ANALISIS PEMANFAATAN REVERSE PROXY UNTUK MENINGKATKAN EFESIENSI PELAYANAN WEB SERVER

LAPORAN TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Jurusan Teknik Informatika

DISUSUN OLEH:

Nama : Krisna Aditya
No Mahasiswa : 05523093

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA

2011

LEMBAR PENGESAHAN PEMBIMBING

ANALISIS PEMANFAATAN REVERSE PROXY UNTUK MENINGKATKAN EFESIENSI PELAYANAN WEB SERVER

TUGAS AKHIR

DISUSUN OLEH:

Nama : Krisna Aditya

No Mahasiswa : 05523093

DOSEN PEMBIMBING,

Drs.Suprivono, M.Sc

LEMBAR PENGESAHAN PENGUJI

ANALISIS PEMANFAATAN REVERSE PROXY UNTUK MENINGKATKAN EFESIENSI PELAYANAN WEB SERVER

TUGAS AKHIR

Disusun oleh:

Nama : Krisna Aditya

No Mahasiswa : 05523093

Telah Dipertahankan di Depan Sidang Penguji Sebagai Salah Syarat
Untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika
Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 25 November 2011

Tim Penguji	Tanda Tangan
Syarif Hidayat, S.Kom,MIT. Anggota I	
Raden Teduh Dirgahayu, Dr.S,T M.Sc. Anggota II	

Mengetahui,
Ketua Jurusan Teknik Informatika
Universitas Islam Indonesia

(Yudi Prayudi, S.Si, M.Kom)

iii

HALAMAN PERSEMBAHAN

Kupersembahkan Tugas Akhir Ini Untuk

Bapak dan ibuku tercinta yang memberikan dukungan moril maupun materil, serta doa dan menghadapi dengan penuh kesabaran, keikhlasan dalam membimbingku hingga aku bisa menyelesaikan masa kuliahku.

Istri dan anaku tercinta yang menjadi semangat dan inspirasiku

Buat adik-adikku tercinta semoga aku bisa jadi inspirasi bagi kalian dan kita dapat saling menolong dalam menjalani hidup ini.

Anggota keluarga yang lainnya yang selalu mendukung dan mendoakanku, terima kasih atas semuanya.
motivasi tiada henti pada jiwaku
semangat dan kerja kerasku terinspirasi kalian semua

HALAMAN MOTTO

"Sesungguhnya sesudah kesulitan ada kemudahan, maka apabila kamu telah selesai (dari suatu perkara), kerjakanlah dengan sungguh - sungguh urusan yang lain."
(Q.S Asy Syarh : ayat 6 dan 7)

"hai orang-orang yang beriman, mintalah pertolongan dari allah dengan kesabaran dan shalat. Sesungguhnya allah bersama orang-orang yang sabar"
(Q.S Al-baqarah : 153)

"Dan dia telah mengajarkan kepadamu apa yang belum kamu ketahui dan karunia allah itu amat besar padamu"
(Q.S An-Nisa : ayat 113)

Hidup ini cuma semenetara, janganlah kau sia-siakan waktumu untuk hal yang tidak berguna (anonim)

tetaplah tersenyum dan bersyukur, karena apa yang kau dapatkan adalah pemberian terbaik dari allah untukmu

KATA PENGANTAR

Assalamu'alaikum wr. wb.

Dengan segala hormat, penulis panjatkan puji syukur kepada Allah SWT, karena atas berkat rahmat dan karunia-Nya akhirnya penulis dapat menyelesaikan tugas akhir ini. Laporan tugas akhir dengan judul "Analisis Pemanfaatan Reverse Proxy Untuk Meningkatkan Efesiensi Pelayanan Web Server"sebagai salah satu syarat untuk meraih gelar sarjana S-1 di Universitas Islam Indonesia.

Tak lupa, dalam tugas akhir ini penulis telah dibantu oleh berbagai pihak, baik berupa bimbingan, semangat, maupun kerjasamanya. Oleh karena itu dalam kesempatan ini ijinkanlah penulis menyampaikan ucapan terima kasih kepada:

1. Allah swt atas segala rahmat dan karunia-Nya sehingga tugas akhir dan penyusunan laporan ini dapat terselesaikan dengan baik.
2. Bapak Yudi Prayudi, S.Si, M.Kom selaku Ketua Jurusan Teknik Informatika UII.
3. Bapak Drs.Supriyono, M.Sc, selaku Dosen Pembimbing tugas akhir , atas waktu dan kesabaran, serta pengertiannya dalam membantu penulis.
4. Kepada Ayah dan Ibunda kita tercinta yang telah memberikan dukungan doa tanpa henti.
5. Sahabat-sahabatku, terima kasih atas perhatian kalian selama ini.
6. Teman-teman mahasiswa Teknik Informatika, Semua angkatan yang selalu menemani dan berbagi pengetahuan dan pengalaman dengan saya selama saya di jogja,tanpa kalian saya bukan apa.

7. Serta semua pihak terkait yang tidak dapat penulis sebutkan satu per satu, yang telah membantu dari awal hingga akhir.

Tak ada yang gading yang tak retak, oleh karena itu penulis menyadari sepenuhnya bahwa masih banyak kekurangan dalam tugas Akhir ini, sehingga segala kritik dan saran akan penulis terima dengan rendah hati. Penulis sangat berharap semoga tugas akhir ini bermanfaat bagi semua pihak.

Yogyakarta, November 2011

Penulis

ABSTRAKSI

Web merupakan salah satu teknologi yang saat ini banyak digunakan dalam berbagai kebutuhan, baik digunakan untuk media promosi, sistem informasi, layanan multimedia, *social networking* dan lain -lain. server penyedia layanan web sering mengalami kendala dalam melayani permintaan layanan yang sangat tinggi dari klien. sehingga perlu adanya solusi dalam meningkatkan suatu layanan web, salah satu solusi yang tepat adalah menambahkan *reverse proxy* untuk meningkatkan kinerja suatu *web server* dengan sebelumnya membuat sebuah rancang bangun serta uji coba dalam mengimplementasikan *reverse proxy*.

Dalam implementasinya, proses rancangan sistem dan konfigurasi, yang dibutuhkan mencakup konfigurasi aplikasi Squid pada server proxy yang berfungsi sebagai *web cache request* yang datang dari klien, kemudian membangun beberapa *web server* yang berada dibelakang *server proxy* dimana site yang digunakan dalam *web server* adalah *wordpress*.

Dengan membuat sebuah *reverse proxy* untuk web server ini diharapkan menjadi solusi yang berguna dalam mengoptimalkan kinerja suatu web server.

Keywords : *Reverse proxy, proxy, web server*

TAKARIR

web server	Aplikasi layanan web
overload	Beban berlebih
performance	Kinerja
standalone	Berdiri sendiri
client	Pengguna jasa
bandwidth	Menunjukan ukuran data dalam jaringan
proxy	Perantara
proxy server	Komputer yang menjembatani akses internet
reverse proxy	Proxy server diletakan disisi server
forward proxy	Proxy server diletakan disisi klien
open proxy	Proxy server yang terbuka pengamanannya
cache	Suatu tempat penyimpanan sementara
open source	Kode program yang tidak berlesensi
distributed system	Sistem terdistribusi
resource	Sumber
hardware	Perangkat keras
general public license	Lisensi aplikasi berbasis open source
source code	Kode sumber program
high-availability	Ketersediaan tinggi
open system interconnection	Sistem standar interkoneksi komputer
header	Bagian atas suatu data
internet object	Obyek yang berda dalam internet
squid	Aplikasi proxy server
sintask	Tata kalimat dalam penulisan program
harddisk	Media penyimpanan kapasitas besar

front-end	Posisi depan
network address translation	Metode translasi alamat ip
firewall	Tembok api pengaman jaringan komputer
gateway	Gerbang penghubung di jaringan komputer
port	Lubang penghubung pada jaringan komputer
hypertext transfer protocol	Protokol layanan web
world wide web	Perangkat lunak web pertama di dunia
request	Permintaan
personal computer	Komputer pribadi yang didapat mudah
central procesing unit	Pusat proses eksekusi perintah dikomputer
interface	Peralatan atau progam penghubung jaringan
network	Jaringan

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1 Latar Belakang

Meningkatnya penggunaan internet di Indonesia memicu perkembangan teknologi informasi yang semakin pesat. Banyak konten-konten lokal bermunculan, menyajikan sebuah layanan web. Dengan begitu timbul persaingan diantara penyaji konten-konten lokal. Mereka berlomba-lomba meningkatkan system informasi yang handal guna memuaskan klien.

Seiring dengan pesatnya penggunaan internet maka *traffic* atau lalu lintas data semakin meningkat drastis, sehingga server layanan website terutama situs popular yang sering diakses klien memiliki beban proses yang tinggi dalam melayani *request* dari klien dan sangat memungkinkan *web server* tidak mampu melayani *request* dari klien yang sangat banyak. Hal ini bisa mengakibatkan *web server* mengalami *overload*, lambat, dan akhirnya server menjadi *down*. Jika server *down* bisa mempengarungi kepuasan klien dalam menggunakan layanan web tersebut.

Untuk mengatasi *overload* penyaji layanan web perlu mengupgrade *hardware* server ke performa yang lebih tinggi, Namun untuk solusi ini sepertinya hanya akan mengatasi masalah jangka pendek, karena apabila suatu saat *traffic* dari klien meningkat dan server tidak mampu lagi melayani beban proses yang lebih tinggi server harus di*upgrade* ke *perfomance* yang lebih tinggi lagi, di sisi lain untuk penyaji konten-konten lokal yang kecil dan baru berkembang tentulah menjadi masalah untuk meng*upgrade* server yang memperlukan biaya yang tinggi.

Oleh sebab itu dengan menggunakan *reverse proxy* dapat memperingan kinerja *web server* dan lebih hemat biaya.*Reverse proxy*, juga dikenal sebagai *web accelerator*, adalah metode untuk mengurangi beban pada *web server* yang sibuk

dengan menggunakan *web cache* antara server dan internet, sehingga mempercepat waktu respon dari web dan waktu download halaman.

Hal ini bisa menjadi solusi yang tepat bagi penyaji konten-konten lokal yang kecil dan baru berkembang. Dengan menggunakan *reverse proxy* atau web accelerator dapat menghemat biaya untuk upgrade server, *bandwidth* yang digunakan serta meningkatkan performa *web server*.

Sebelum solusi tersebut diimplementasikan maka perlu adanya analisa perbandingan antara *web server* yang standalone dengan *web server* menggunakan *reverse proxy* atau web accelerator.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah diuraikan di atas, maka dapat dirumuskan sebuah permasalahan yang dapat dijadikan sebagai acuan yaitu. Berapakah peningkatan efesiensi kecepatan, bandwidth dan traffic dengan menggunakan perbandingan antara *web server standalone* dan *web server* yang menggunakan *reverse proxy*

1.3 Batasan Masalah

Mengingat luasnya ruang lingkup permasalahan dalam *reverse proxy*, maka batasan masalah di tentukan sebagai berikut :

a. Analisis penggunaan *resource* komputermenampilkan sebuah halaman web.
b. Analisis *traffic* yang digunakan oleh *web server*
c. Web site yang terdapat di dalam *web server* menggunakan wordpress
d. Analisis perbandingan antara *web server* yang menggunakan *reverse proxy* dan tanpa *reverse proxy* di bangun terdiri dari1 buah komputer sebagai *web server,* satu buah sebagai *reverse proxy* server dan satu buah komputer sebagai klien

e. Klien adalah satu buah komputer yang berfungsi sebagai tester ke *web server*.

