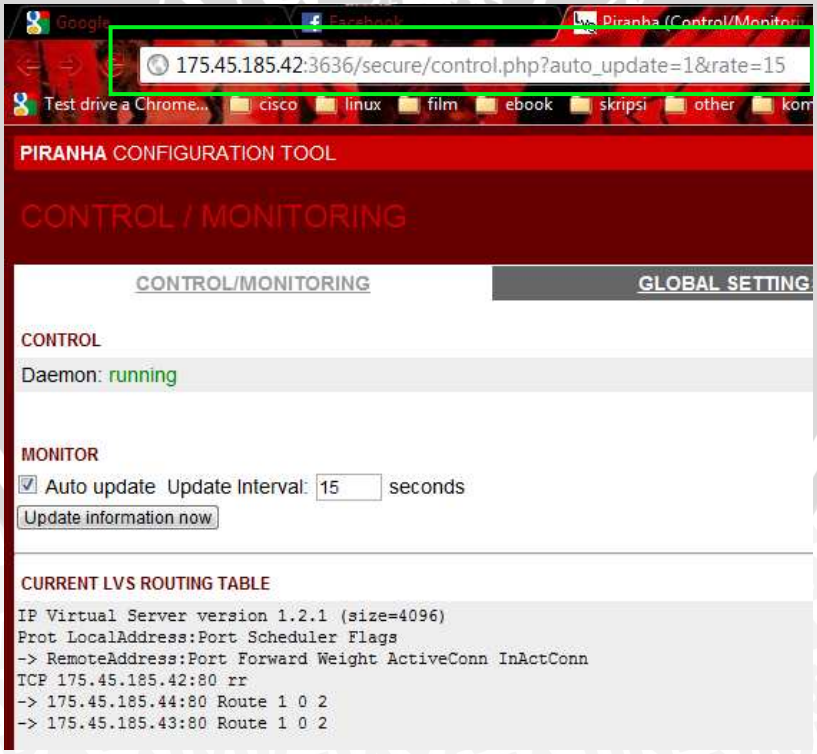
Pengujian unjuk kerja web server dilakukan bagaimana kinerja server selama dilakukan pengujian dengan pemberian beban secara bertahap. Skenario yang dilakukan yaitu:

1. Pengujian unjuk kerja single server untuk jaringan lokal Universitas Brawijaya
2. Pengujian unjuk kerja *Load balancing* server untuk jaringan lokal Univeristas Brawijaya menggunakan Algoritma *Round robin*
3. Pengujian unjuk kerja *Load balancing* server untuk jaringan lokal Univeristas Brawijaya menggunakan Algoritma *Least connection*

6.3.2. Hasil Pengujian Unjuk Kerja Web Server

Pengujian dilakukan dengan *stress tool* dengan menggunakan 50, 100, 150, 200, 250, dan 300 *client*

Pada gambar 6.6 terlihat *director* dengan ip https://175.45.185.42:3636 dan *real server* dengan alamat ip masing-masing 175.45.185.43 dan 175.45.184.44.



Gambar 6.6 *Director* Menggunakan Piranha

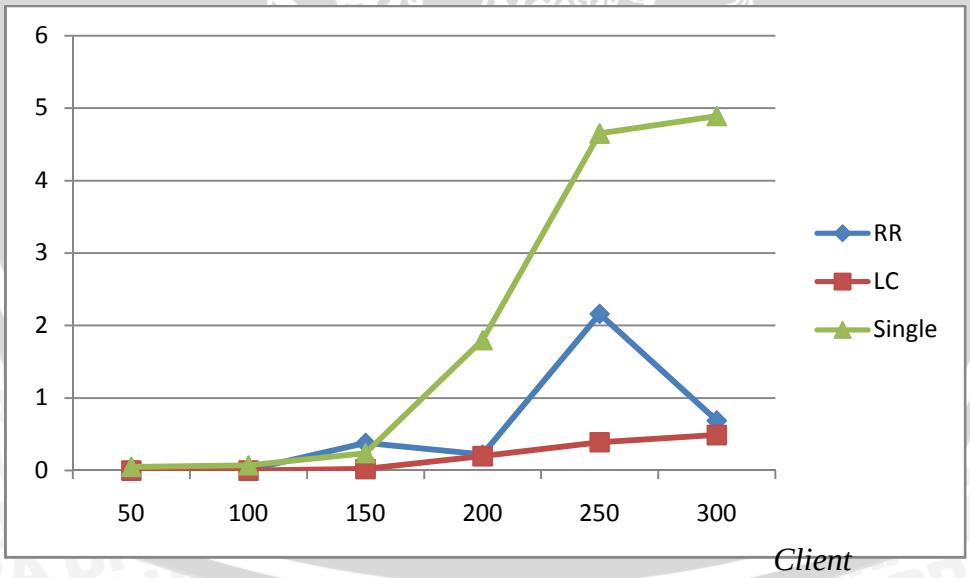
1. Hasil Pengujian unjuk server pada web server di jaringan Universitas Brawijaya

   Hasil dari pengujian dengan mengunakan single server pada *web server* Universitas Brawijaya terlihat pada tabel 6.3 dan grafik 6.7

Tabel 6-3 Total Error Pada Web Server Di Jaringan Universitas Brawijaya

Jumlah *Client*	RR (%)	LC (%)	Single(%)
50	0	0	0.05
100	0.01	0	0.07
150	0.38	0.02	0.24
200	0.22	0.2	1.8
250	2.16	0.39	4.65
300	0.69	0.49	4.89

Terlihat dari tabel bahwa nilai prosentase error *request* paling kecil terdapat pada *Least connection*. Untuk lebih jelas tampak pada gambar 6.7



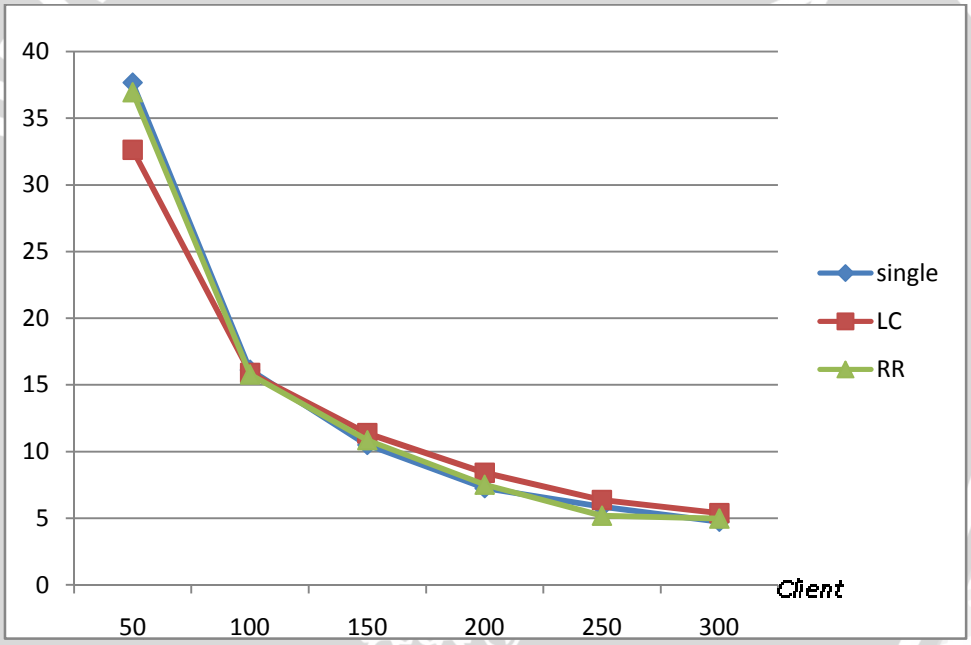
Gambar 6.7 Prosentase Error

Pada gambar 6.7 tampak bahwa prosentase nilai error dari single server paling tinggi jika dibandingkan dengan menggunakan load balancing. Prosentase error masih kecil ketika beban tidak melewati jumlah *client* maksimal untuk single server.

