IMPLEMENTASI *LOAD BALANCING* DENGAN PENDEKATAN *ROUND TRIP TIME* DAN *CPU USAGE* PADA LAYANAN HTTP

SKRIPSI

Untuk memenuhi sebagian persyaratan untuk mencapai gelar
Sarjana Komputer

Disusun Oleh:
NUR CAHYO NUGROHO
NIM. 0810680049

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS BRAWIJAYA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
MALANG
2013

LEMBAR PERSETUJUAN

IMPLEMENTASI *LOAD BALANCING* DENGAN PENDEKATAN *ROUND TRIP TIME* DAN *CPU USAGE* PADA LAYANAN HTTP

SKRIPSI

KONSENTRASI KOMPUTASI BERBASIS JARINGAN

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Ilmu Komputer

Disusun Oleh :
Nur Cahyo Nugroho
NIM. 0810680049

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Heru Nurwasito, M.Kom	**Sabriansyah R.A., S.T., M.Eng**
NIP. 196504021990021001	**NIP. 197508191999031001**

LEMBAR PENGESAHAN

IMPLEMENTASI *LOAD BALANCING* DENGAN PENDEKATAN *ROUND TRIP TIME* DAN *CPU USAGE* PADA LAYANAN HTTP

SKRIPSI

KONSENTRASI KOMPUTASI BERBASIS JARINGAN

Diajukan untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer

Disusun Oleh :

Nur Cahyo Nugroho
NIM. 0810680049

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 19 Juli 2013

Penguji I	Penguji II
Achmad Basuki, ST.,M.MG.,Ph.D	**Aswin Suharsono, ST., MT**
NIP. 197411182003121002	**NIK. 84091906110251**

Penguji III

Barlian Henryanu P., ST., MT
NIK. 82102406110254

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, Juli 2013
Mahasiswa,

Nur Cahyo Nugroho
0810680049

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Implementasi *Load Balancing* dengan Pendekatan *Round Trip Time* dan *CPU Usage* pada Layanan HTTP" dengan baik. Penyusunan skripsi ini dapat terlaksana dengan baik karena adanya bantuan secara langsung maupun tidak langsung dari pihak tertentu, diantaranya :

1. Bapak **Ir. Heru Nurwasito, M. Kom.** selaku dosen pembimbing I yang telah memberikan ilmu dan saran untuk skripsi ini.
2. Bapak **Sabriansyah R. Akbar, ST., M.Eng.** selaku dosen pembimbing II yang telah memberikan ilmu dan saran untuk skripsi ini.
3. **Orang tua,** yang telah memberikan dukungan moral dan material.
4. Aniisya Rachim yang selalu memberikan dukungan dan perhatiannya selama ini.
5. Teman teman yang telah membantu memberi saran dan kritik atas skripsi ini.

Penulis menyadari bahwa skripsi ini masih banyak kekurangan dan jauh dari sempurna, karena keterbatasan materi dan pengetahuan yang dimiliki penulis. Akhirnya semoga skripsi ini dapat bermanfaat dan berguna bagi pembaca terutama mahasiswa Teknik Informatika Universitas Brawijaya.

Malang, Juni 2013

Penulis

ABSTRAK

Nur Cahyo Nugroho. 2013. Implementasi *Load Balancing* dengan Pendekatan *Round Trip Time* dan *CPU Usage* pada Layanan HTTP. Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing: Ir. Heru Nurwasito, M. Kom. dan Sabriansyah R.A., S.T., M.Eng.

Situs-situs populer yang kaya akan konten memiliki pengunjung yang sangat banyak dalam hitungan detik. Dengan tujuan untuk menghindari *overload* pada sebuah *server* yang menangani seluruh layanan yang ada, salah satu metode yang digunakan adalah dengan membangun *server cluster*. Pembagian beban kepada *server* didalam *server cluster* secara adil salah satunya dilakukan dengan teknik *load balancing*. Salah satu faktor yang mempengaruhi beban komputasi dalam memproses halaman web adalah konten didalamnya. Dalam penelitian ini, penulis mengimplementasikan teknik *load balancing* dengan mendistribusikan beban konten karena pada sebuah halaman web yang memerlukan kemampuan proses yang lebih tinggi adalah kontennya. Penulis melakukan pendekatan *round trip time(rtt)* dan *CPU usage* dalam penerapan teknik *load balancing* sebagai metrik untuk menentukan *server* tujuan. Penelitian yang dilakukan akan membandingkan sistem *load balancing* yang dibangun dengan kemampuan sebuah *server* untuk melayani layanan yang sama. Hasil yang diperoleh menunjukkan *CPU usage* pada *origin server* dengan sistem *load balancing* yang dilakukan mengalami penurunan karena beban pemrosesan konten ditanggung oleh *content server*, *origin server* hanya menanggung beban pemrosesan teks pada halaman web dengan demikian *origin server* dapat melayani permintaan dalm jumlah yang lebih besar.