1.4 Tujuan Penelitian

Penelitian tugas akhir ini memiliki beberapa tujuan yaitu :

a. Melakukan studi terhadap *open source* terutama pada sistem operasi linux.
b. Melakukan studi terhadap *distributed system* dalam pengembangan aplikasi server.
c. Merancang suatu system untuk mengoptimalkan *resource hardware* dan *performance* server khususnya *web server.*

1.5 Manfaat Penelitian

Manfaat Penelitian antara lain :

a. Sebagai referensi dalam pengembangan aplikasi server.
b. Sebagai salah satu solusi untuk mengoptimalkan *performance web server*.

1.6 Metodologi Penelitian

Metode penelitian yang di lakukan adalah sebagai berikut :

a. Studi Pustaka

   Studi pustaka adalah mengumpulkan data-data yang berhubungan dengan kasus dan metode yang di gunakan.

b. Analisis pustaka dan system

   Analisis pustaka adalah melakukan penilaian terhadap hasil studi pustaka untuk mencari teknik yang paling tepat untuk kasus yang di hadapi dan metode yang akan di lakukan.

c. Implementasi

   Setelah melakukan analisis kemudian melakukan implementasi system secara.

d. Pengujian

Pengujian hasil implementasi dari *reverse proxy* yang telah dibuat dengan melakukan test dari klien dan menganalisa trafik yang terjadi serta beban akses yang diterima oleh *web server.*

1.7 Sistematika Penulisan

Laporan tugas akhir ini dibagi dalam lima bab, dengan rincian sebagai berikut.

BAB I : PENDAHULUAN

Berisi gambaran singkat mengenai latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitlian serta sistematika penulisan.

BAB II : LANDASAN TEORI

Berisi gambaran singkat tentang teori dasar kasus akan dibahas dalam tugas akhir mengenai sistem operasi linux, computer server *reverse proxy,* dan apliksi pendukung lainya.

BAB III : METODOLOGI

Bagian ini memuat analisis kebutuhan sistem yang akan dibuat, yang terdiri dari metode analisis, hasil analisis, kebutuhan perangkat lunak, cara pengujian dan perangkat keras.

BAB IV : HASIL DAN PEMBAHASAN

Berisi langkah langkah pengerjaan sistem mulai dari instalasi perangkat lunak, proses konfigurasi sistem, hingga proses pengujian sistem.

BAB V : PENUTUP

Memuat tentang kesimpulan serta saran saran dalam pembangunan sebuah cluster sistem yang baik.

BAB II
LANDASAN TEORI

2.1 PROXY

Proxy dapat dipahami sebagai pihak ketiga yang berdiri ditengah-tengah antara pihak kedua yang saling berhubungan dan berfungsi sebagai perantara, sehingga pihak pertama dan kedua tidak secara langsung berhubungan, akan tetapi berhubungan melalui perantara yaitu *proxy*. Sebagai perantara antara pengguna dan server-server di internet, *proxy server* bekerja dengan cara menerima permintaan layanan dari user, dan kemudian sebagai gantinya *proxy server* akan mewakili permintaan pengguna, ke server-server di internet yang dimaksudkan. Dengan demikian, sebenarnya *proxy server* hanya meneruskan permintaan pengguna ke *server* yang dimaksud, akan tetapi disini identitas peminta sudah berganti, bukan lagi pengguna asal, tetapi *proxy server* tersebut. *Server-server* di internet hanya akan mengeahui identitas *proxy server* tersebut, sebagai yang meminta, tetapi tidak akan tahu peminta sebenarnya (yaitu pengguna asalnya) karena permintaan yang sampai kepada server-server di internet bukan lagi dari pengguna asal, tetapi dari *proxy server*.

Proxy dalam pengertianya sebagai perantara, bekerja dalam berbagaqi jenis *protocol* komunikasi jaringan dan dapat berbeda pada level-level hirarki *layer protocol* komunikasi jaringan. Suatu perantara dapat saja bekerja pada *layer Data-Link, layer Network, layer Transport* dan *layer Application* dalam *layer* komunikasi jaringan menurut OSI *(Open System Interconnection).* Namun pengertian *proxy server* sebagian besar adalah untuk menunjuk suatu server yang bekerja sebagai *proxy* pada *layer application.*

Karena *proxy* bekerja pada *layer* aplikasi, proxy server dapat berjalan pada banyak aplikasi antara lain HTTP *Proxy* atau Web *Proxy* untuk *protoco*l HTTP atau Web, FTP *Proxy*, SMTP *Proxy* atau POP *Proxy*, NNTP *Proxy* untuk Newsgroup,

RealAudio atau Real Video *Proxy* untuk multimedia streaming, IRC *Proxy* untuk Internet Relay Chat, dan lain-lain. Masing-masing hanya akan meneriama menruskan atau melakuakan filter atas paket yang dihasilkan oleh layan yang bersesuaian.

Proxy aplikasi spesifik memiliki pilihan konfigurasi yang sangat banyak. Sebagai contoh, *web proxy* dapat di konfigurasi untuk menolak akses ke situs *web* tertentu pada waktu-waktu tertentu. Demikian juga *proxy* yang lain, misalnya dapat dikonfigurasi hanya memperbolehkan *download* FTP dan tidak memperbolehkan pengguna tertentu yang bisa memainkan file-file *RealAudio*, mencegah akses ke *email server* sebelum tanggal tertentu, dan masih banyak lagi.

2.1.1 Tiga Fungsi Utama atau Kegunaan Proxy

a. Firewall atau Filtering

Proxy server yang dikonfigursai secara benar, akan meningkatkan performa dan security. Karena *proxy server* bekerja pada *layer application* (dalam OSI layer), maka filtering yang dilakukan oleh *proxy* lebih "cerdas" daripada firewall biasa. *Proxy web server* dapat mengecek *URL* dari *outgoing request* (permintaan akses keluar) untuk halaman *web* dengan memeriksa pesan HTTP, GET dan POST. Dengan kemampuan ini administrator dapat melarang atau mengijinkan akses kedomain tertentu. *Firewall* biasa, tidak dapat melihat nama domain di dalam pesan tersebut, karena *firewall* hanya memeriksa *header* dari paket data.

b. Gateway

Untuk dapat mengakses *internet*, sebuah computer harus memiliki sebuah *IP public*. Hal ini menjadi mustahil karena *IP public* yang tersedia di dunia sangat terbatas, sehingga untuk dapat mengakses *internet* secara bersama-sama dengan menggunakan satu *IP public*, dibutuhkan satu sebuah sebuah computer yang memiliki *IP public*, yang di gunakan sebagai gateway komputer-komputer lain. Dalam hal ini *proxy server* juga berfungsi sebagai gateway. *Server* ini mempunyai dua *interface*, stau

antarmuka dengan internet dan satu untuk antarmuka dengan jaringan *local*.

c. Cache

Fungsi *proxy* yang lain adalah untuk *web caching*. *Caching* disini diartikan sebagai mekanisme penyimpanan *internet object* (gambar atau halaman *web*) dari suatu *web server* yang pernah diakses. Karena *proxy* bertindak sebagai perantara, maka *proxy* mendapat objek terlebih dahulu dari sumbernya untuk kemudian diteruskan kepada peminta sebenernya. Proses caching ini tidak kelihatan bagi klien (transparant), karena bagi klien tidak tampak siapa sebenarnya yang memberi objek yang dimintanya, apakah *proxy* yang mengambil dari *cache*-nya atau web server. Dari sisi klien, semua balasan langsung dari web server.

Dalam proses *caching*, *proxy* juga menyimpan objek tersebut untuk dirinya sendirinya dalam ruang disk yang disediakan. Sehingga pada suatu saat klien akan meminta suatu layanan ke *web server* dengan objek yang sama pada penyimpanan, *proxy* akan langsung dapat memberikan objek yang diminta tersebut, tanpa harus meminta lagi ke *web server*. Bila objek yang diminta tidak ada, baru *proxy* akan meminta pada *web server* dan memberikanya kepada klien.

Content yang disimpan di dalam hard disk disebut *cache object* yang nantinya akan digunakan jika klien kembali mengunjungi *web server*. Dalam kunjungan berikutnya, browser akan memeriksa *validasi* conten yang disimpanya, *validasi* ini dilakukan dengan membandingkan *header* content yang ada pada *cache object* dengan yang ada pada *web server*, jika content belum expired maka content tadi akan ditampilkan kembali ke browser.

Ada dua jenis metode caching, yaitu pasif dan aktif. Seperti telah diketahui objek yang disimpan bisa saja menjadi *expired*, untuk memriksanya dilakukan *validasi*. Jika *validasi* ini dilakukan setelah ada

permintaan dari klien, metode ini disebut pasif. Pada *caching* aktif, *proxy* akan mengamati obyek dan pola perubahanya. Misalkan pada sebuah objek didapati setiap harinya berubah setiap jam 12 siang dan klien biasanya membacanya jam 14, maka *proxy* tanpa diminta akan memperbaharui objek tersebut antara jam 12 dan 14 siang, dengan cara update otomatis ini waktu yang dibutuhkan pengguna untuk mendapatkan objek yang fresh akan semakin dikit.

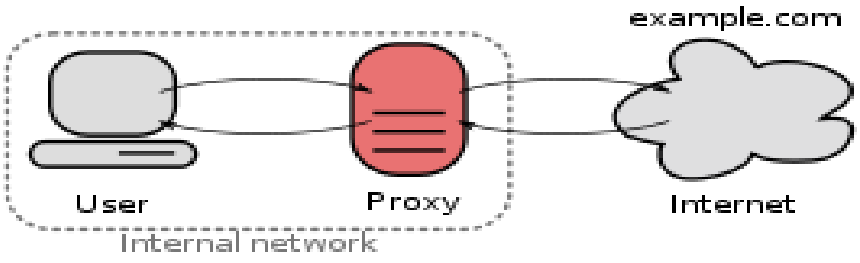
Pada kondisi tertentu, kapasitas penyimpanan akan terkuras habis oleh objek. Namun *proxy* mempunyai beberapa metode penghapusan untuk menjaga kapasitas. Penghapusan ini berdasarkan umur dan kepopuleran, semakin lama umur objek akan tinggi prioritasnya untuk dihapus. Serta untuk objek yang tidak popular akan lebih cepat terhapus juga.

Dengan adanya *caching* ini *bandwidth* yang di pakai anak lebih hemat, dan mempercepat akses.

2.1.2 Jenis Proxy Berdasarkan Cara Kerjanya

a. Forward Proxy

Proxy yang berjalan atau bekerja antara klien dan internet. Umumnya berfungsi sebagai *caching* halaman-halaman *web* yang pernah dikunjungi, pengalokasian bandwidth atau rule mengenai *user, content filtering* dan fungsi-fungsi lainnya yang diusung oleh bermacam aplikasi *proxy.*

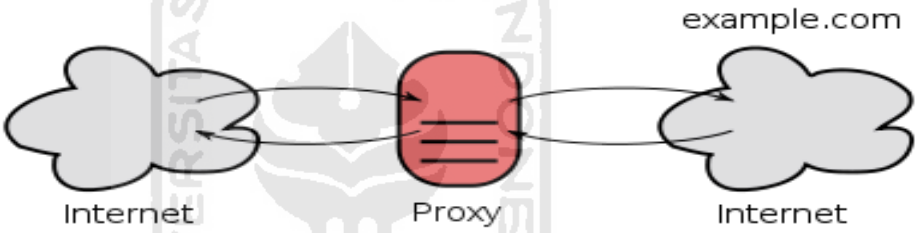


Gambar 2.1 Arsitektur jaringan forward proxy

b. Open Proxy

Open *proxy* adalah forward *proxy server* yang dapat di akses oleh *user* atau *host* lain di internet selain *host internal* kita untuk ber bagai keperluan. Termasuk menyusup kedalam *system host* atau *server* lain via *open proxy* kita.