2. Pengujian troughput *Load balancing* server untuk jaringan Univeristas Brawijaya terlihat pada tabel 6-4 dan grafik 6.8

Tabel 6-4 Tabel Troughput

Jumlah *Client*	RR (kb/s)	LC (kb/s)	Single (kb/s)
50	36.9366	32.617	37.6518
100	15.7913	15.8822	16.1137
150	10.85313	11.36647	10.5036
200	7.5222	8.40425	7.25805
250	5.19972	6.35244	5.85144
300	4.986833	5.379633	4.734133



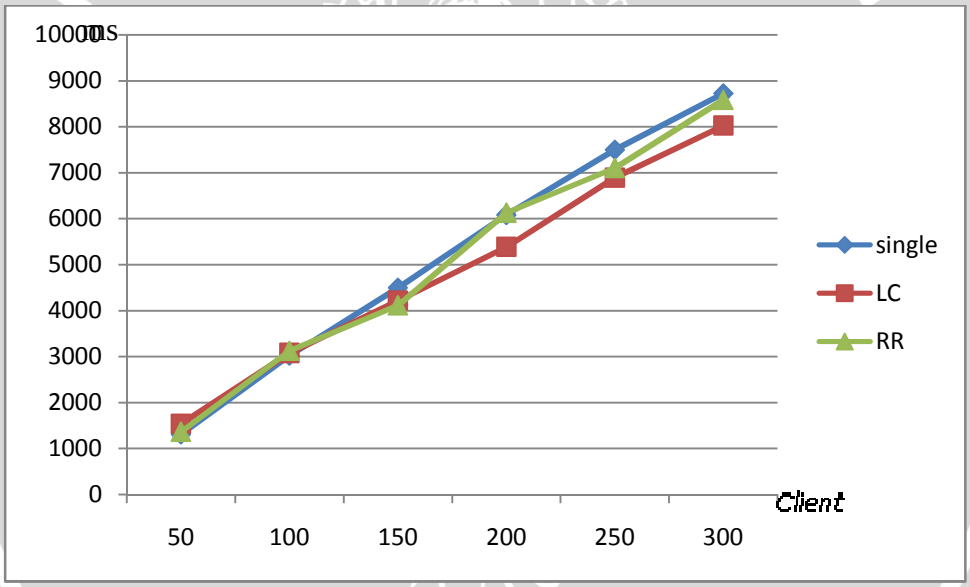
Gambar 6.8 Troughput Untuk Single Server, *Round robin* dan *Least connection*

Pada gambar 6.8 terlihat bahwa troughput dari pengujian semuanya tidak berbeda jauh. Untuk jumlah *client* sedikit nilai troughput paling baik jika menggunakan single server, tetapi jika jumlah *client* melebihi dari jumlah *client* maksimal maka nilai troughput terbaik tampa pada algoritma *Least connection*.

3. Dari pengujian juga didapatkan waktu respon dari masing-masing algoritma yang tampak pada tabel 6-5 dan grafik 6.9

Tabel 6-5 Waktu Respon Untuk Web Server PadaJaringan Universitas Brawijaya

Jumlah *Client*	RR (ms)	LC (ms)	Single (ms)
50	1365	1540	1313
100	3124	3079	3025
150	4114	4218	4499
200	6128	5386	6083
250	7102	6892	7498
300	8582	8023	8723



Gambar 6.9 Rata-rata Respon untuk Load Balacing Server untuk Single Server, Algoritma *Least connection* dan *Round robin*

Pada gambar 6.9 terlihat bahwa waktu repon dari pengujian semuanya tidak berbeda jauh. Untuk jumlah *client* sedikit nilai waktu respon paling baik jika menggunakan single server, tetapi jika jumlah *client* melebihi dari jumlah *client* maksimal maka nilai waktu respon terbaik tampa pada algoritma *Least connection*

Pada pengujian pada jam sibuk tampak pada gambar 6.10, gambar 6.11 dan gambar 6.12 serta tabel 6-6, tabel 6-7 dan tabel 6-8

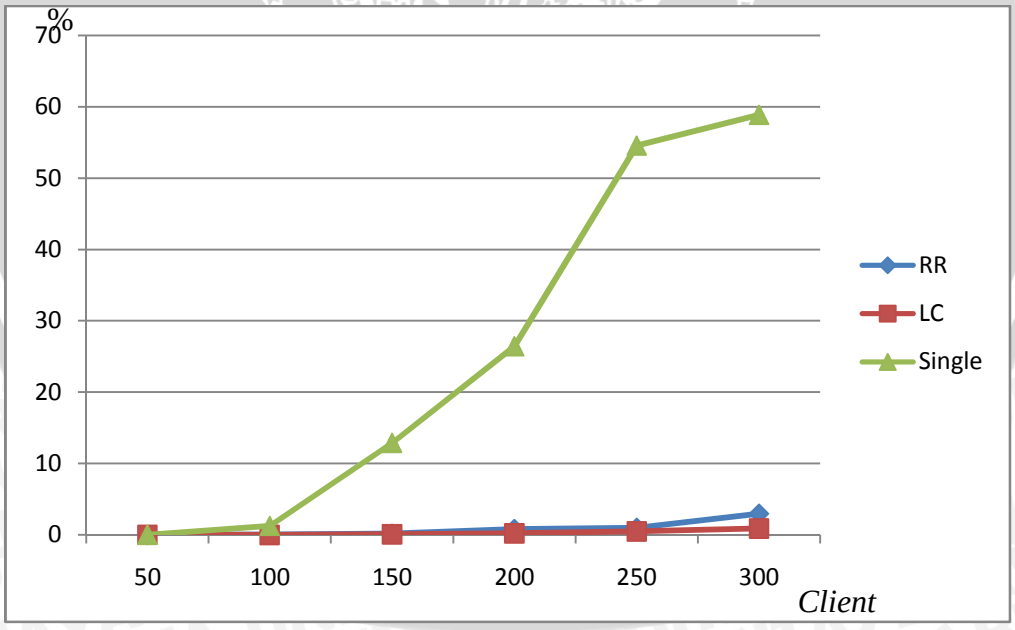
1. Hasil Pengujian unjuk server pada web server di jaringan Universitas Brawijaya

   Hasil dari pengujian dengan mengunakan single server pada *web server* Universitas Brawijaya terlihat pada tabel 6.3 dan grafik 6.7

Tabel 6-6 Total Error Pada Web Server Di Jaringan Universitas Brawijaya

Jumlah *Client*	RR (%)	LC (%)	Single(%)
50	0	0.03	0.01
100	0.05	0	1.25
150	0.17	0.1	12.9
200	0.81	0.23	26.44
250	1.01	0.5	54.56
300	2.99	0.9	58.87