Kata Kunci: Load Balancing, Cluster, Layanan HTTP, Load Balancing Adaptif

ABSTRACT

Nur Cahyo Nugroho. 2013. Implementasi *Load Balancing* dengan Pendekatan *Round Trip Time* dan *CPU Usage* pada Layanan HTTP. *Information Technology and Computer Science Program, Brawijaya University,* Malang. *Advisor*: Ir. Heru Nurwasito, M. Kom. *and* Sabriansyah R.A., S.T., M.Eng.

Popular rich content website have a huge number of visitor in a short period of time. Server cluster architecture is one of several methods to avoid an "overload" state in single server caused by a large number of request. Load balancing technique can be implemented in server cluster to split load request fairly among servers in the cluster. One of several factor affecting the load of computing resource in handling HTTP request is content processing. In this paper, we propose a load balancing technique with content distribution based on round trip time(rtt) and CPU usage. We compared our load balancing system with single server system to serve HTTP service. The result showed that our system can get lower CPU usage and response time with better throughput than the single server system. Our system processes the text object in HTTP request while the contents are served by another servers. With lower resource to process the requests, server can handle more request than the single server system.

Keywords: Load Balancing, Cluster, HTTP service, Adaptive Load Balancing

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Seiring dengan pertumbuhan teknologi internet yang sangat cepat, pertumbuhan konten-konten juga sangat melimpah dengan berbagai macam jenis dan ukuran yang bervariasi pula. Beberapa situs yang memiliki konten-konten yang menarik akan terbanjiri pengunjung yang sangat banyak dalam waktu yang sangat singkat. *Server* membutuhkan kemampuan proses yang sangat cepat untuk melayani seluruh permintaan pengunjung dengan jumlah yang banyak dan dalam waktu singkat. Hal ini akan sulit dilakukan apabila seluruh konten pada situs tersebut hanya ditampung pada satu buah *server* mengingat konten-konten yang ada saat ini memiliki ukuran yang tidak kecil.

Kemampuan proses sebuah *server* sangat terbatas untuk melayani seluruh permintaan pengunjung, dan kemampuan proses tersebut dapat menurun apabila beban komputasi pada *server* tersebut sudah sangat tinggi yang dapat menyebabkan pengiriman konten kepada pengunjung menjadi lebih lambat, sedangkan Pengunjung tidak menginginkan waktu yang lama saat mengakses situs tersebut. Dari penjelasan di atas, sangat sulit untuk memenuhi keinginan pengunjung dengan bertumpu pada kemampuan satu buah *server*. Untuk menangani hal ini, salah satu metode yang diajukan adalah menambah jumlah *server* yang disebut *web-server cluster* [DTO-97]. Untuk menangani layanan dengan sejumlah *server* dibutuhkan strategi pembagian permintaan user yang seimbang ke beberapa *server* tersebut. Beberapa metode pembagi beban yang ada antara lain pembagi beban dengan pendekatan DNS [DTO-97] [HEO-00] dan metode yang dikemas dalam aplikasi *Linux Virtual Service* [ZHA-03] dengan beberapa algoritma penjadwalan didalamnya.

Pada sebuah halaman web, konten yang terdapat di dalamnya yang membuat halaman web tersebut menjadi berat. Jika dalam halaman web

yang lebih mengutamakan tulisan dan hanya memiliki sedikit konten, *server* tidak akan terlalu terbebani untuk memproses permintaan pengunjung terhadap halaman web tersebut walaupun dengan jumlah pengunjung yang besar. Semakin banyak konten didalamnya, semakin tinggi pula beban yang dialami oleh *server* untuk memproses halaman web tersebut kepada pengunjung. Dalam penelitian ini penulis mengangkat permasalahan kemampuan sebuah *server* untuk melayani permintaan pengunjung terhadap halaman web yang memiliki banyak konten. Permasalahan yang dimaksud penulis terkait bagaimana *server* dapat mengurangi sumber daya komputasi yang diakibatkan permintaan pengunjung dalam jumlah besar terhadap halaman web pada *server* tersebut. Penulis memfokuskan penelitian ini pada pembagian beban dengan cara distribusi konten pada halaman web.