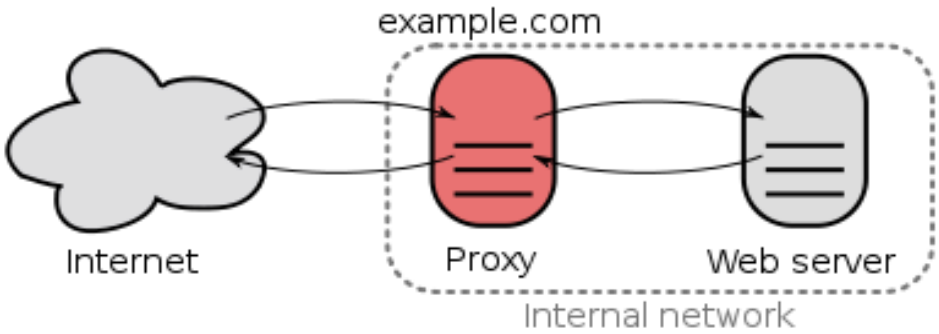
Open disini dalam arti *port-port* atau *service-service* tertentu yang bersifat vital yang dibiarkan terbuka tanpa penyaringan (*Filtered - Firewall*). Dampak dari *open proxy* sangat merugikan kita.



Gambar 2.2 *Arsitektur* jaringan *open proxy*

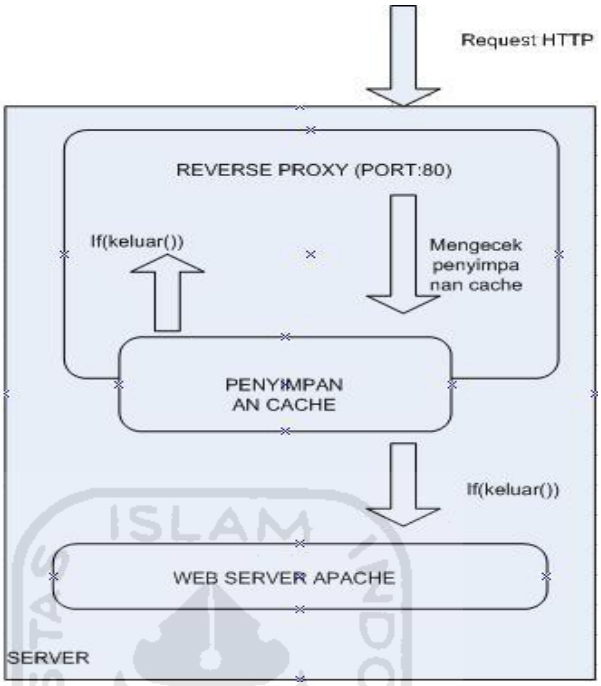
c. Reverse proxy

Proxy yang bekerja diantara *web server* dan klien, yang berfungsi sebagai *web acceleration* dan sebagai *front-end* untuk mengotrol serta melindungi akses ke *web server*. *Reverse proxy* biasanya diletakan di lingkungan *web server*. Karena semua lalu lintas yang datang dari internet dan dengan tujuan dari salah satu *web server* berjalan melalui *reverse proxy*.



Gambar 2.3 Arsitektur jaringan *reverse proxy*

Reverse proxy berjalan pada port 80 untuk melayani *request* HTTP. Pada port 80 *reverse proxy* tidak menggantikan fungsi web server, melainkan melanjutkan *request* HTTP tersebut ke web server untuk diolah. Setelah web server selesai mengolah *request* tersebut, web server akan mengembalikan lagi ke *reverse proxy*. *Reverse proxy* akan menyimpan *request* tersebut kedalam media penyimpanan sementara sebelum mengirimkan kembali *request* HTTP tersebut ke klien sebagai respons. Dan apabila ada *request* HTTP kembali yang sama *reverse proxy* akan langsung merespon *request* tersebut, tanpa harus meneruskan *request* tersebut ke web server. Contoh diagaram cara kerja *reverse proxy* di bawah ini.



Gambar 2.4 Cara kerja *reverse proxy*

Manfaat dari menginstal *reverse proxy* antara lain:

1. *Reverse proxy* dapat mengurangi overload pada web server dengan menyimpan content statis seperti gambar dan grafis pada cahce.
2. Mengurangi penggunaan sumber daya akibat klien yang lambat, dengan mengirim caching content *web server* perlahan-lahan terutama untuk halaman yang dinamis.
3. Meningkatkan keuntungan bisnis dengan mengurangi biaya oprasional termasuk biaya *bandwidth* yang dibutuhkan untuk melayani content
4. Mempercepat waktu respon dari web dan waktu download halaman untuk pengguna akhir.
5. Memberi pengalaman browsing lebih cepat dan lebih baik untuk pengunjung situs.
6. *Reverse proxy* juga bias sebagai *load balancing*
7. Meningkatkan keamanan sebagai lapisan tambahan untuk pertahanan web server.

2.2 Squid

Squid adalah adalah sebuah *aplikasi* yang digunakan sebagia proxy server dan *web cache*. Squid memeiliki banyak jenis kegunaan, meskipun squid sering digunakan untuk protocol HTTP dan FTP. Squid juga bias digunakan untuk beberapa protocol lainya seperti *Transport Layer Scurity* (TLS), *Secure Socket Layer* (SSL), *Internet Ghoper*, dan HTTPS. Sekarang squid juga mendukung IPv6 dan *Internet Content Adaptaqtion Protocol* (IACP).

Squid pada awalnya dikembangkan oleh Duane Wessels sebagai "Harvest *object cache*", yang merupakan bagian dari proyek Harvest yang dikembangkan di University of California. San Diego dan diperbaiki di National Science. Squid kini hampir secara eksklusif dikembangkan dengan cara sukarela.

Squid pada umumnya berjalan di atas system operasi Linux, meski squid juga bias berjalan di atas system operasi windows. Squid dirilis di bawah lisensi GNU *General Public License*, sehingga squid merupakan perangkat lunak bebas atu gratis. Squid biasnya bekerja pada port 80 (mode *reverse proxy*), port 8080 atau 3130 (port setandar yang biasa digunakan untuk *cache server*). Pada saat browser mengirimkan *header* permintaan, sinyal http *request* dikirimkan ke server. *Header* tersebut diterima squid dan dibaca. Dari hasil pembacaan, squid memprasing URL yg dibutuhkan lalu URL ini dicocokan dengan database *cache* yang ada.

Database ini beruba kumpulan *metadata* (semacam *header*) dari objek yang sudah ada didalam *harddisk*. Jika ada, objek akan dikirimkan ke klien dan tercatat dalam logging bahwa klien telah mendapat objek yg diminta. Dalam log kejadian tersebut akan dicatat sebagai TCP_HIT. Sebaliknya, jika objek yang diminta ternayta tidak ada, squid akan meminta ke server tujuan. Setelah mendapat objeknya, squid akan menyimpan objek tersebut ke dalam penyimpanan sementara. Selama dalam proses download objek ini dinamakan "*object* in transit" yang sementara menghuni ruang memori. Dalam masa download tadi, objek tersebut mulai dikirim ke klien hingga selesai, kejadian ini tercatat dalam log sebagai TCP_MISS.

2.2.1 Konfigurasi Squid

Konfigurasi, penggunaan dan metode Squid. Beberapa konfigurasi-konfigurasi mendasar squid antara lain :

a. ***http_port nomor port***. *Sintask* ini akan menunjukkan nomor port yang digunakan untuk menjalankan squid. Nomor *port*, digunakan untuk berhubungan dengan klien dan *peer*.

b. ***icp_port nomor port***. *Sintask* ini menunjukkan nomor port yang akan dipakai untuk menjalankan squid. Nomor port, digunakan untuk berhubungan dengan klien dan *peer*.

c. ***cache_peer nama_peer tipe_peer nomor_port_http nomor_port_icp* option**. *Sintask* dari *cache peer* ini digunakan untuk berhubungan dengan *peer* lain. *Peer* lainnya yang dihubungkan tipenya bergantung dari tipe *peer* yang telah dideklarasikan, bisa bertipe *sibling* maupun bertipe *parent*,dan *port* yang digunakan untuk hubungan ICP maupun HTTP juga dideklarasikan disini, sedangakan untuk parameter option disini ada bermacam-macam salah satunya adalah *default* yang berarti dia adalah satu-satunya *parent* yang harus dihubungi (jika bertipe *parent*) dan *proxy-only* yang berarti bahwa objek yang diambil dari *peer* tersebut tidak perlu disimpan dalam *hardisk* local.

d. ***Hierarcy_stoplist pola1 pola2*** *Sintaks* ini digunakan untuk menyatakan apa yang harus tidak diminta dari *peer*, melainkan langsung dari *web server origin*, jika pola1 dan pola 2 adalah parameter *cgi-bin ?* dan lain-lain, maka jika ada request URL yang mengandung karakter tersebut maka akan diambilkan langsung ke server origin.

e. ***Cache*_mem jumlah_memori (dalam bytes).** *Sintaks* ini akan menentukan batas atas jumlah memori yang digunakan untuk menyimpan.

f. ***Cache*_swap_low/high jumlah (dalam persen).** Squid akan menghapus objek yang ada didalam *hardisk*nya jika media tersebut mulai penuh. Batasan

penuh ini biasa diset pada *cache_swap_low* dan *cache_swap_high*. Bila batas *swap_low* telah tercapai maka squid mulai menghapus dan jika batas *swap_high* tercapai maka squid akan semakin sering menghapus.

g. ***Cache*_dir jenis_file_sistem direktori kapasitas_*cache* dir_1 jumlah dir_2**. *Sintaks* ini akan menjelaskan direktori *cache* yang dipakai, pertama adalah jenis file sistemnya, lalu didirektori mana *cache* tersebut akan disimpan, selanjutnya ukuran *cache* tersebut dalam *MegaBytes* (MB) lalu jumlah direktori level 1 dan direktori level 2 yang akan digunakan squid untuk menyimpan objeknya.

h. **ACL (*Access Control List*)**. Konfigurasi-konfigurasi lanjutan squid, selain sebagai *cache server*, squid yang memang bertindak sebagai "parent" untuk meminta objek dari kliennya dapat juga dikonfigurasi untuk pengaturan hak akses lebih lanjut, untuk pertama kali yang dibicarakan adalah ACL (access control list), ACL sendiri terdiri dari beberapa tipe antara lain :

- **Src** - *IP Address* asal yang digunakan klien
- **Dst** - *IP Address* tujuan yang diminta klien
- **Myip** - *IP Address* lokal dimana klien terhubung
- **srcdomain** - Nama domain asal klien
- **dstdomain** - Nama domain tujuan klien
- **srcdom_regex**- Pencarian pola secara *string* dari nama domain asal klien
- **dstdom_regex** - Pencarian pola secara *string* dari nama domain tujuan klien
- **Time** - Waktu dinyatakan dalam hari dan jam
- **Proto** - Protokol transfer (http, ftp, gopher)
- **Method** - Metode permintaan http (get, post, connect)

Berikutnya adalah control list yang akan digunakan untuk mengatur control dari ACL, control list tersebut antara lain :

- **http_access** - memperbolehkan *acess* http.
- **icp_access** - memperbolehkan *peer* untuk mengirimkan icp untuk *menquery* objek
- **miss_access** - memperbolehkan klien meminta objek yang belum ada (*miss*) didalam *cache.*
- **no_*cache*** - objek yang diminta klien tidak perlu disimpan ke *hardisk*.
- **always_direct** - permintaan yang ditangani langsung ke *server origin.*
- **never direct** - permintaan yang ditangani secara tidak langsung ke *server origin*.

i. ***Peer*ing**, konfigurasi *peer*ing. Maka dalam squid option atau parameter-parameter untuk pengaturan squid banyak sekali variasinya antara lain terdapat dalam contoh dibawah ini :

Cache_peer uii.ac.id sibling 8080 3130 proxy-only
Cache_peer klasiber.net parent 3128 3130 no-digest round-robin
Cache_peer tugasakhir.com parent 3128 3139 weight=2 no-digest

Untuk konfigurasi diatas, tipe *peer* baik sibling maupun parent, nomor port untuk hubungan *icp* maupun http telah dijelaskan pada bab sebelumnya, disini akan dibahas tentang option yang ada yaitu *proxy-only*, round-robin, dan *no-digest*.