Terlihat dari tabel bahwa nilai prosentase error *request* paling kecil terdapat pada *Least connection*. Untuk lebih jelas tampak pada gambar 6.7



Gambar 6.10 Prosentase Error

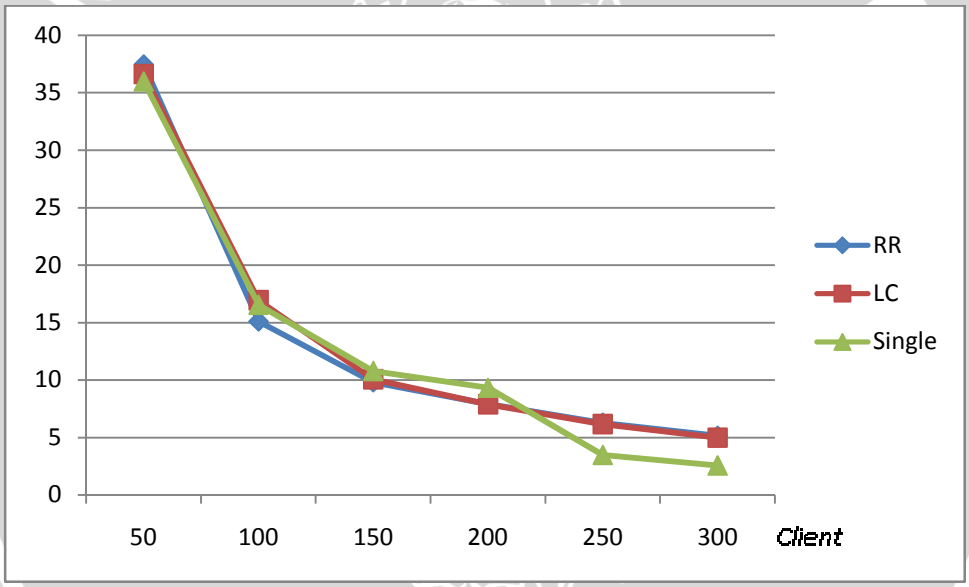
Pada gambar 6.7 tampak bahwa prosentase nilai error dari single server paling tinggi jika dibandingkan dengan menggunakan load balancing. Prosentase error masih kecil ketika beban tidak melewati jumlah *client*

maksimal untuk single server.

2. Pengujian troughput *Load balancing* server untuk jaringan Univeristas Brawijaya terlihat pada tabel 6-4 dan grafik 6.8

Tabel 6-7 Tabel Troughput

Jumlah *Client*	RR (kb/s)	LC (kb/s)	Single (kb/s)
50	36.9366	32.617	37.6518
100	15.7913	15.8822	16.1137
150	10.85313	11.36647	10.5036
200	7.5222	8.40425	7.25805
250	5.19972	6.35244	5.85144
300	4.986833	5.379633	4.734133



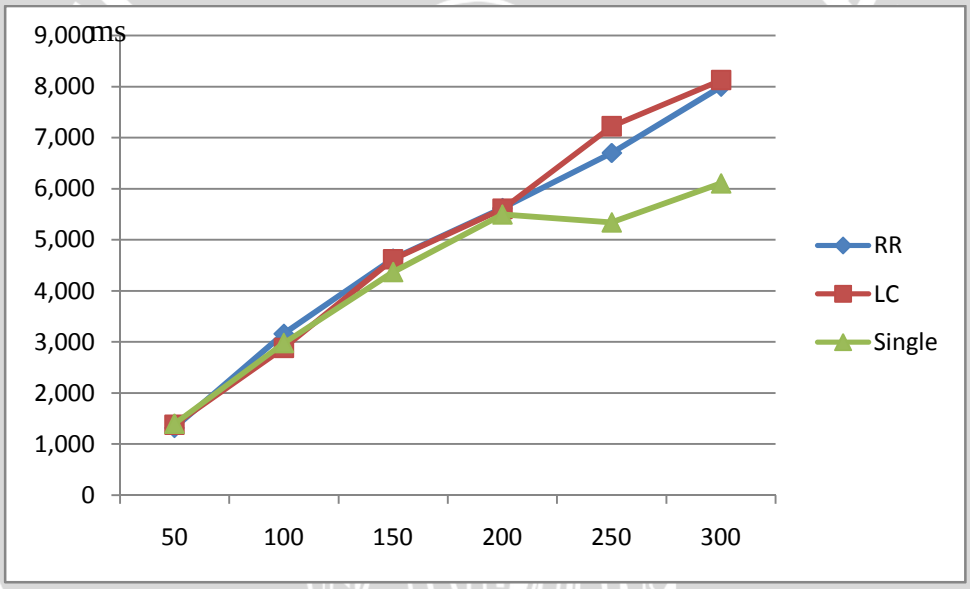
Gambar 6.11 Troughput Untuk Single Server, *Round robin* dan *Least connection*

Pada gambar 6.11 terlihat bahwa troughput dari pengujian semuanya tidak berbeda jauh. Untuk jumlah *client* sedikit nilai troughput paling baik jika menggunakan single server, tetapi jika jumlah *client* melebihi dari jumlah *client* maksimal maka nilai troughput terbaik tampak pada algoritma *Least connection*.

3. Dari pengujian juga didapatkan waktu respon dari masing-masing algoritma yang tampak pada tabel 6-5 dan grafik 6.9

Tabel 6-8 Waktu Respon Untuk Jaringan Lokal Universitas Brawijaya

Jumlah *Client*	RR (ms)	LC (ms)	Single (ms)
50	1365	1540	1313
100	3124	3079	3025
150	4114	4218	4499
200	6128	5386	6083
250	7102	6892	7498
300	8582	8023	8723



Gambar 6.12 Rata-rata Respon untuk Load Balacing Server untuk Single Server, Algoritma *Least connection* dan *Round robin*

Pada gambar 6.9 terlihat bahwa waktu repon dari pengujian semuanya tidak berbeda jauh. Untuk jumlah *client* sedikit nilai waktu respon paling baik jika menggunakan single server, tetapi jika jumlah *client* melebihi dari jumlah *client* maksimal maka nilai waktu respon terbaik tampak pada algoritma *Least connection*

Pengujian pada saat jam sibuk atau pada saat *offpeak* hasil yang ada tidak berbeda terlalu jauh dan tidak ada perubahan yang signifikan.

BAB VII
PENUTUP

7.1. Kesimpulan

Berdasarkan hasil pengujian dan analisis yang dilakukan terhadap kinerja sistem dapat diambil kesimpulan sebagai berikut:

1. Penggunaan *Load balancing* mengurangi tingkat error yang terjadi dalam mengakses website. Tingkat error yang terjadi dalam menggunakan single web server adalah 4.89% dari total *request* sedangkan jika menggunakan *Load balancing* adalah 0.69% dengan algoritma *Round robin* dan 0.49% pada *Least connection* dengan jumlah *client* 300 orang.

2. Troughput yang dihasilkan oleh *least connection* pada saat jumlah *client* kecil juga memiliki troughput yang kecil yaitu 32.6 kb/s jika dibandingkan dengan single server yang mempunyai troughput 37.65 kb/s dan *Round robin* dengan 36.93 kb/s. Ketika jumlah *client* mencapai puncak nilai troughput yang dihasilkan adalah *round robin* 4.99 kb/s, *least connection* 5.38 kb/s dan single server 4.73 kb/s.