Pada penelitian ini penulis akan membangun sebuah sistem *load balancing* adaptif dengan melakukan pengukuran beberapa nilai metrik untuk menentukan tujuan *content server*. Beberapa metrik yang dapat digunakan untuk menentukan keputusan pada sistem *load balancing* antara lain: *CPU usage, memory usage, disk usage, process number* saat ini dan waktu *ICMP request-reply* [KON-07]. Pada penelitian ini penulis hanya memfokuskan metrik *CPU usage* dan waktu *ICMP request-reply* terhadap user. *Content server* yang dimaksud disini adalah *server* tempat menyimpan konten-konten web. Konten-konten yang berada terdapat pada suatu halaman web akan disebar ke beberapa *content server*. Teks yang berada pada halaman web akan dilayani oleh *origin server*. Untuk dapat melakukan pemilihan *content server* dengan tepat, sistem akan melakukan pengurutan *server* berdasarkan hasil pengukuran metrik diatas. *Content server* yang akan menjadi tujuan merupakan *server* yang menempati urutan tertinggi. Lokasi konten dalam halaman web akan berubah secara dinamis sesuai dengan hasil pengukuran pada *load balancer*. Lokasi konten pada halaman web yang dimaksud adalah *domain name* dari *content server* tujuan.

1.2. Rumusan Masalah

Berdasarkan permasalahan yang telah dijelaskan pada latar belakang, maka rumusan masalah dapat disusun sebagai berikut:

1. Menerapkan pengukuran metrik waktu *ICMP request-reply* terhadap user dan *CPU usage* pada *content server* dalam sistem *load balancing* agar sistem dapat melakukan pemilihan *server* pada layanan *HTTP*.
2. Membangun sistem *load balancing* yang dapat mengurangi beban komputasi yang dialami pada *origin server* dengan sistem *single server*.

1.3. Batasan Masalah

Agar permasalahan yang dirumuskan lebih terfokus, maka penelitian ini dibatasi oleh hal-hal sebagai berikut:

1. Seluruh komputer *server* fisik yang digunakan memiliki spesifikasi yang sama.
2. Pengujian sistem hanya pada protokol layanan berbasis web.
3. Diimplementasikan pada pengalamatan jaringan pada IPv4.
4. Sistem yang dibangun berada di dalam segmen jaringan yang sama.
5. Implementasi program menggunakan *PHP* dan *bash shell.*
6. Penentuan batas overload *CPU usage* dilakukan secara statik yaitu 80%.
7. Halaman web yang digunakan dalam pengujian merupakan halaman web statis.

1.4. Tujuan

Berdasarkan rumusan masalah yang ditulis diatas, tujuan dari penyelenggaraan proyek akhir ini adalah untuk memberikan mekanisme alternatif sistem *load balancing* dan berhasil membagi beban kepada *web-server* berdasarkan hasil pengukuran metrik waktu *ICMP request-reply* terhadap user dan *CPU usage* pada *content server* dengan tepat serta dapat mengurangi beban komputasi yang dialami oleh sebuah *server*.

1.5. Manfaat

Penelitian ini diharapkan dapat bermanfaat untuk berbagai pihak. Manfaat dari penelitian ini adalah sebagai berikut:

- Bagi penulis

- Mengaplikasikan ilmu yang diperoleh selama mengikuti perkuliahan di Teknik Informatika Universitas Brawijaya.
- Mendapatkan pengetahuan mengenai sistem *load balancing*.

- Bagi pengguna
  - Memberikan solusi alternatif sistem *load balancing*.
  - Menjadi bahan referensi penelitian yang berkaitan dengan sistem *load balancing*.

1.6. Sistematika Pembahasan

Sistematika pembahasan dari penyusunan proyek akhir ini direncanakan sebagai berikut :

BAB I PENDAHULUAN

Pendahuluan terdiri dari latar belakang, identifikasi dan pembatasan masalah, rumusan masalah, tujuan dan manfaat serta sistematika pembahasan dari proyek akhir ini.