Pada bagian sibling *cache peer* itu didefinisikan sebagai *proxy-only* yang berarti seluruh objek yang didapatkan dari *sibling* tidak akan disimpan ke dalam hardsik, begitu objek selesai didownload maka objek tersebut akan langsung diserahkan kepada klien dan objek akan dihapus dari memori. *Option* selanjutnya adalah *weight*, *option weight* adalah digunakan untuk pengaturan prioritas yang semakin tinggi nilainya maka dia adalah *cache*

parent yang akan dihubungi terlebih dahulu. *Option round-robin* berfungsi untuk memutar giliran parent mana yang akan diminta mencarikan objek, pada kasus ini jika ada terdapat banyak parent yang tidak diberi option *weight* untuk prioritas, maka *option round-robin* digunakan untuk menggilir *cache* yang akan dihubungi secara bergantian.

Sedang option *no-digest* adalah merupakan salah satu alternative squid berbicara dengan *peer*. *Cache digest* menggunakan cara mengumpulkan *header* masing-masing objek yang telah disimpan kedalam sebuah file. File ini yang nantinya akan diforward atau didownload oleh *peer* dengan menggunakan protokol http. *Header* ini dikumpulkan dalam versi terkompres dengan rasio tinggi.

Dengan memperoleh *cache*-digest dari *peer*, squid memperoleh kejelasan status ada tidaknya objek yang diminta, tanpa perlu bertanya dulu sebelumnya lewat protokol ICP. Dari sini squid dapat mengoptimisasi banwitdh, terutama jika *peer* terletak dalam jarak logika hoop yang cukup jauh. *Cache* digest itu sendiri *degenerate* secara berkala dan besarnya tergantung dari jumlah setiap objek, masing-masing objek tersebut disimpan dalam header sebanyak 10 bits.

2.2.2 *Object* Cache Dalam Squid

Pengaturan objek sebuah *cache server* merupakan salah satu hal yang perlu diperhatikan disini. Telah dijelaskan sebelumnya bahwa objek disimpan pada dua level *cache_dir* yang besar levelnya didefinisikan pada konfigurasi utama squid. Objek itu sendiri berisikan content URL yang diminta klien dan disimpan dalam bentuk file *binary*, masing-masing objek mempunyai metadata yang sebagian dari isinya disimpan didalam memori untuk memudahkan melacak dimana letak objek dan apa isi dari objek tersebut.

Umur objek Umur obect merupakan sebuah ukuran waktu yang dihabiskan sebuah objek untuk tinggal didalam *hardisk cache*.

metode penghapusan objek bisa melalui beberap algoritma penghapusan :

1. *Logistic Regression* adalah metode menghapus objek dengan kemungkinan *logistic regression* terkecil. Kemungkinan *logistic regression* bisa diartikan sebagai besarnya kemungkinan objek tersebut akan diakses diwaktu yang akan datang.
2. *Least Recently Used* adalah metode penghapusan objek berdasarkan waktu kapan objek tersebut terakhir diakses. Semakin lama waktunya, kemungkin dihapus juga akan semakin besar.
3. *Least Frequently Used* adalah metode penghapusan objek yang paling jarang diakses.
4. *First In First Out* adalah metode penghapusan berdasarkan waktu masuk ke dalam *cache_dir*, objek yang paling awal masuk, berarti itu adalah objek yang akan dihapus terlebih dahulu.
5. *Random* adalah metode menghapus objek secara acak.

2.2.3 Memori Squid

Squid menggunakan memori dalam banyak hal. Salah satu contoh pemakaiannya adalah untuk menyimpan objek yang popular, biasanya disebut *hot object*. Jumlah *hot object* yang disimpan dalam memori bisa diatur dengan option *cache_mem* pada *squid.conf*

Sebenarnya yang paling memakan memori adalah *metadata object*, karena kebanyakan objek sendiri sebenarnya disimpan dalam direktori *cache_dir hardsik* lokal. Semakin banyak kapasitas *cache_dir*, semakin banyak pula *metadat*a dan semakin membebani pemakaian memori. Pada kebanyakan kasus untuk setiap 1.000.000 jumlah objek, rata-rata dibutuhkan sebesar 72 MB memori untuk keseluruhan objek dan 1,25 MB untuk *metadata*. Jumlah objek ini bisa didapatkan dari besar *cache_dir* dibagi dengan jumlah rata-rata kapasitas objek, biasanya setiap objek bernilai 13 KB.

Mengingat pentingnya ketersediaan memori, penting untuk melihat sebagus apa aplikasi pengalokasian memori yang ada pada sistem operasi yang sedang bekerja. Secara *default* pada sistem operasi sudah tersedia rutin program untuk alokasi memori atau *malloc* (*memory allocation*). Namun pada beban yang sangat besar dan tanpa diimbangi penambahan memori yang memadai, *malloc* akan mencapai batas atas performansi dan kemudian mencapai status ketidakstabilan, dan squid akan menuliskan banyak pesan error pada log, misalnya seperti : "*xmalloc* : *Unable to allocate 4096 bytes!*".

Jika ini terjadi, langkah yang dapat dilakukan adalah melakukan penambahan memori, dan langkah kedua jika ingin lebih stabil adalah menginstall library untuk rutin program *malloc* yang lebih baru.

2.3 Web server

Web Server adalah sebuah perangkat lunak server yang menjadi tulang belakang dari *world wide web (*www*)* dan berfungsi menerima permintaan HTTP (*Hypertext Transfer Protocol*) atau HTTPS (*Hypertext Transfer Protocol Secure*) dari client yang dikenal dengan *web browser* serta mengirimkan kembali hasilnya dalam bentuk halaman-halaman web yang umumnya berbentuk dokumen HTML. Informasi yang dapat ditampilkan lewat web dapat berupa tulisan, gambar, dan bahkan audio visual. Dalam sebuah *web server* terdapat aplikasi-aplikasi yang digunakan, dibawah ini adalah aplikasi-aplikasi yang biasa dinstal dalam *web server*:

2.3.1 Web server apache

Server yang dapat dijalankan debrbagai sistem operasi dan berguna untuk melayani serta memfungsikan situs *web*. *Protocol* yang digunakan server ini adalah Http. *Apache* memiliki fitur-fitur canggih serta didukung oleh sejumlah antarmuka pengguna berbasis grafik (GUI) yang mempermudah penanganan server.

Sejarah awal mula apache adalah perangkat lunak sumber terbuka yang menjadi alternativ dari *web server Netscape* (sekarang dikenal *SUN Java System Web*

Server). Asal mula nama apache berasal dari kata *"A Patchy Server"*, server perbaikan yang penuh dengan tambalan *(patch).* Tambalan yang dimaksud adalah penambahan fitur dan penambalan *bug* dari *NCSA httpd 1.3* pada awal 1995. Tetapi pada situs resminya disebutkan bahwa "*Apache*" di pilih untuk menghormati suku asli Indian Amerika *Apache (Inde),* yang dikenal karena keahlian dan strategi perangnya. Sekarang versi 2 dari apache ditulis dari awal tanpa mengandung kode sumber dari *NCSA.*

2.4 DNS server

Fungsi utama dari sebuah sistem DNS adalah menerjemahkan nama-nama domain menjadi *IP address* ataupun sebaliknya, sehingga nama tersebut mudah diingat oleh pengguna internet. Fungsi lainnya adalah untuk memberikan suatu informasi tentang suatu host ke seluruh jaringan internet.

BAB III

METODOLOGI PENELITIAN

3.1 Analisis kebutuhan system

Analisis kebutuhan sistem merupakan bagian yang penting dalam rancang bangun sebuah sistem server, karena tahap inilah konsep awal sebuah sistem akan dibangun sehingga didapatkan rancangan awal sistem yang akan dibuat serta langkah-langkah apa saja yang akan dilakukan dalam proses konfigurasi server.

Analisis kebutuhan sistem yang digunakan dalam implementasi meningkatkan efisiensi pelayanan *web server* dengan menggunakan *reverse proxy* adalah dengan merancang skema jaringannya serta membandingkan sebuah *web server* yang menggunakan *reverse proxy* dengan yang tidak menggunakan *reverse proxy*.

Proses analisis dimulai dengan perancangan jaringan *web server* yang menggunakan *reverse proxy* dan *web server* yang tidak menggunakan *reverse proxy*. Dari analisis yang dilakukan, menghasilkan rancangan sebuah lingkungan *server* komputer yang terdiri dari satu buah komputer server yang berfungsi sebagai *web server*, satu buah komputer proxy server, dan satu buah komputer klien.

Dalam pembuatan rancang jaringan lingkungan server, langkah yang dilakukan dimulai dari instalasi sistem operasi pada masing-masing komputer. Setelah semua sistem operasi terinstal, dilanjutkan dengan instalasi aplikasi-aplikasi pada tiap-tiap komputer sesuai dengan fungsi tiap-tiap server. Apabila aplikasi sudah terinstal dilanjutkan dengan konfigurasi aplikasi-aplikasi yang akan berjalan.

Langkah terakhir adalah pengujian kinerja *web server* yang telah dibuat, guna membandingkan kinerja antara *web server* yang menggunakan reverse proxy dengan yang tidak menggunakan *reverse proxy*. proses pengujian dititik beratkan pada

seberapa banyak *request* yang bisa dilayani dan *transfer rate* yang hasilkan pada saat proses pengujian.