3. Penggunaan Single server untuk jumlah *client* 50 lebih responsif jika dibandingkan dengan penggunaan load balancing. Akan tetapi penggunaan load balancing akan responsive pada saat jumlah *client* banyak. Load balancing akan responsif ketika jumlah *client* mencapai maksimal untuk memori 2GB.

4. Penggunaan load balancing lebih efektif jika sebuah server sudah tidak mampu menangani *request* yang ada, karena load balancing akan bekerja dengan efektif ketika beban yang ada sudah melebihi kapasitas sebuah single server. Algoritma yang baik dari hasil penelitian ini adalah algoritma *least connection*, karena algoritma ini membagi beban dengan melakukan pengecekan melalui server dengan jumlah koneksi paling sedikit

7.2. Saran

Saran yang dapat diberikan untuk pengembangan lebih lanjut adalah mengetahui karakteristik web yang digunakan dan spesifikasi server yang akan digunakan, sehingga distribusi beban dapat berjalan dengan baik. Untuk server sebaiknya juga diberikan beban yang berbeda untuk spesifikasi server yang berbeda.

DAFTAR PUSTAKA

anonymous, "http://www.worldfriend.web.id," [Online]. Available: http://www.worldfriend.web.id/pengertian-web-server. [Accessed oktober 2010]. [1]

I. M. Wiryana, "http://www.ilmukomputer.com/," [Online]. Available: http://www.ilmukomputer.com/2006/08/24/membangun-server-dengan-opensource/. [Accessed 6 oktober 2010]. [2]

Anonymous, "http://netmon.ub.ac.id," 6 oktober 2010. [Online]. Available: http://netmon.ub.ac.id. [Accessed 2010 oktober 2010]. [3]

anonymous, "http://repository.usu.ac.id/," [Online]. Available: http://repository.usu.ac.id/bitstream/123456789/20228/4/Chapter%20II.pdf. [Accessed 10 oktober 2010]. [4]

Anonymous, "http://id.wikipedia.org," [Online]. Available: http://id.wikipedia.org/wiki/Protokol_(komputer). [Accessed 6 oktober 2010]. [5]

anonymous, "http://library.binus.ac.id/eColls/eThesis/Bab2/2007-2-00161-IF-Bab%202.pdf," [Online]. Available: http://library.binus.ac.id/eColls/eThesis/Bab2/2007-2-00161-IF-Bab%202.pdf. [Accessed 17 februari 2012]. [6]

Dhoto, Jaringan Komputer, Surabaya: Politeknik Elektronika Negeri Surabaya - Institut Teknologi Sepuluh Nopember (PENS-ITS), 2007. [7]

Anonymous, "http://www.proweb.co.id/articles/web_design/website_adalah.html," [Online]. Available: http://www.proweb.co.id/articles/web_design/website_adalah.html. [Accessed 9 desember 2010]. [8]

I. Rijayana, Teknologi Load balancing Untuk Mengatasi Beban Server, 2005. [9]

W. H. A, 2005.

10]

E. S. Dewo, "Bandwidth dan Throughput," 2003. [Online]. Available: 11] http://dany.blog.stisitelkom.ac.id/files/2012/07/Bandwidth-dan-Throughput.pdf. [Accessed 6 Oktober 2010].

M. Schwartz, Telecomunications Network : Protocols, Modeling and 12] Analysis, New York: Addison Weasly Publishing Company, 1994.

M. Pietruszka and R. Host, "High-Availability Solutions:Building Low 13] Cost Data Centers Using Open Source UNIX-based ServerClusters," [Online]. Available: http://mpietruszka.com/pub/High%20Availability%20Solutions.pdf. [Accessed 20 januari 2011].

I. Red Hat, "Linux VirtualServer (LVS) for Red Hat Enterprise Linux 14] 5.1.," 2007. [Online]. [Accessed 2010 oktober 2010].

R. S. P. Pressman, Software Enginnering A Practitioner's Approach, 15] McGraw-Hill, 2001.

LAMPIRAN

Lampiran 1 : **Konfigurasi Sistem Operasi**

1. Server 1

```
    [root@server  ~]#vi  /etc/sysconfig/network-scripts/ifcfg-
eth0
       DEVICE=eth0
       BOOTPROTO=static
       HWADDR=00:24:8c:aa:9e:38
       IPADDR=175.45.185.43
       NETMASK=255.255.255.128
       ONBOOT=yes
    [root@server  ~]#vi  /etc/sysconfig/network-scripts/ifcfg-
eth0:1
       DEVICE=eth0:1
       BOOTPROTO=static
       HWADDR=00:24:8c:aa:9e:38
       IPADDR=175.45.185.42
       NETMASK=255.255.255.255
       ONBOOT=yes
    [root@server ~]# iptables -A RH-Firewall-1-INPUT -p tcp --
dport 80 -j ACCEPT
    [root@server ~]# iptables -A RH-Firewall-1-INPUT -p tcp --
dport 3636 -j ACCEPT
    [root@server ~]# yum install httpd php mysql-server php-
mysql
    [root@server ~]# chkconfig mysqld on
    [root@server ~]# chkconfig httpd on
    [root@server ~]# chkconfig Piranha-gui on
    [root@server ~]# chkconfig pulse on
    [root@server ~]# chkconfig ipvsadm on
    [root@server ~]# setenforce 0
    [root@server ~]# echo "1" >/proc/sys/net/ipv4/ip_forward
    [root@server ~]# service mysqld start
    [root@server ~]# service httpd start
    [root@server ~]# service ipvsadm start
    [root@server ~]# service Piranha-gui start
```

2. Server 2

```
    [root@server2  ~]#vi  /etc/sysconfig/network-scripts/ifcfg-
eth0
       DEVICE=eth0
       HWADDR=00:24:8c:aa:9e:32
       IPADDR=175.45.185.44
       NETMASK=255.255.255.128
       ONBOOT=yes
    [root@server2  ~]#vi  /etc/sysconfig/network-scripts/ifcfg-
lo:1
   DEVICE=lo:1
   IPADDR=175.45.185.42
   NETMASK=255.255.255.255
   ONBOOT=yes
    [root@server ~]# iptables -A RH-Firewall-1-INPUT -p tcp --
dport 80 -j ACCEPT
```

```
    [root@server ~]# iptables -A RH-Firewall-1-INPUT -p tcp --
dport 3636 -j ACCEPT
    [root@server ~]# yum install httpd php mysql-server php-
mysql
    [root@server ~]# chkconfig mysqld on
    [root@server ~]# chkconfig httpd on
    [root@server ~]# service mysqld start
    [root@server ~]# service httpd start
```