BAB II DASAR TEORI

Bab II membahas teori-teori yang berkaitan dan menunjang dalam penyelesaian penelitian ini. Teori-teori yang diambil berasal dari jurnal, buku, dan sumber referensi lainnya yang berhubungan dengan topik yang akan diteliti.

BAB III METODOLOGI PENELITIAN DAN PERANCANGAN

Langkah-langkah dalam membangun sistem virtualisasi *server* dijelaskan pada Bab III. Penjelasan langkah-langkah seperti perancangan, implementasi, pengujian, dan analisis sistem dibahas secara umum dengan menampilkan diagram alir proses. Selain itu pada BAB III juga dijelaskan mengenai perancangan sistem *load balancing* yang akan dibangun secara keseluruhan seperti perangkat keras, perangkat lunak, topologi sistem yang akan digunakan dalam implementasi sistem.

BAB IV IMPLEMENTASI

Pada Bab IV, akan dijelaskan pembuatan sistem *load balancing* secara terperinci dengan menampilkan gambar-gambar dari hasil implementasi yang telah dilakukan.

BAB V PENGUJIAN DAN ANALISIS

Pada Bab V, berisi pengujian dan analisis terhadap sistem *load balan-cing* yang dibangun. Pengujian tersebut dilakukan bertahap sesuai dengan skenario pengujian yang ditentukan pada Bab III Metodologi Penelitian.

BAB VII PENUTUP

Kesimpulan dari pelaksanaan penelitian ini yang dibuat berdasarkan hasil pengujian dan analisis terhadap sistem yang dibangun dan dirangkum pada bab penutup. Untuk meningkatkan hasil kinerja dari sistem yang dibangun, maka diberikan saran-saran yang dapat digunakan untuk penyempurnaan sistem untuk penelitian selanjutnya.

BAB II
KAJIAN PUSTAKA

2.1. Penelitian Terkait

Penelitian terkait mengenai *load balancing* yang salah satunya dilakukan oleh S. Kontogiannis dkk [KON-07] dengan judul *"An Adaptive Load Balancing Algorithm for Cluster-Based Web Systems"*. Pada penelitian ini dilakukan implementasi algoritma *load balancing* dengan melakukan pengukuran metrik *HTTP response time* dan *network delay* untuk menentukan *weight* dari *mirror server* yang dilakukan pada *load balancer*. *Weight* awal diberikan oleh administrator untuk mengetahui estimasi awal sistem *load balancing* serta mengetahui kondisi *web-server*. Setelah itu algoritma penjadwalan *weighted round robin (wrr)* dijalankan untuk mendapatkan metrik *HTTP response time* dan *network delay* dari masing-masing *server* hingga proses perhitungan metrik selesai pada periode tersebut. Penelitian ini membandingkan algoritma adaptif yang dilakukan dengan algoritma *"blind selection"* seperti *round robin (rr)* dan *least connection(lc)*. Dari hasil pengujian yang dilakukan beberapa skenario menghasilkan sistem yang dibangun dapat menandingi bahkan mengungguli kinerja algoritma *rr* dan *lc*. Secara khusus algoritma yang dilakukan membagi *traffic HTTP* secara efisien pada beberapa kondisi tidak stabil yang berubah secara dinamis seperti: karena penggunaan sumber daya pada jaringan dan karena keterbatasan suber daya komputasi pada *web-server*, sementara sumber daya jaringan memadai.

Penelitian terkait lainnya mengenai *load balancing* dilakukan oleh Tzung-Shi Chen Kuo-Lian Chen [TZU-04] dengan judul *"Balancing Workload Based on Content Types for Scalable Web Server Clusters"*. Pada penelitian ini pembagian beban dilakukan berdasarkan tipe konten yang diakses. Masing-masing konten