3.2 Metode analisis

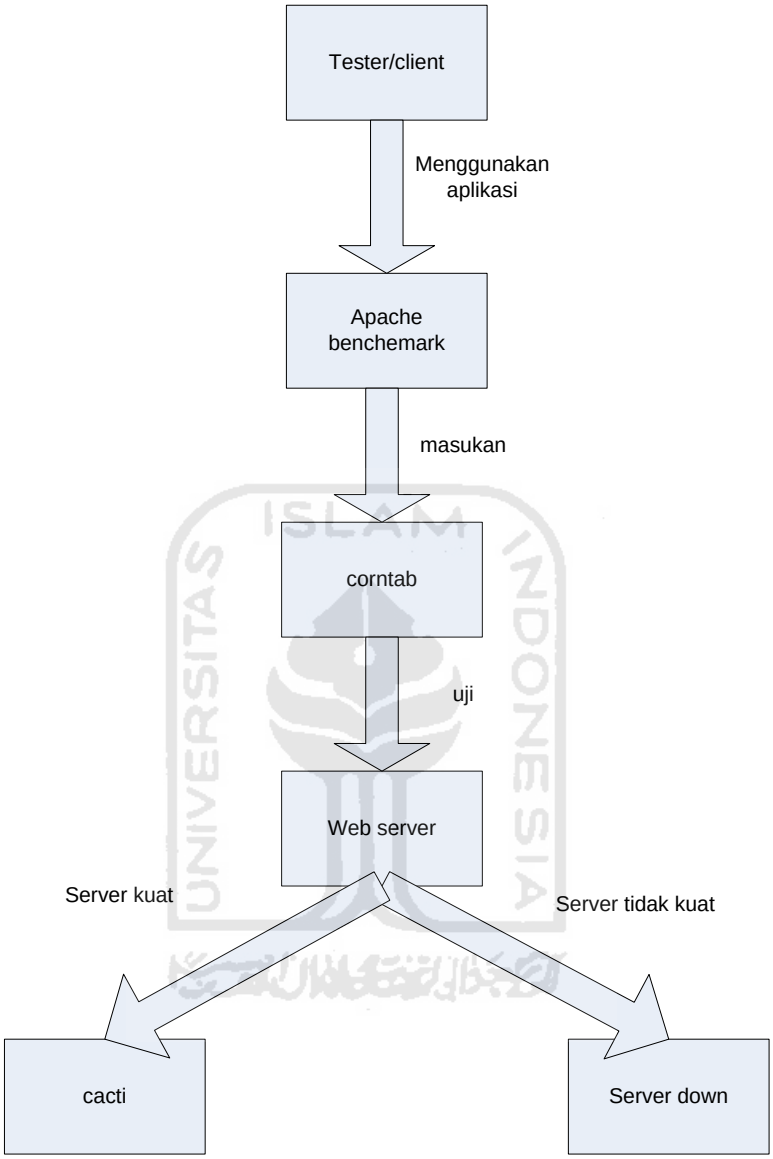
Metode yang di gunakan dalam implementasi Analisis Pemanfaatan Reverse Proxy Untuk Meningkatkan Efesiensi Pelayanan Web Server adalah dengan merancang skema jaringannya serta membandingkan antara web server menggunakan reverse proxy dengan yang tidak menggunakan reverse proxy.

3.3 Hasil analisis

Berdasarkan analisis perbandingan yang telah dilakukan antara *web server* yang menggunkan *reverse proxy* dengan *web server* tanpa menggunkan *reverse proxy*. Dapat diketahui apa saja yang menjadi kelebihan *web server* yang menggunakan *reverse proxy* dari *web server* tanpa menggunakan *reverse proxy*. Dilihat dari sisi pelayanan *web server*, *resource hardware* yang digunakan dan *traffic*.

3.4 Cara pengujian

Dalam komputer klien atau tester terdapat aplikasi benchmark yaitu apachebenchmark yang berfungsi untuk stressing web server, untuk mengetahui kinerja web server, dan selain apachebenchmark dalam komputer klien atau tester terdapat juga aplikasi corntab untuk menjalankan printah apachebencmark secara otomatis, serta di sisi web server terpasang aplikasi cacti untuk memonitoring jaringan dari situ bias terlihat jumlah traffic dan resource yang terpakai oleh web server. Ada pun gambaran pengujiannya sebagai berikut.



Gambar 3.1 flowchart cara pengujian

Jadi pertama-tama komputer klien akan menguji dengan apachebenchmark. Maka buatlah file bash untuk menjalakan printah otomatis apachebeechmark dalam corntab. Setelah itu masukan kedalam corntab dan corntab akan menjalankan secara otomastis sesuai dengan waktu yang telah di tentukan. Maka komputer tester mulai stressing web server.

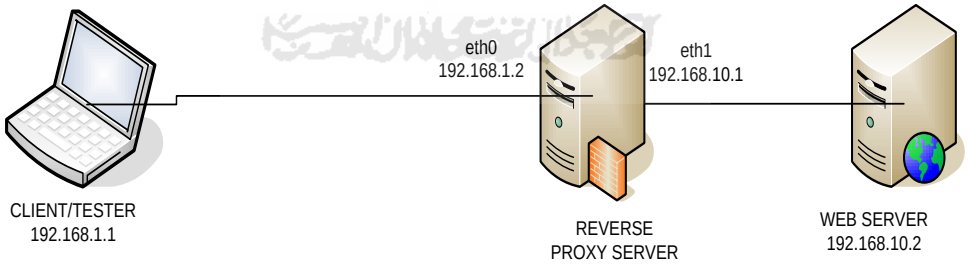
Dalam *web server* yang sudah terinstal aplikasi monitoring yaitu cacti akan membacanya dan akan mengeluarkan hasilnya, tetapi jika ternyata web server tidak kuat menerima stressing yang dilakukan oleh komputer client atau tester *web server* akan mengalami overload dan *web server* akan down.

3.5 Rancang jaringan

Dalam rancang jaringan perbandingan ini dibangun menjadi dua rancang jaringan arsitektur lingkungan server yaitu:

1. Rancang jaringan *web server* yang menggunakan reverse proxy

   rancang jaringan ini dibangun dengan terdiri dari sebuah *web server* dengan 1 buah *reverse proxy* dan digunakan sebagi *web accelerator* dengan menggunakan aplikasi squid serta satu buah klien sebagai tester. *Reverse proxy* bertindak sebagai *gateway* penghubung antara klien dan *web server*, serta berfungsi sebagai *cache web server* atau *cache serve* untuk meringankan kinerja *web server*, sehingga *request* dari klien tidak langsung menuju *web server* melainkan melalu *reverse proxy*. Sehingga beban *request* yang diterima oleh *web server* diperingan.



Gambar 3.1 Arsitektur *web server* yang menggunakan *reverse proxy*

2. Rancang jaringan *web server* yang tidak menggunakan *reverse proxy*

   Arsitektur jaringan ini ter diri dari sebuah *web server*, 1 buah PC sebagai gateway dan 1 buah klien sebagai penguji atau tester.

Gambar 3.2 Arsitektur *web server* tanpa menggunakan *reverse proxy*

Untuk uji coba pengaksesan digunakan 1 buah komputer klien terpisah. Untuk melakukan akses ratusan hingga ribuan hit ke server yang dilakukan secara simulasi dengan menggunakan tools yang mampu mengakses server dan mengambil halaman webnya sebanyak proses yang diinginkan.

Semua komputer baik komputer server maupun klien, terhubung dalam koneksi LAN, akan tetapi mempunyai domain IP yang berbeda antara server dan klien. Lingkungan server mempunyai network 192.168.10.0/24 dan klien mempunyai network 192.168.1.0/24.

3.3 Kebutuhan Perangkat Lunak

Perangkat lunak yang dibutuhkan untuk implementasi dari uji coba *web server* dengan *reverse proxy* dan tanpa *reverse proxy* adalah:

1. Sistem operasi ubuntu 9.04.
2. Sistem operasi ubuntu 10.04.
3. *DNS server* menggunakan BIND9 untuk membangun domain web site (berjalan di Linux).
4. Apache sebagai *web server*.
5. PHP5 sebagai bahasa pemrograman *web*
6. MySQL sebagai database *web server*
7. phpMyadmin sebagai aplikasi untuk mengelola database MySQL
8. Cacti, tool monitoring jaringan.
9. Corntab, tool dalam linux untuk menjalankan perintah secara otomatis dan berkala dalam waktu yang telah ditentukan.

10. *Apache benchmark*, tools yang digunakan untuk mengukur benchmark pada *web server* (berjalan di linux).

3.4 Kebutuhan Perangkat Keras

Dalam implementasinya, perbandingan antara *web server* yang menggunkan *reverse proxy* dengan web sever tanpa menggunakan *reverse proxy* dijalankan dalam dua buah komputer *notebook* dan satu buah PC (*Personal Computre*) yang memiliki spesifikasi sebagai berikut :

1. Notebook satu (*web server*)
   - Sistem operasi : *Linux ubuntu 9.04*
   - CPU : Intel Centrino duo (1,60 Ghz)
   - RAM : 2045 MB
   - HDD : 10 GB
2. Notebook dua (klien)
   - System operasi : *Linux ubuntu 10.04*
   - CPU : Intel Core 2 duo (2,10 Ghz)
   - RAM : 1024 MB
   - HDD : 10 GB
3. Personal Computer (Reverse Proxy)
   - Sistem Operasi : Linux Fedora 3
   - CPU : Intel Pentium 4 (2,60 Ghz)
   - RAM : 512 MB
   - HDD : 10 GB
4. Kabel utp

BAB IV

HASIL DAN PEMBAHASAN

4.1 Proses Instalasi dan Konfigurasi

Langkah-langkah yang dilakukan dalam proses instalasi dalam pembuatan tugas akhir ini adalah sebagai berikut:

4.1.1 Instalasi dan Konfigurasi Web Server

Langkah yang diperlukan untuk instalasi dan konfigurasi sebuah *web server* pada *Notebook*1 adalah:

1. Instal *web server* apache dan php5
   - Aktifkan konsol, kemudian ketik eksekusi perintah:
     # apt-get install apache2
     # apt-get install php5
2. Instalasi database MySQL server sebagai database *web server*, ada 3 tahap instalasi yang pertama yaitu:
   - Instal libapache2 sebagai autentifikasi MySQL dan install konektor php5 dan MySQL:
     # apt-get install libapache2-mod-auth-mysql
     # apt-get install php5-mysql
   - Yang terakhir instalasi MySQL-server dengan perintah
     # apt-get install mysql-server
   - Untuk instalasi MySQL server harus menentukan *password root*
3. Instalasi PHPMyAdmin dan konfigurasi untuk mengedit MySQL server
   - Ketikan perintah sebagai beriku
     # apt-get install phpmyadmin

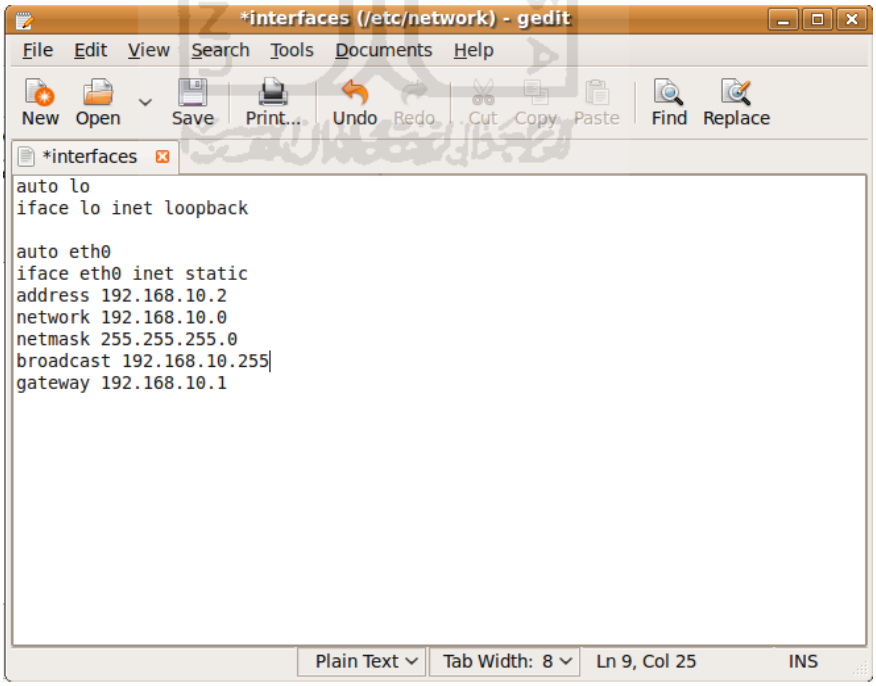
- Setelah proses instalasi selesai, sekarang menambahkan konfigurasi phpMyAdmin-shipped apaceh kedalam apache, untuk menampilakan phpMyAdmin dalam *browser*. Ketikan perintah:

  #ln –s /etc/phpmyadmin/apache.conf

  /etc/apache2/conf.d/phpmyadmin.conf
- Lalu reload apache2, ketikan perintah:

  # /etc/init.d/apache2 reload

4. Instalasi wordpress sebagai web site
   - Selesai menginstal *web server*. Download file wordpress dari http://wordpress.com jika telah selesai mendownload masukan file wordpress ke /var/www
   - Buka phphmyadmin dan login sebagi "*root*" buat databes dengan nama "*wordpress*"
   - Setelah wordpress terinstal, konfigurasi ip pada etc/network/interface seperti gambar di bawah ini



Gambar 4.1 Konfigurasi IP address *web server*

- Setelah memasukan dan menseting *IP address* pada /etc/network/interface, selanjutnya *restart network interface* dengan mengetikan perintah:

  # /etc/init.d/networking restart

5. Install tool cacti

   # apt-get install snmp snmpd rrdtool cacti

   - Seting snmp, buka file /etc/snmp/snmpd.conf

```
#                sec.name        source          community
com2sec          readonly        192.168.10.2    root
com2sec          readonly        localhost       root
com2sec          readonly        ujicoba.com     root

#                      sec.model  sec.name
group MyROGroup v1              readonly
group MyROGroup v2c             readonly
group MyROGroup usm                     readonly
group MyRWGroup v1              readwrite
group MyRWGroup v2c                     readwrite
group MyRWGroup usm                     readwrite

#                incl/excl  subtree       mask
view all          included .1            80
#                      context  sec.model  sec.level  match      read  write notif
access MyROGroup  ""           any         noauth     exact all  none  none
access MyRWGroup ""            any         noauth     exact all  all   none
syslocation ujicoba.com
syscontact sukse@berhasil.com
```

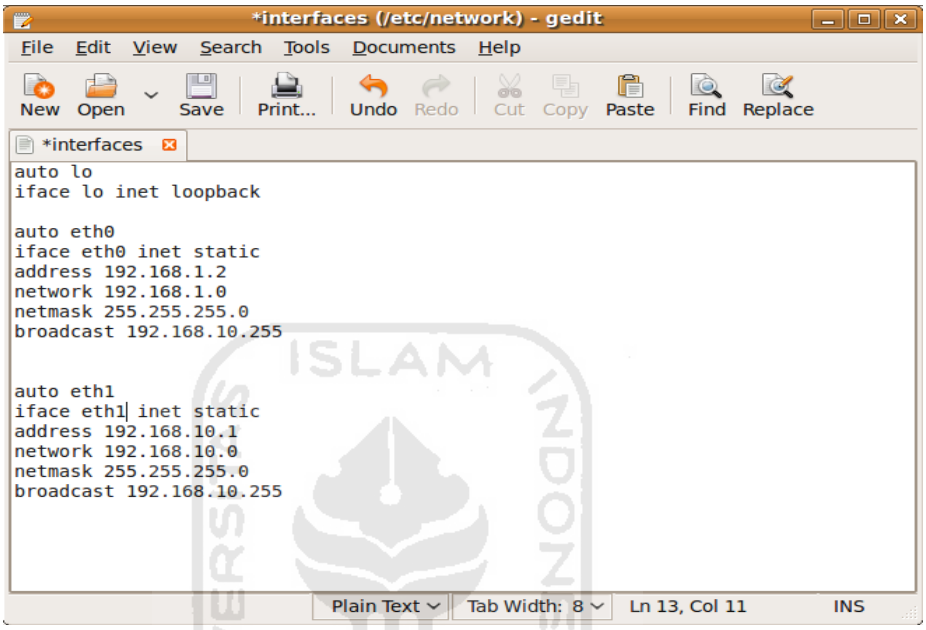
4.1.2 Instalasi dan konfigurasi reverse proxy

Menginstal *reverse proxy* pada PC (Personal Computer) yang menggunakan system operasi *Linux Ubuntu 9.04*, dalam konfigurasi *reverse proxy* menggunakan aplikasi squid. Berikut ini langkah-langkah Instalsi dan konfigurasi *reverse proxy*.

1. Seting IP address pada *reverse proxy* server

a. Menseting IP address server yang mempunyai dua *etherned card*, ketikan perintah sebagai berikut:

gedit /etc/network/interface



Gambar 4.2 Konfigurasi IP address *reverse proxy*

b. Setelah mengisikan *IP address*, maka *restart interface*. Ketik perintah untuk merestart *interface*.

/etc/init.d/networking restart

2. Instalasi squid sebagai aplikasi yang digunkan untuk membuat *reverse proxy*

a. Instal squid kedalam server, untuk mengistal squid ketikan perintah

apt-get install squid3

3. Berikutnya seting dan konfigurasi squid kedalam mode *reverse proxy*, edit file squid.conf dengan mengetikan.

gedit /etc/squid3/squid.conf

4. Masukan sintask berikut ini, kedalam file sqduid.conf.

#Parameter untuk menetapkan addresss atau alamat nomor port dimana squid akan mendengarkan request klien HTTP

http_port 192.168.1.2:80 vhost defaultsite=ujicoba.com

#Parameter aturan dalam squid untuk mengatur akses ke ujicoba.com

acl mysite url_regex ujicoba.com

acl all src 0.0.0.0/0.0.0.0

acl numeric_IPs url_regex -i ^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+

acl numeric_IPs url_regex -i (25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)

acl numeric_IPs url_regex -i (25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\:443/\?

#Parameter menetapkan cache lainnya dalam herarki cache, atau bagaimana memperlakukan “peer tetangga” dalam herarki cache.

cache_peer 192.168.10.2 parent 80 0 no-query originserver

#Parameter yang digunakan untuk menyaring mana saja yang diizinkan mengakses port HTTP

cache_peer_access 192.168.10.2 allow !numeric_IPs mysitecache_peer_access 192.168.10.2 deny numeric_IPscache_peer_access 192.168.10.2 deny all

http_access allow mysite

http_access deny all

#Parameter untuk menetapkan jumlah memori ideal untuk digunakan dalam objek In transit, Hot Objects dan negative cached.

cache_mem 8 MB

#Parameter untuk member tanda batas atau bawah kapan penggantian objek dilakukan

cache_swap_low 90

cache_swap_high 95

```
#Parameter untuk menetapkan ukuran maksimum objek yang disimpan dalam disk

maximum_object_size 1024 MB

#Parameter untuk menetapkan ukuran minimum objek yang disimpan dalam disk

minimum_object_size 0 bytes

#Parameter  untuk menetapkan ukuran maksimum obejek yang dijaga dalam memori cache

maximum_object_size_in_memory 1 KB


#Parameter untuk menetapkan objek mana yang dihapus dan diganti saat dibutuhkan  ruang disk

cache_replacement_policy heap LFUDA

memory_replacement_policy heap GDSF

#Parameter untuk menetapkan lokasi direktori cache, tipe, dan ukuranya

cache_dir ufs /var/spool/squid 5000 16 256

#Parameter untuk menetapkan lokasi host local alamat IP

hosts_file /etc/hosts

#Parameter untuk menentukan daftar dns mana yang digunkan

dns_nameservers /etc/resolv.conf

#Parameter untuk menetapkan penyimpanan file proses id squid

pid_filename /var/run/squid.pid

#Parameter untuk mencatat semua aktifitas client dalam file ini

cache_access_log /var/log/squid/access.log

#Parameter untuk menetapkan file logging cache dan lokasi direktori serta file tempat berbagi informasi umum cache keluar

cache_log /var/log/squid/cache.log
```

```
#Parameter untuk menetapkan file tempat menyimpan metadata objek-objek dalm disk

cache_swap_log /var/log/squid/swap.state

#Parameter untuk merefresh objek dalam cache

refresh_pattern ^http:          720     90%     432000

refresh_pattern -i \.(gif|png|jpg|jpeg|ico)$ 10080 90% 43200 override-expire ignore-no-cache ignore-private

refresh_pattern -i \.(iso|avi|wav|mp3|mp4|mpeg|mpg|swf|flv|x-flv)$ 43200 90% 432000 override-expire ignore-no-cache ignore-private

refresh_pattern -i \.(deb|rpm|exe|ram|bin|pdf|ppt|doc|tiff)$ 10080 90% 43200 override-expire ignore-no-cache ignore-private

refresh_pattern -i \.(zip|gz|arj|lha|lzh|tar|tgz|cab|rar)$ 10080 95% 43200 override-expire ignore-no-cache ignore-privaterefresh_pattern -i \.(html|htm|css|js|php|asp|aspx|cgi) 1440 40% 40320

refresh_pattern .               0       20%     4320

#Parameter untuk menetapkan user name dan group name dalam squid

cache_effective_user proxy

cache_effective_group proxy

#Parameter untuk menetukan berapa jumlah file lama yang dipertahankan  setelah rotasi file

logfile_rotate 10
```

5. Konfigurasi file squid.conf selesai. Selanjutnya restart squid dengan ketikan perintah:

   # /etc/init.d/squid3 restart

6. Instalasi DNS server untuk membuat domain www.ujicoba.com
   a. Untuk menginstal DNS ketikan perintah

      # apt-get install bind9

b. DNS telah terinstall lanjutkan dengan konfigurasi edit file /etc/bind/named.conf.local

gedit /etc/bind/named.conf.local

```
//include "/etc/bind/zones.rfc1918";
zone "ujicoba.com" {
 type master;
 file "/etc/bind/db.forward";
};
zone "1.168.192.in-addr.arpa" {
 type master;
 file "/etc/bind/db.reverse";
};
```

7. Buat file zona forward di /var/bind/db.forward

```
$TTL    604800
@       IN      SOA     server.ujicoba.com. root.ujicoba.com. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419500         ; Expire
                         604800 )       ; Negative Cache TTL
@       IN      NS      server.ujicoba.com.
@       IN      A       192.168.1.2
server  IN      A       192.168.1.2
www     IN      CNAME   server.ujicoba.com.
```