Lampiran 2 : **Konfigurasi httpd.conf**

```
ServerTokens OS
ServerRoot "/etc/httpd"
PidFile run/httpd.pid
Timeout 120
KeepAlive Off
MaxKeepAliveRequests 100
KeepAliveTimeout 15
<IfModule prefork.c>
            StartServers       8
            MinSpareServers    5
            MaxSpareServers   20
            ServerLimit      193
            MaxClients       193
            MaxRequestsPerChild  4000
</IfModule>
<IfModule worker.c>
            StartServers       2
            MaxClients       150
            MinSpareThreads    25
            MaxSpareThreads    75
            ThreadsPerChild    25
            MaxRequestsPerChild  0
</IfModule>
Listen 80
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule auth_digest_module modules/mod_auth_digest.so
LoadModule authn_file_module modules/mod_authn_file.so
LoadModule authn_alias_module modules/mod_authn_alias.so
LoadModule authn_anon_module modules/mod_authn_anon.so
LoadModule authn_dbm_module modules/mod_authn_dbm.so
LoadModule authn_default_module modules/mod_authn_default.so
LoadModule authz_host_module modules/mod_authz_host.so
LoadModule authz_client_module modules/mod_authz_client.so
LoadModule authz_owner_module modules/mod_authz_owner.so
LoadModule                              authz_groupfile_module
modules/mod_authz_groupfile.so
LoadModule authz_dbm_module modules/mod_authz_dbm.so
LoadModule authz_default_module modules/mod_authz_default.so
LoadModule ldap_module modules/mod_ldap.so
LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
LoadModule include_module modules/mod_include.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule logio_module modules/mod_logio.so
LoadModule env_module modules/mod_env.so
LoadModule ext_filter_module modules/mod_ext_filter.so
LoadModule mime_magic_module modules/mod_mime_magic.so
LoadModule expires_module modules/mod_expires.so
```

```
LoadModule deflate_module modules/mod_deflate.so
LoadModule headers_module modules/mod_headers.so
LoadModule clienttrack_module modules/mod_clienttrack.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule mime_module modules/mod_mime.so
LoadModule dav_module modules/mod_dav.so
LoadModule status_module modules/mod_status.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule info_module modules/mod_info.so
LoadModule dav_fs_module modules/mod_dav_fs.so
LoadModule vhost_alias_module modules/mod_vhost_alias.so
LoadModule negotiation_module modules/mod_negotiation.so
LoadModule dir_module modules/mod_dir.so
LoadModule actions_module modules/mod_actions.so
LoadModule speling_module modules/mod_speling.so
LoadModule clientdir_module modules/mod_clientdir.so
LoadModule alias_module modules/mod_alias.so
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_connect_module modules/mod_proxy_connect.so
LoadModule cache_module modules/mod_cache.so
LoadModule suexec_module modules/mod_suexec.so
LoadModule disk_cache_module modules/mod_disk_cache.so
LoadModule file_cache_module modules/mod_file_cache.so
LoadModule mem_cache_module modules/mod_mem_cache.so
LoadModule cgi_module modules/mod_cgi.so
LoadModule version_module modules/mod_version.so
Include conf.d/*.conf
Client apache
Group apache
ServerAdmin root@localhost
UseCanonicalName Off
DocumentRoot "/var/www/html"
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>
<Directory "/var/www/html">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
<IfModule mod_clientdir.c>
    ClientDir disable
</IfModule>
DirectoryIndex index.html index.html.var
AccessFileName .htaccess
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>
TypesConfig /etc/mime.types
DefaultType text/plain
<IfModule mod_mime_magic.c>
    MIMEMagicFile conf/magic
```

```
</IfModule>
HostnameLookups Off
ErrorLog logs/error_log
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\"
\"%{Client-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{Client-agent}i" agent
CustomLog logs/access_log combined
ServerSignature On
Alias /icons/ "/var/www/icons/"
<Directory "/var/www/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
<IfModule mod_dav_fs.c>
    DAVLockDB /var/lib/dav/lockdb
</IfModule>
ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
IndexOptions FancyIndexing VersionSort NameWidth=* HTMLTable
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip
AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*
AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core
AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^
DefaultIcon /icons/unknown.gif
ReadmeName README.html
HeaderName HEADER.html
IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t
AddLanguage en .en
```

```
LanguagePriority en
ForceLanguagePriority Prefer Fallback
AddDefaultCharset UTF-8
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz
AddHandler type-map var
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
Alias /error/ "/var/www/error/"
<IfModule mod_negotiation.c>
<IfModule mod_include.c>
    <Directory "/var/www/error">
        AllowOverride None
        Options IncludesNoExec
        AddOutputFilter Includes html
        AddHandler type-map var
        Order allow,deny
        Allow from all
        LanguagePriority en es de fr
        ForceLanguagePriority Prefer Fallback
    </Directory>
</IfModule>
</IfModule>
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-
response-1.0
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0
BrowserMatch "Microsoft Data Access Internet Publishing
Provider" redirect-carefully
BrowserMatch "MS FrontPage" redirect-carefully
BrowserMatch "^WebDrive" redirect-carefully
BrowserMatch "^WebDAVFS/1.[0123]" redirect-carefully
BrowserMatch "^gnome-vfs/1.0" redirect-carefully
BrowserMatch "^XML Spy" redirect-carefully
BrowserMatch "^Dreamweaver-WebDAV-SCM1" redirect-carefully
```

Lampiran 3 : **Konfigurasi php.ini**

```
[PHP]
engine = On
zend.ze1_compatibility_mode = Off
short_open_tag = On
asp_tags = Off
precision    =  14
y2k_compliance = On
output_buffering = 4096
zlib.output_compression = Off
implicit_flush = Off
unserialize_callback_func=
serialize_precision = 100
allow_call_time_pass_reference = Off
safe_mode = Off
safe_mode_gid = Off
safe_mode_include_dir =
safe_mode_exec_dir =
safe_mode_allowed_env_vars = PHP_
safe_mode_protected_env_vars = LD_LIBRARY_PATH
disable_functions =
disable_classes =
expose_php = On
max_execution_time = 30     ; Maximum execution time of each
script, in seconds
max_input_time = 60  ; Maximum amount of time each script may
spend parsing request data
memory_limit = 128M      ; Maximum amount of memory a script
may consume
error_reporting  =  E_ALL
display_errors = Off
display_startup_errors = Off
log_errors = On
log_errors_max_len = 1024
ignore_repeated_errors = Off
ignore_repeated_source = Off
report_memleaks = On
track_errors = Off
variables_order = "EGPCS"
register_globals = Off
register_long_arrays = Off
register_argc_argv = Off
auto_globals_jit = On
post_max_size = 8M
magic_quotes_gpc = Off
magic_quotes_runtime = Off
magic_quotes_sybase = Off
auto_prepend_file =
auto_append_file =
default_mimetype = "text/html"
doc_root =
client_dir =
extension_dir = "/usr/lib/php/modules"
enable_dl = On
file_uploads = On
upload_max_filesize = 2M
```

```
allow_url_fopen = On
default_socket_timeout = 60

[Date]
[Syslog]
define_syslog_variables  = Off
[mail function]
SMTP = localhost
smtp_port = 25
sendmail_path = /usr/sbin/sendmail -t -i
[SQL]
sql.safe_mode = Off
[ODBC]
odbc.allow_persistent = On
odbc.check_persistent = On
odbc.max_persistent = -1
odbc.max_links = -1
odbc.defaultlrl = 4096
odbc.defaultbinmode = 1
[MySQL]
mysql.allow_persistent = On
mysql.max_persistent = -1
mysql.max_links = -1
mysql.default_port =
mysql.default_socket =
mysql.default_host =
mysql.default_client =
mysql.default_password =
mysql.connect_timeout = 60
mysql.trace_mode = Off
[MySQLi]
mysqli.max_links = -1
mysqli.default_port = 3306
mysqli.default_socket =
mysqli.default_host =
mysqli.default_client =
mysqli.default_pw =
mysqli.reconnect = Off
[mSQL]
msql.allow_persistent = On
msql.max_persistent = -1
msql.max_links = -1
[PostgresSQL]
pgsql.allow_persistent = On
pgsql.auto_reset_persistent = Off
pgsql.max_persistent = -1
pgsql.max_links = -1
pgsql.ignore_notice = 0
pgsql.log_notice = 0
[Sybase]
sybase.allow_persistent = On
sybase.max_persistent = -1
sybase.max_links = -1
sybase.min_error_severity = 10
sybase.min_message_severity = 10
sybase.compatability_mode = Off
[Sybase-CT]
sybct.allow_persistent = On
sybct.max_persistent = -1
```

```
sybct.max_links = -1
sybct.min_server_severity = 10
sybct.min_client_severity = 10
[bcmath]
bcmath.scale = 0
[browscap]
[Session]
session.save_handler = files
session.save_path = "/var/lib/php/session"
session.use_cookies = 1
session.name = PHPSESSID
session.auto_start = 0
session.cookie_lifetime = 0
session.cookie_path = /
session.cookie_domain =
session.serialize_handler = php
session.gc_probability = 1
session.gc_divisor     = 1000
session.gc_maxlifetime = 1440
session.bug_compat_42 = 0
session.bug_compat_warn = 1
session.referer_check =
session.entropy_length = 0
session.entropy_file =
session.cache_limiter = nocache
session.cache_expire = 180
session.use_trans_sid = 0
session.hash_function = 0
session.hash_bits_per_character = 5
url_rewriter.tags =
"a=href,area=href,frame=src,input=src,form=fakeentry"
[MSSQL]
mssql.allow_persistent = On
mssql.max_persistent = -1
mssql.max_links = -1
mssql.min_error_severity = 10
mssql.min_message_severity = 10
mssql.compatability_mode = Off
mssql.secure_connection = Off
[Assertion]
[Verisign Payflow Pro]
pfpro.defaulthost = "test-payflow.verisign.com"
pfpro.defaultport = 443
pfpro.defaulttimeout = 30
[COM]
[mbstring]
[FrontBase]
[gd]
[exif]
[Tidy]
tidy.clean_output = Off
[soap]
soap.wsdl_cache_enabled=1
soap.wsdl_cache_dir="/tmp"
soap.wsdl_cache_ttl=86400
```