memiliki jenis dan ukuran yang bervariasi, dan sistem *load balancing* akan menentukan pengunjung akan diarahkan ke *server* yang tepat sesuai dengan perhitungan *cost* dari masing-masing *server* ditinjau dari konten yang diakses. Perhitungan *cost* masing-masing *server* diperoleh dari beberapa metrik, antara lain: rata-rata *throughput* pada masing-masing *server* yang diperoleh pada periode sebelumnya, jumlah permintaan yang sedang berlangsung pada *server* saat ini, tingkat keutamaan konten yang diakses dan *overhead startup* pada *server*. Pengujian yang dilakukan membandingkan algoritma yang diterapkan dengan algoritma penjadwalan yang sudah ada seperti *rr*, *lc* dan *wrr*. Tujuan penelitian ini yaitu untuk memperoleh waktu *request delay* yang rendah dari berbagai macam konten yang diakses. Hasil pengujian yang dilakukan menghasilkan pendekatan algoritma yang dilakukan memiliki rata-rata waktu *delay* paling sedikit ketika beban kerja permintaan semakin berat. Hal ini menunjukkan pendekatan yang dilakukan sesuai dengan waktu pemrosesan yang bervariasi. Bahkan ketika pada beban kerja yang ringan pendekatan yang dilakukan masih memiliki karakteristik yang dimiliki sebuah *load balancer*.

Penelitian terkait mengenai pemilihan *server* tujuan pada Youtube yang dilakukan oleh Vijay dan kawan-kawan [VSZ-11]. Penelitian ini memeriksa arsitektur distribusi video pada Youtube, bagaimana *server* video dapat melakukan pemilihan ketika user mencoba untuk menonton video di Youtube. Disamping menggunakan *static hash-based load distribution* dan *DNS based dynamic mapping*, Youtube menggunakan pendekatan yang lebih dinamis menggunakan *HTTP redirection* dimana server yang berfungsi untuk melayani video mengirimkan header *HTTP 302* kepada user untuk meminta user mengambil video dari *server* yang lain. Analisis yang dilakukan menunjukkan pengelompokan *redirection* yaitu *intra-tier redirection* dan *inter-tier redirection*. Pada *intra-tier redirection* dua buah alamat IP *server* (yang melakukan *redirect* dan yang menerima dan melayani permintaan yang di-*redirect*) berada dalam satu lokasi (kota). Pada *intra-tier redirection* menunjukkan bahwa Youtube mencoba me-*redirect* user menuju *server* fisik yang lain dalam satu lokasi yang dapat

disebabkan apabila *server* yang dituju sedang sibuk sehingga akan di-*redirect* ke *server* yang kurang sibuk. Pada kasus *inter-tier redirection*, user di-*redirect* menuju hirarki *cache* yang lebih tinggi. Sebagaimana lokasi *cache* sekunder dan tersier yang semakin sedikit, biasanya user akan di-*redirect* menuju *server* yang berlokasi jauh. Penelitian ini menemukan ide kunci dibalik mekanisme *redirection* pada level aplikasi yang dilakukan oleh Youtube dan bagaimana hal itu dapat berdampak pada kinerja yang diobservasi oleh user.

2.2. *Load Balancing*

Berdasarkan penelitian-penelitian yang telah dilakukan, teknik *load balancing* pada umumnya dilakukan untuk menangani permasalahan beban kerja yang dialami oleh sebuah *server*. Seluruh layanan yang dimiliki seluruhnya ditangani hanya dengan sebuah *server* utama. Teknik *load balancing* dilakukan dengan cara membagi beban kerja ke beberapa *mirror server* yang memiliki layanan yang sama dengan *server* utama, sehingga memungkinkan *server* utama melayani jumlah permintaan yang lebih banyak karena berkurangnya penggunaan sumber daya yang dimiliki oleh *server* utama [BOU-01]. Ada beberapa algoritma penjadwalan pada teknik *load balancing* seperti *round robin, weighted round robin, least connection, weighted least connection* [ZHA-03] dan lain-lain.

S. Kontogiannis dalam penelitiannya [KON-07] mengelompokkan algoritma-algoritma penjadwalan yang sudah ada ke dalam empat kategori, antara lain: *Stateless non adaptive, State full non adaptive*, *Stateless adaptive* dan *State full adaptive* dimana *state full-stateless* adalah algoritma penjadwalan yang melacak atau tidak pada permintaan koneksi user. Sedangkan *adaptive-non adaptive* adalah algoritma penjadwalan yang melakukan pengecekan status *web-server* atau tidak melalui beberapa metrik.