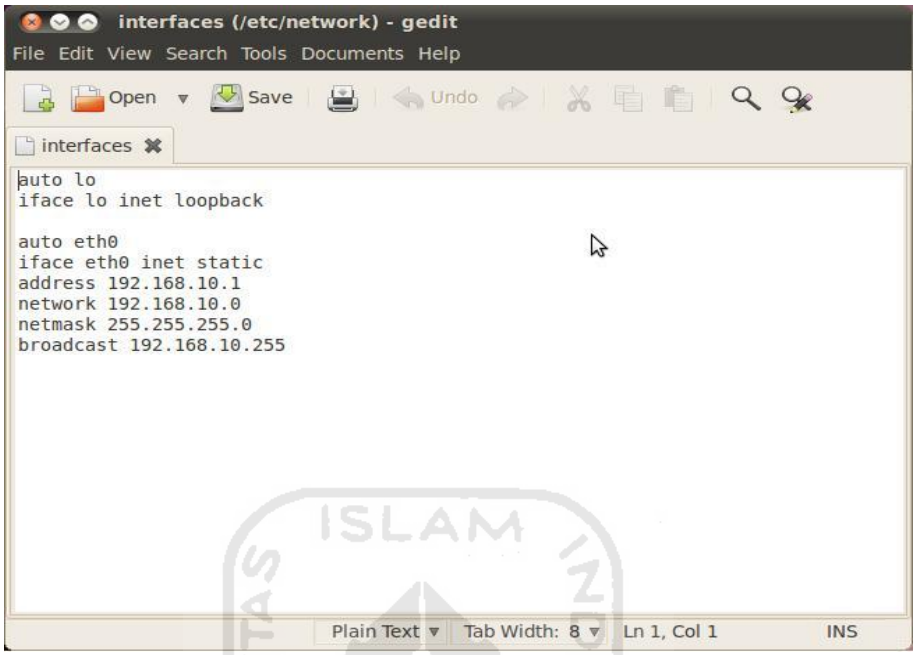
8. Buat file zona reverse di /var/bind/db.reverse

```
$TTL    604800
@       IN      SOA     server.ujicoba.com. root.ujicoba.com. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419500         ; Expire
                         604800 )       ; Negative Cache TTL
@       IN      NS      server.ujicoba.com.
2.1.168 IN      PTR     server.ujicoba.com.
2.1.168 IN      PTR     www.ujicoba.com.
```

4.1.3 Instal dan konfigurasi client

Instalasi klien dilakukan pada *Notebook*2 yang menggunakan sistem operasi *linux ubuntu 10.04.* Berikut langkah-langkah install dan konfigurasi:

1. Seting IP address pada computer
   # gedit /etc/network/interface



Gambar 4.3 Konfigurasi IP address client

2. Install *tool benchmark AB* (*apache benchmark*)
3. # apt-get install apache2-utils

4.2 Proses pengujian

Pada dasarnya *reverse proxy* ini dirancang untuk mempercepat proses pelayanan suatu *Web Server* terhadap permintaan halaman web oleh beberapa client. Pengujian dan analisa perbandingan yang dilakukan terhadap *web server* yang menggunakan *reverse proxy* dengan yang tidak menggunkan *reverse proxy* dimaksudkan untuk menunjukkan bahwa *web server*s yang menggunakan *reverse proxy* bisa meningkatkan proses pelayanan suatu *web server*.

4.2.1 Perangkat pengujian dan analisis perbandingan

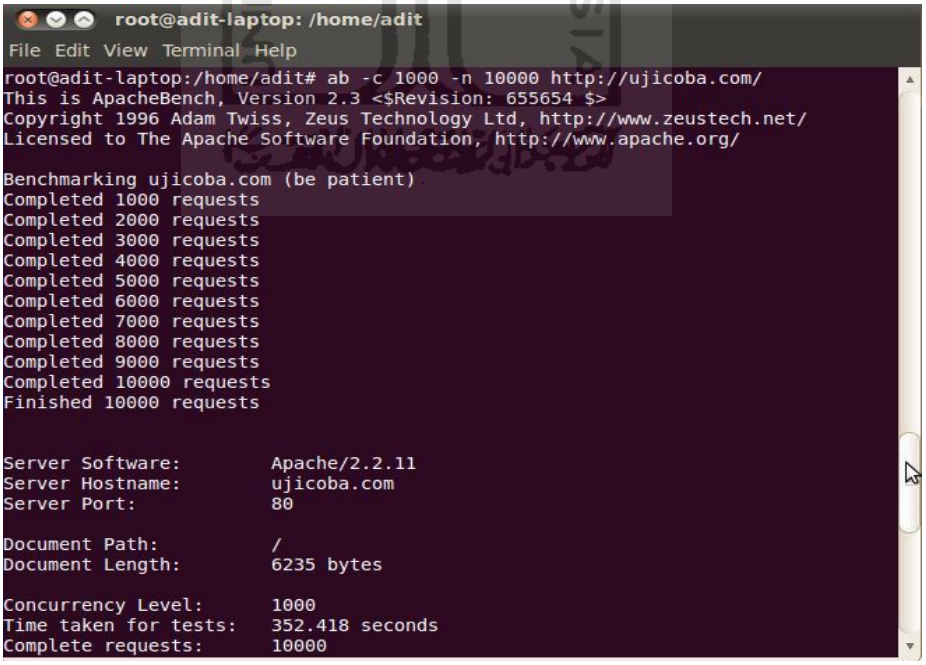
Peralatan yang dipergunakan dalam penyusunan penelitian ini untuk menguji dan menganalisa perbandingan kinerja *web server* yang menggunakan *reverse proxy* dengan yang tidak menggunakan *reverse proxy*, berupa 2 buah komputer *notebook*, 1 buah PC (*Personal Computer)*

4.2.2 Pengujian kinerja web server

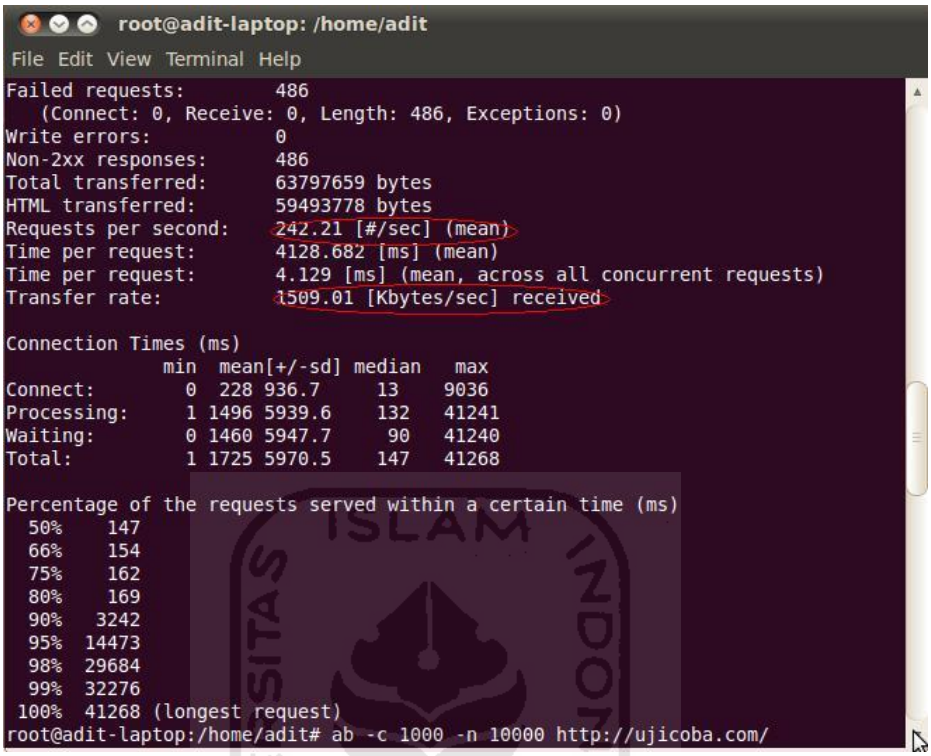
Pengujian dilakukan dengan mengaktifkan layanan *server* kemudian diakses oleh client menggunakan *tool benchmark AB* (*apache benchmark*) yang merupakan utilitas bawaan pada *web server apache*, sehingga *apache benchmark* sudah otomatis terinstal pada saat kita menginstal *web server apache*. Gunakan *tools apache benchmark* diclient untuk mengukur *benchmark* kinerja *cluster web server*.

#ab -c 100 -n 1000 http://ujicoba.com/ Keterangan :

- -c 100 adalah jumlah *concurrent connection*, sebanyak 100 koneksi secara bersamaan.
- -n 1000 adalah jumlah *request* yang mau dikirim, sebanyak total 1000 request.
- http://www.ujicoba.com adalah *website* yang akan ditest *benchmarknya*.



Gambar 4.4 Benchmark web server menggunakan apache benchmark



Gambar 4.5 Benchmark web server menggunakan apache benchmark

4.2.3 Hasil pengujian

Dari hasil pengujian perbandingan yang sudah dilakukan, dengan melakukan benchmark *web server* yang menggunakan reverse proxy dengan yang tidak menggunakan reverse proxy. Dengan menguji beberapa halaman-halaman web site utama.

1. Hasil Uji Satu

   Hasil uji pertama dilakukan dengan melakukan benchmark *web server* yang menggunakan dan yang tidak menggunakan reverse proxy. Menggunakan parameter sebagai brikut:

   Request : 1000

Concurention : 100 - 1000

Halaman : http://ujicoba.com/

#ab -c 100 -n 1000 http://ujicoba.com/

Tabel 4.1 Hasil uji satu physical memory.

Concurention	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
	Average (Bytes)	Average (Bytes)
100	597,10M	791,87M
200	551,63M	918,76M
300	573,66M	1002,74M
400	591,33M	Over load
500	588,87M	Over load
600	563,12M	Over load
700	598,99M	Over load
800	600,02M	Over load
900	623,81M	Over load
1000	664,95M	Over load

Tabel 4.2 Hasil uji satu CPU Utilization.

Concurention	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
	Average (%)	Average (%)
100	10	38
200	13	44
300	13	50
400	16	Over load
500	16	Over load
600	18	Over load
700	19	Over load
800	20	Over load
900	23	Over load
1000	23	Over load

Tabel 4.3 Hasil uji satu Traffic

Concurention	Traffic	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
		Average (bits)	Average (bits)
100	Inbound	1,03k	3,04k
	Outbound	5,93k	16,56k
200	Inbound	2,76k	3,68k

	Outbound	11,42k	37,78k
300	Inbound	3,99k	4,32k
	Outbound	11,90k	40,84k
400	Inbound	5,87k	Over load
	Outbound	17,59k	Over load
500	Inbound	6,44k	Over load
	Outbound	19,69k	Over load
600	Inbound	8,68k	Over load
	Outbound	20,88k	Over load
700	Inbound	8,97k	Over load
	Outbound	21,96k	Over load
800	Inbound	9,58k	Over load
	Outbound	23,98k	Over load
900	Inbound	9,81k	Over load
	Outbound	25,40k	Over load
1000	Inbound	11,67k	Over load
	Outbound	28,87k	Over load

2. Hasil uji dua

Hasil uji dua dilakukan dengan melakukan benchmark *web server* yang menggunakan dan yang tidak menggunakan reverse proxy. Menggunakan parameter sebagai brikut:

Request : 1000

Concurention : 100 - 1000

Halaman : http://ujicoba.com/?page_id=2

#ab -c 100 -n 1000 http://ujicoba.com/?page_id=2

Tabel 4.4 Hasil uji dua physical memory.

Concurention	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
	Average (Bytes)	Average (Bytes)
100	533,42M	778,80M
200	568,76M	988,55M
300	588,54M	1,14G
400	592,71M	Over load
500	598,32M	Over load
600	581,62M	Over load
700	599,25M	Over load
800	611,36M	Over load
900	645,82M	Over load
1000	655,51M	Over load

Tabel 4.5 Hasil uji dua CPU Utilization.