Lampiran 4 : **Konfigurasi MySQL dan LVS**

1. Konfigurasi MySQL pada my.cnf

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
client=mysql
old_passwords=1
[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

2. Konfigurasi LVS pada lvs.cf

```
serial_no = 27
primary = 172.20.3.4
service = lvs
backup = 0.0.0.0
heartbeat = 1
heartbeat_port = 539
keepalive = 6
deadtime = 18
network = direct
debug_level = NONE
virtual LoadBalance {
     active = 1
     address = 172.20.3.2 eth0:1
     vip_nmask = 255.255.255.0
     port = 80
     send = "GET /test HTTP/1.0\r\n\r\n"
     expect = "HTTP"
     use_regex = 0
     load_monitor = none
     scheduler = rr
     protocol = tcp
     timeout = 6
     reentry = 15
     quiesce_server = 0
     server Server1 {
         address = 172.20.3.4
         active = 1
         port = 80
         weight = 1
     }
     server Server2 {
         address = 172.20.3.5
         active = 1
         port = 80
         weight = 1
     }
}
```

Lampiran 5 : **Mapping Memori pada prosses httpd**

```
[root@server ~]# pmap 2822
2822:   /usr/sbin/httpd
00110000    108K r-x--  /lib/ld-2.5.so
0012b000      4K r-x--  /lib/ld-2.5.so
0012c000      4K rwx--  /lib/ld-2.5.so
0012d000      4K r-x--    [ anon ]
0012e000     36K r-x--  /lib/libcrypt-2.5.so
00137000      4K r-x--  /lib/libcrypt-2.5.so
00138000      4K rwx--  /lib/libcrypt-2.5.so
00139000    156K rwx--    [ anon ]
00160000    156K r-x--  /usr/lib/libapr-1.so.0.2.7
00187000      4K rwx--  /usr/lib/libapr-1.so.0.2.7
00188000     84K r-x--  /lib/libpthread-2.5.so
0019d000      4K r-x--  /lib/libpthread-2.5.so
0019e000      4K rwx--  /lib/libpthread-2.5.so
0019f000      8K rwx--    [ anon ]
001a1000    236K r-x--  /lib/libsepol.so.1
001dc000      4K rwx--  /lib/libsepol.so.1
001dd000     40K rwx--    [ anon ]
001e7000     64K r-x--  /lib/libresolv-2.5.so
001f7000      4K r-x--  /lib/libresolv-2.5.so
001f8000      4K rwx--  /lib/libresolv-2.5.so
001f9000      8K rwx--    [ anon ]
001fb000      4K r-x--  /usr/lib/httpd/modules/mod_authn_file.so
001fc000      8K rwx--  /usr/lib/httpd/modules/mod_authn_file.so
001fe000      4K r-x--  /usr/lib/httpd/modules/mod_authn_alias.so
001ff000      8K rwx--  /usr/lib/httpd/modules/mod_authn_alias.so
00202000     52K r-x--  /usr/lib/liblber-2.3.so.0.2.31
0020f000      4K rwx--  /usr/lib/liblber-2.3.so.0.2.31
00210000     96K r-x--  /usr/lib/libsasl2.so.2.0.22
00228000      4K rwx--  /usr/lib/libsasl2.so.2.0.22
00229000    180K r-x--  /usr/lib/libgssapi_krb5.so.2.2
00256000      4K rwx--  /usr/lib/libgssapi_krb5.so.2.2
00257000     72K r-x--  /lib/libz.so.1.2.3
00269000      4K rwx--  /lib/libz.so.1.2.3
0026a000    228K r-x--  /usr/lib/libldap-2.3.so.0.2.31
002a3000      4K rwx--  /usr/lib/libldap-2.3.so.0.2.31
002a4000   1356K r-x--  /lib/libc-2.5.so
003f7000      8K r-x--  /lib/libc-2.5.so
003f9000      4K rwx--  /lib/libc-2.5.so
003fa000     12K rwx--    [ anon ]
003fd000     24K r-x--  /usr/lib/httpd/modules/mod_auth_digest.so
00403000      8K rwx--  /usr/lib/httpd/modules/mod_auth_digest.so
00405000    152K r-x--  /usr/lib/libk5crypto.so.3.1
0042b000      4K rwx--  /usr/lib/libk5crypto.so.3.1
0042c000      4K r-x--  /usr/lib/httpd/modules/mod_authn_dbm.so
0042d000      8K rwx--  /usr/lib/httpd/modules/mod_authn_dbm.so
0042f000      8K r-x--  /usr/lib/httpd/modules/mod_authz_host.so
00431000      8K rwx--  /usr/lib/httpd/modules/mod_authz_host.so
00433000      8K r-x--  /usr/lib/httpd/modules/mod_authz_groupfile.so
00435000      8K rwx--  /usr/lib/httpd/modules/mod_authz_groupfile.so
00437000      8K r-x--  /usr/lib/httpd/modules/mod_authz_dbm.so
00439000      8K rwx--  /usr/lib/httpd/modules/mod_authz_dbm.so
0043b000      4K r-x--  /usr/lib/httpd/modules/mod_authz_default.so
0043c000      8K rwx--  /usr/lib/httpd/modules/mod_authz_default.so
0043e000     20K r-x--  /usr/lib/httpd/modules/mod_log_config.so
00443000      8K rwx--  /usr/lib/httpd/modules/mod_log_config.so
00445000      4K r-x--  /usr/lib/httpd/modules/mod_logio.so
00446000      8K rwx--  /usr/lib/httpd/modules/mod_logio.so
00448000    124K r-x--  /lib/libexpat.so.0.5.0
00467000      8K rwx--  /lib/libexpat.so.0.5.0
00469000   1192K r-x--  /lib/libcrypto.so.0.9.8e
00593000     76K rwx--  /lib/libcrypto.so.0.9.8e
```

```
005a6000     16K rwx--    [ anon ]
005aa000    592K r-x--  /usr/lib/libkrb5.so.3.3
0063e000     12K rwx--  /usr/lib/libkrb5.so.3.3
00641000     44K r-x--  /usr/lib/httpd/modules/mod_ldap.so
0064c000      8K rwx--  /usr/lib/httpd/modules/mod_ldap.so
0064e000      4K r-x--  /usr/lib/httpd/modules/mod_env.so
0064f000      8K rwx--  /usr/lib/httpd/modules/mod_env.so
00651000     20K r-x--  /usr/lib/httpd/modules/mod_mime_magic.so
00656000      8K rwx--  /usr/lib/httpd/modules/mod_mime_magic.so
00658000      8K r-x--  /usr/lib/httpd/modules/mod_expires.so
0065a000      8K rwx--  /usr/lib/httpd/modules/mod_expires.so
0065c000     16K r-x--  /usr/lib/httpd/modules/mod_deflate.so
00660000      8K rwx--  /usr/lib/httpd/modules/mod_deflate.so
00662000      8K r-x--  /usr/lib/httpd/modules/mod_clienttrack.so
00664000      8K rwx--  /usr/lib/httpd/modules/mod_clienttrack.so
00666000     16K r-x--  /usr/lib/httpd/modules/mod_info.so
0066a000      8K rwx--  /usr/lib/httpd/modules/mod_info.so
0066c000      4K r-x--  /usr/lib/httpd/modules/mod_dir.so
0066d000      4K --x--  /usr/lib/httpd/modules/mod_dir.so
0066e000      8K rwx--  /usr/lib/httpd/modules/mod_dir.so
00670000      4K r-x--  /usr/lib/httpd/modules/mod_actions.so
00671000      8K rwx--  /usr/lib/httpd/modules/mod_actions.so
00673000      8K r-x--  /usr/lib/httpd/modules/mod_speling.so
00675000      8K rwx--  /usr/lib/httpd/modules/mod_speling.so
00677000      8K r-x--  /usr/lib/httpd/modules/mod_clientdir.so
00679000      8K rwx--  /usr/lib/httpd/modules/mod_clientdir.so
0067b000      8K r-x--  /usr/lib/httpd/modules/mod_proxy_connect.so
0067d000      8K rwx--  /usr/lib/httpd/modules/mod_proxy_connect.so
0067f000      4K r-x--  /usr/lib/httpd/modules/mod_version.so
00680000      8K rwx--  /usr/lib/httpd/modules/mod_version.so
00683000     28K r-x--  /usr/lib/httpd/modules/mod_autoindex.so
0068a000      4K --x--  /usr/lib/httpd/modules/mod_autoindex.so
0068b000      8K rwx--  /usr/lib/httpd/modules/mod_autoindex.so
0068d000     56K r-x--  /usr/lib/httpd/modules/mod_rewrite.so
0069b000      8K rwx--  /usr/lib/httpd/modules/mod_rewrite.so
0069d000     28K r-x--  /usr/lib/httpd/modules/mod_proxy_ftp.so
006a4000      8K rwx--  /usr/lib/httpd/modules/mod_proxy_ftp.so
006a6000      4K r-x--  /usr/lib/libpspell.so.15.1.3
006a7000      4K rwx--  /usr/lib/libpspell.so.15.1.3
006ab000     44K r-x--  /usr/lib/httpd/modules/mod_dav_fs.so
006b6000      8K rwx--  /usr/lib/httpd/modules/mod_dav_fs.so
006b9000    272K r-x--  /lib/libssl.so.0.9.8e
006fd000     16K rwx--  /lib/libssl.so.0.9.8e
00701000     16K r-x--  /usr/lib/httpd/modules/mod_disk_cache.so
00705000      4K --x--  /usr/lib/httpd/modules/mod_disk_cache.so
00706000      8K rwx--  /usr/lib/httpd/modules/mod_disk_cache.so
00708000     20K r-x--  /usr/lib/httpd/modules/mod_mem_cache.so
0070d000      8K rwx--  /usr/lib/httpd/modules/mod_mem_cache.so
00714000     28K r-x--  /usr/lib/httpd/modules/mod_authnz_ldap.so
0071b000      8K rwx--  /usr/lib/httpd/modules/mod_authnz_ldap.so
0071d000     20K r-x--  /usr/lib/httpd/modules/mod_cgi.so
00722000      8K rwx--  /usr/lib/httpd/modules/mod_cgi.so
00724000     64K r-x--  /usr/lib/libbz2.so.1.0.3
00734000      4K rwx--  /usr/lib/libbz2.so.1.0.3
00739000     28K r-x--  /usr/lib/httpd/modules/mod_cache.so
00740000      8K rwx--  /usr/lib/httpd/modules/mod_cache.so
00742000    204K r-x--  /usr/lib/sse2/libgmp.so.3.3.3
00775000      4K rwx--  /usr/lib/sse2/libgmp.so.3.3.3
00776000     84K r-x--  /lib/libnsl-2.5.so
0078b000      4K r-x--  /lib/libnsl-2.5.so
0078c000      4K rwx--  /lib/libnsl-2.5.so
0078d000      8K rwx--    [ anon ]
00793000     40K r-x--  /lib/libnss_files-2.5.so
0079d000      4K r-x--  /lib/libnss_files-2.5.so
0079e000      4K rwx--  /lib/libnss_files-2.5.so
```

```
0079f000     44K r-x--  /lib/libgcc_s-4.1.2-20080825.so.1
007aa000      4K rwx--  /lib/libgcc_s-4.1.2-20080825.so.1
007ab000     88K r-x--  /lib/libselinux.so.1
007c1000      8K rwx--  /lib/libselinux.so.1
007c3000    236K r-x--  /usr/lib/libcurl.so.3.0.0
007fe000      4K rwx--  /usr/lib/libcurl.so.3.0.0
00802000      8K r-x--  /usr/lib/httpd/modules/mod_alias.so
00804000      4K --x--  /usr/lib/httpd/modules/mod_alias.so
00805000      8K rwx--  /usr/lib/httpd/modules/mod_alias.so
00808000      8K r-x--  /usr/lib/httpd/modules/mod_auth_basic.so
0080a000      8K rwx--  /usr/lib/httpd/modules/mod_auth_basic.so
0080c000    192K r-x--  /usr/lib/libidn.so.11.5.19
0083c000      4K rwx--  /usr/lib/libidn.so.11.5.19
0083d000     28K r-x--  /usr/lib/php/modules/dbase.so
00844000      4K rwx--  /usr/lib/php/modules/dbase.so
00845000      4K r-x--  /usr/lib/httpd/modules/mod_authn_default.so
00846000      8K rwx--  /usr/lib/httpd/modules/mod_authn_default.so
00848000     32K r-x--  /usr/lib/httpd/modules/mod_proxy_ajp.so
00850000      8K rwx--  /usr/lib/httpd/modules/mod_proxy_ajp.so
00852000     24K r-x--  /usr/lib/php/modules/pdo_mysql.so
00858000      4K rwx--  /usr/lib/php/modules/pdo_mysql.so
0085b000    980K r-x--  /lib/libdb-4.3.so
00950000     12K rwx--  /lib/libdb-4.3.so