Concurention	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
	Average (%)	Average (%)
100	11	33
200	13	40
300	14	49
400	15	Over load

500	18	Over load
600	20	Over load
700	20	Over load
800	20	Over load
900	22	Over load
100	23	Over load

Tabel 4.6 Hasil uji dua Traffic

Concurention	Traffic	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
		Average (bits)	Average (bits)
100	Inbound	1,28k	3,56k
	Outbound	6,14k	16,79k
200	Inbound	2,96k	4,78k
	Outbound	11,55k	37,43k
300	Inbound	4,33k	4,98k
	Outbound	11,89k	40,75k
400	Inbound	5,88k	Over load
	Outbound	17,75k	Over load
500	Inbound	6,66k	Over load
	Outbound	20,01k	Over load
600	Inbound	8,73k	Over load
	Outbound	21,55k	Over load

700	Inbound	8,97k	Over load
	Outbound	22,44k	Over load
800	Inbound	9,58k	Over load
	Outbound	24,96k	Over load
900	Inbound	9,89k	Over load
	Outbound	25,43k	Over load
1000	Inbound	11,55k	Over load
	Outbound	28,36k	Over load

3. Hasil uji tiga

Hasil uji tiga dilakukan dengan melakukan benchmark *web server* yang menggunakan dan yang tidak menggunakan reverse proxy. Menggunakan parameter sebagai brikut:

Request : 1000

Concurention : 100-1000

Halaman : http://ujicoba.com/?p=1

#ab -c 100 -n 1000 http://ujicoba.com/?p=1

Tabel 4.7 Hasil uji tiga physical memory.

Concurention	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
	Average (Bytes)	Average (Bytes)
100	583,37M	787,54M
200	567,84M	973,24M
300	589,21M	1002,74M

400	593,65M	Over load
500	597,90M	Over load
600	571,43M	Over load
700	598,78M	Over load
800	609,35M	Over load
900	651,18M	Over load
1000	674,05M	Over load

Tabel 4.8 Hasil uji tiga CPU Utilization.

Concurention	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
	Average (%)	Average (%)
100	9	38
200	11	44
300	12	50
400	16	Over load
500	17	Over load
600	18	Over load
700	20	Over load
800	20	Over load
900	22	Over load
1000	23	Over load

Tabel 4.9 Hasil uji tiga Traffic

Concurention	Traffic	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
		Average (bits)	Average (bits)
100	Inbound	1,32k	3,78k
	Outbound	5,86k	16,85k
200	Inbound	2,40k	3,55k
	Outbound	11,53k	37,64k
300	Inbound	3,82k	4,11k
	Outbound	16,67k	40,58k
400	Inbound	4,65k	Over load
	Outbound	17,77k	Over load
500	Inbound	6,44k	Over load
	Outbound	18,89k	Over load
600	Inbound	8,94k	Over load
	Outbound	20,37k	Over load
700	Inbound	8,90k	Over load
	Outbound	22,54k	Over load
800	Inbound	9,26k	Over load
	Outbound	24,61k	Over load
900	Inbound	9,86k	Over load
	Outbound	25,48k	Over load
1000	Inbound	11,59k	Over load

	Outbound	29,68k	Over load

4. Hasil uji empat

Hasil uji empat dilakukan dengan melakukan benchmark *web server* yang menggunakan dan yang tidak menggunakan reverse proxy. Menggunakan parameter sebagai brikut:

Request : 1000

Concurention : 100-1000

Halaman : http://ujicoba.com/?m=201108

#ab -c 100 -n 1000 http://ujicoba.com/?m=201108

Tabel 4.10 Hasil uji empat physical memory.

Concurention	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
	Average (Bytes)	Average (Bytes)
100	578,45M	767,50M
200	532,67M	938,77M
300	565,61M	1,09G
400	589,78M	Over load
500	590,04M	Over load
600	596,34M	Over load
700	598,51M	Over load
800	611,34M	Over load
900	645,87M	Over load
1000	654,93M	Over load

Tabel 4.11 Hasil uji empat CPU Utilization.

Concurention	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
	Average (%)	Average (%)
100	11	37
200	13	45
300	13	50
400	15	Over load
500	15	Over load
600	18	Over load
700	20	Over load
800	20	Over load
900	21	Over load
1000	23	Over load

Tabel 4.12 Hasil uji empat Traffic

Concurention	Traffic	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
		Average (bits)	Average (bits)
100	Inbound	1,57k	3,85k
	Outbound	5,38k	16,86k
200	Inbound	2,76k	3,34k

	Outbound	11,35k	37,84k
300	Inbound	3,23k	4,97k
	Outbound	11,81k	40,52k
400	Inbound	4,7 7k	Over load
	Outbound	16,96k	Over load
500	Inbound	6,78k	Over load
	Outbound	17,88k	Over load
600	Inbound	8,08k	Over load
	Outbound	19,33k	Over load
700	Inbound	8,85k	Over load
	Outbound	20,32k	Over load
800	Inbound	9,28k	Over load
	Outbound	24,56k	Over load
900	Inbound	9,22k	Over load
	Outbound	25,59k	Over load
1000	Inbound	11,72k	Over load
	Outbound	28,91k	Over load

5. Hasil uji lima

Hasil uji lima dilakukan dengan melakukan benchmark *web server* yang menggunakan dan yang tidak menggunakan reverse proxy. Menggunakan parameter sebagai brikut:

Request : 1000

Concurention : 100-1000

Halaman : http://ujicoba.com/?cat=1

#ab -c 100 -n 1000 http://ujicoba.com/?cat=1

Tabel 4.13 Hasil uji lima physical memory.

Concurention	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
	Average (Bytes)	Average (Bytes)
100	536,44M	780,56M
200	563,74M	931,34M
300	580,16M	1001,52M
400	587,52M	Over load
500	588,66M	Over load
600	574,53M	Over load
700	598,99M	Over load
800	622,14M	Over load
900	645,53M	Over load
1000	675,68M	Over load

Tabel 4.14 Hasil uji lima CPU Utilization.

Concurention	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
	Average (%)	Average (%)
100	11	39
200	12	40
300	13	48

400	15	Over load
500	17	Over load
600	18	Over load
700	18	Over load
800	20	Over load
900	21	Over load
1000	23	Over load

Tabel 4.15 Hasil uji limaTraffic

Concurention	Traffic	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
		Average (bits)	Average (bits)
100	Inbound	1,18k	3,35k
	Outbound	6,78k	17,89k
200	Inbound	2,89k	3,99k
	Outbound	11,53k	37,79k
300	Inbound	3,91k	4,46k
	Outbound	11,95k	40,85k
400	Inbound	4,87k	Over load
	Outbound	16,59k	Over load
500	Inbound	6,44k	Over load
	Outbound	17,45k	Over load
600	Inbound	8,68k	Over load

	Outbound	19,88k	Over load
700	Inbound	8,97k	Over load
	Outbound	20,96k	Over load
800	Inbound	9,58k	Over load
	Outbound	24,98k	Over load
900	Inbound	9,81k	Over load
	Outbound	25,40k	Over load
1000	Inbound	11,67k	Over load
	Outbound	28,87k	Over load

6. Hasil uji enam

Hasil uji enam dilakukan dengan melakukan benchmark *web server* yang menggunakan dan yang tidak menggunakan reverse proxy. Menggunakan parameter sebagai brikut:

Request : 1000

Concurention : 100-1000

Halaman : http://ujicoba.com/?feed=rss2

#ab -c 100 -n 1000 http://ujicoba.com/?feed=rss2

Tabel 4.16 Hasil uji enam physical memory.

Concurention	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
	Average (Bytes)	Average (Bytes)
100	556,25M	758,31M
200	571,67M	989,24M

300	575,42M	1,14M
400	589,25M	Over load
500	590,78M	Over load
600	583,18M	Over load
700	595,47M	Over load
800	627,61M	Over load
900	637,56M	Over load
1000	658,82M	Over load

Tabel 4.17 Hasil uji enam CPU Utilization.

Concurention	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
	Average (%)	Average (%)
100	9	37
200	13	42
300	14	47
400	14	Over load
500	16	Over load
600	17	Over load
700	20	Over load
800	20	Over load
900	22	Over load
1000	23	Over load

Tabel 4.18 Hasil uji enam Traffic

Concurention	Traffic	Web server menggunakan reverse proxy	Web server tanpa menggunakan reverse proxy
		Average (bits)	Average (bits)
100	Inbound	1,89k	3,04k
	Outbound	6,21k	16,65k
200	Inbound	2,67k	3,78k
	Outbound	11,35k	37,78k
300	Inbound	3,83k	4,39k
	Outbound	11,55k	40,78k
400	Inbound	4,44k	Over load
	Outbound	16,59k	Over load
500	Inbound	6,58k	Over load
	Outbound	17,68k	Over load
600	Inbound	8,75k	Over load
	Outbound	19,92k	Over load
700	Inbound	8,68k	Over load
	Outbound	21,39k	Over load
800	Inbound	9,67k	Over load
	Outbound	23,98k	Over load
900	Inbound	9,92 k	Over load
	Outbound	25,57k	Over load
1000	Inbound	11,71k	Over load

	Outbound	28,85k	Over load

BAB V

PENUTUP

5.1 Kesimpulan

Dari hasil analisis dalam pengujian *reverse proxy server*, dapat ditarik kesimpulan beberapa hal:

1. *Web server* dengan *reverse proxy* dapat melayani lebih banyak permintaan dibandingkan *web server* tunggal.
2. *Web server* dengan *reverse proxy* dapat meminimalisir penggunaan resource *web server*
3. Menggunakan *reverse proxy* pada web server dengan *trafic* tinggi dapat menghindari *over load*.

5.2 Saran

Saran yang perlu diperhatikan dalam pengembangan suatu *reverse proxy server* antara lain :

1. Perlu diperhatikan penggunaan kebutuhan pemasangan *reverse proxy* pada *web site* dengan *trafic* rendah tidak terlalu berpengaruh besar.

DAFTAR PUSTAKA

Kadir, Abdul. 2002, Pengenalan Unix Dan Linux. Yogyakarta: Penerbit ANDI Yogyakarta.

2006, Modul Praktikum Jaringan Komputer. Yogyakarta : Penerbit Laboratorium Sistem dan Jaringan Komputer Teknik Informatika Universitas Islam Indonesia.

Wahana Komputer. 2009, Langkah Mudah Administrasi Jaringan Menggunakan Linux Ubuntu 9. Yogyakarta: Penerbit ANDI Yogyakarta. Semarang: Penerbit WAHANA KOMPUTER

Rafiudin, Rahmat. 2008, Squid Koneksi Anti Mogok. Yogyakarta: Penertbit ANDI Yogyakarta.

Mahbub,Cecep,2009, Benchmark Performa Web Server dengan ApacheBench (ab), http://ngadimin.com/2009/09/27/benchmark-performa-web-server-dengan-apachebench-ab/, di akses tanggal 27 maret 2010.

Wahyu,http://wahyuimalone.wordpress.com/2009/11/12/pengertian-mail-dhcp-web-dns-ftp-dan-proxy-server/, di akses tanggal 28 maret 2010.

Anonim1,http://metalova.web.id/2009/06/30/sejarah-unix-dan-linux/, di akses tanggal 28 maret 2010 .

Anonim 3

http://id.wikipedia.org/wiki/Network_address_translation, di akses tanggal 28 maret 2010.