00953000     76K r-x--  /usr/lib/php/modules/pdo.so
00966000      8K rwx--  /usr/lib/php/modules/pdo.so
0096a000      8K r-x--  /usr/lib/httpd/modules/mod_file_cache.so
0096c000      8K rwx--  /usr/lib/httpd/modules/mod_file_cache.so
0096e000      8K r-x--  /usr/lib/httpd/modules/mod_setenvif.so
00970000      8K rwx--  /usr/lib/httpd/modules/mod_setenvif.so
00991000     12K r-x--  /usr/lib/httpd/modules/mod_mime.so
00994000      8K rwx--  /usr/lib/httpd/modules/mod_mime.so
009ce000    124K r-x--  /lib/libpcre.so.0.0.1
009ed000      4K rwx--  /lib/libpcre.so.0.0.1
00a0c000      8K r-x--  /usr/lib/httpd/modules/mod_vhost_alias.so
00a0e000      8K rwx--  /usr/lib/httpd/modules/mod_vhost_alias.so
00a25000    100K r-x--  /usr/lib/libaprutil-1.so.0.2.7
00a3e000      4K rwx--  /usr/lib/libaprutil-1.so.0.2.7
00a7a000      4K r-x--  /usr/lib/httpd/modules/mod_suexec.so
00a7b000      8K rwx--  /usr/lib/httpd/modules/mod_suexec.so
00a92000     16K r-x--  /usr/lib/httpd/modules/mod_status.so
00a96000      8K rwx--  /usr/lib/httpd/modules/mod_status.so
00ad2000     72K r-x--  /usr/lib/httpd/modules/mod_proxy.so
00ae4000      8K rwx--  /usr/lib/httpd/modules/mod_proxy.so
00ae8000     28K r-x--  /usr/lib/httpd/modules/mod_proxy_http.so
00aef000      8K rwx--  /usr/lib/httpd/modules/mod_proxy_http.so
00af1000    372K r-x--  /usr/lib/libsqlite3.so.0.8.6
00b4e000      8K rwx--  /usr/lib/libsqlite3.so.0.8.6
00b96000     32K r-x--  /usr/lib/libkrb5support.so.0.1
00b9e000      4K rwx--  /usr/lib/libkrb5support.so.0.1
00ba7000    304K r-x--  /usr/sbin/httpd
00bf3000      8K rwx--  /usr/sbin/httpd
00bf5000     12K rwx--    [ anon ]
00c4a000      4K r-x--  /usr/lib/httpd/modules/mod_authn_anon.so
00c4b000      8K rwx--  /usr/lib/httpd/modules/mod_authn_anon.so
00ccd000     12K r-x--  /usr/lib/httpd/modules/mod_headers.so
00cd0000      8K rwx--  /usr/lib/httpd/modules/mod_headers.so
00d00000      4K r-x--  /usr/lib/httpd/modules/mod_authz_client.so
00d01000      8K rwx--  /usr/lib/httpd/modules/mod_authz_client.so
00d39000     16K r-x--  /lib/libuuid.so.1.2
00d3d000      4K rwx--  /lib/libuuid.so.1.2
00da7000    156K r-x--  /lib/libm-2.5.so
00dce000      4K r-x--  /lib/libm-2.5.so
00dcf000      4K rwx--  /lib/libm-2.5.so
00df1000      8K r-x--  /lib/libcom_err.so.2.1
00df3000      4K rwx--  /lib/libcom_err.so.2.1
```

```
00e03000      4K r-x--  /usr/lib/httpd/modules/mod_authz_owner.so
00e04000      8K rwx--  /usr/lib/httpd/modules/mod_authz_owner.so
00e19000     96K r-x--  /usr/lib/php/modules/mysqli.so
00e31000      8K rwx--  /usr/lib/php/modules/mysqli.so
00e6d000     44K r-x--  /usr/lib/php/modules/mysql.so
00e78000      4K rwx--  /usr/lib/php/modules/mysql.so
00e7a000     24K r-x--  /usr/lib/httpd/modules/mod_proxy_balancer.so
00e80000      8K rwx--  /usr/lib/httpd/modules/mod_proxy_balancer.so
00edc000     12K r-x--  /lib/libdl-2.5.so
00edf000      4K r-x--  /lib/libdl-2.5.so
00ee0000      4K rwx--  /lib/libdl-2.5.so
00eed000     36K r-x--  /usr/lib/httpd/modules/mod_include.so
00ef6000      8K rwx--  /usr/lib/httpd/modules/mod_include.so
00f00000     20K r-x--  /usr/lib/php/modules/pdo_sqlite.so
00f05000      4K rwx--  /usr/lib/php/modules/pdo_sqlite.so
00f4d000      8K r-x--  /lib/libkeyutils-1.2.so
00f4f000      4K rwx--  /lib/libkeyutils-1.2.so
00f57000     84K r-x--  /usr/lib/httpd/modules/mod_dav.so
00f6c000      8K rwx--  /usr/lib/httpd/modules/mod_dav.so
00fa5000     28K r-x--  /usr/lib/httpd/modules/mod_negotiation.so
00fac000      8K rwx--  /usr/lib/httpd/modules/mod_negotiation.so
00fb6000     16K r-x--  /usr/lib/httpd/modules/mod_ext_filter.so
00fba000      8K rwx--  /usr/lib/httpd/modules/mod_ext_filter.so
00fbc000   2828K r-x--  /usr/lib/httpd/modules/libphp5.so
0127f000    128K rwx--  /usr/lib/httpd/modules/libphp5.so
0129f000     28K rwx--    [ anon ]
019fa000    896K r-x--  /usr/lib/libstdc++.so.6.0.8
01ada000     16K r-x--  /usr/lib/libstdc++.so.6.0.8
01ade000      4K rwx--  /usr/lib/libstdc++.so.6.0.8
01adf000     24K rwx--    [ anon ]
0246d000    740K r-x--  /usr/lib/libaspell.so.15.1.3
02526000     16K rwx--  /usr/lib/libaspell.so.15.1.3
0252a000     16K rwx--    [ anon ]
0385a000   1156K r-x--  /usr/lib/mysql/libmysql*client*.so.15.0.0
0397b000    264K rwx--  /usr/lib/mysql/libmysql*client*.so.15.0.0
039bd000      4K rwx--    [ anon ]
06a19000   1200K r-x--  /usr/lib/libxml2.so.2.6.26
06b45000     20K rwx--  /usr/lib/libxml2.so.2.6.26
06b4a000      4K rwx--    [ anon ]
09661000   1856K rw---    [ anon ]
b7f5a000     80K rw-s-  /dev/zero (deleted)
b7f6e000    100K rw-s-  /dev/zero (deleted)
b7f87000     28K rw---    [ anon ]
bfdbe000     84K rw---    [ stack ]
total    21816K
```