Pada kategori *stateless non adaptive* tidak memperhatikan koneksi user maupun kondisi dari *server*. Beberapa algoritma yang termasuk ke dalam kategori ini adalah *random* dan *round robin.* Untuk kategori *state full non adaptive* melakukan pelacakan terhadap permintaan koneksi user. Algoritma yang termasuk

ke dalam kategori ini adalah *wrr, lc, wlc, destination hashing* dan *Shortest Expected Delay(SED).* Pada kategori *stateless adaptive* melakukan pengecekan terhadap kondisi *server* tetapi tidak melakukan pelacakan terhadap permintaan koneksi user. Informasi keadaan *server* diterima oleh *web switch/load balancer* yang kemudian digunakan untuk menentukan *server* tujuan. Informasi-informasi yang biasanya digunakan untuk pengukuran metrik antara lain: *cpu usage, memory usage, disk usage, current process number* dan waktu *ICMP request-reply.* Biasanya metrik-metrik ini disimpan pada layanan monitoring seperti *SNMP*. Algoritma yang termasuk ke dalam kategori ini adalah *fastest response time* yang hanya menggunakan metrik waktu *ICMP request-reply.* Untuk kategori *statefull adaptive* melakukan pelacakan terhadap permintaan koneksi user dan melihat kondisi server. Algoritma yang termasuk ke dalam kategori ini adalah *least loaded* yang menggunakan algoritma *wrr* adaptif dengan melakukan pengecekan kondisi *server* menggunakan *SNMP*.

2.3. *Web Server Cluster*

Cluster dapat diartikan sebagai sebuah kelompok *server* independen yang terhubung dengan jaringan yang dapat menggunakan sumber daya dari masing-masing *server* untuk digunakan secara bersama. *Cluster* di dalam *web-server* merupakan gabungan beberapa *web-server* menjadi sebuah sistem tunggal untuk mengatasi peningkatan beban kerja secara bersama dan menyediakan redundansi sistem. Permintaan user ke dalam sistem *cluster* akan diarahkan secara dinamis menuju *server* yang tepat berdasarkan metrik yang ditentukan dari sistem tersebut [HAD-04].

2.4. Distribusi Konten

Provider *Content Delivery Network (CDN)* seperti Akamai melakukan pendistribusian konten menuju *edge server* agar lebih dekat dengan user. Dengan mendistribusikan konten menuju *edge server* dapat memberikan layanan yang lebih cepat dan mengurangi tuntutan infrastruktur dari suatu situs [JOH-02].

Beberapa teknik distribusi konten diantaranya adalah: *unicast, cache-based, peer-to-peer* dan *multicast* [ANO-08].

2.5. Parameter Pengukuran

a. *CPU Utilization*

Nilai yang ditunjukkan oleh aplikasi pengukur *CPU Utilization* menunjukkan prosesor sedang melakukan eksekusi instruksi. Semakin tinggi *CPU Utilization* yang ditunjukkan semakin besar kemungkinan pekerjaan harus menunggu prosesor tersebut dapat tersedia [GER-09]. Pada parameter yang digunakan untuk pengukuran metris sistem *load balancing* yang akan dibangun melakukan penjumlahan seluruh penggunaan sumber daya *CPU* yang sedang terjadi.

```
Cpu(s): 87.3%us,  1.2%sy,  0.0%ni, 27.6%id,  0.0%wa,  0.0%hi,
0.0%si,  0.0%st

us: CPU yang digunakan user
sy: CPU yang digunakan kernel
ni: CPU yang digunakan oleh proses prioritas rendah
id: CPU idle
wa: CPU yang digunakan proses waiting (dalam disk)
hi: CPU yang digunakan untuk menangani interrupt hardware
si: CPU yang digunakan untuk menangani interrupt software
st: CPU yang digunakan virtual machine
```

Penjumlahan yang dilakukan untuk memperoleh total *CPU usage* adalah dengan menjumlah komponen *us, sy, ni, wa, hi, si,* dan *st.*

b. *Round Trip Time*

TCP mencatat waktu ketika masing-masing segmen melakukan pengiriman dan waktu ketika suatu *acknowledgement* suatu data diperoleh pada segmen tersebut. Dari kedua waktu tersebut *TCP* menghitung selisih waktu yang terjadi yang dikenal dengan *round trip time* [DOU-95].

2.6. Apache jMeter

JMeter merupakan aplikasi desktop yang di desain untuk melakukan pengukuran dan tes kinerja fungsional dari aplikasi *client*/*server*, seperti aplikasi

FTP dan lain-lain. JMeter bekerja dari sisi *client* dari sebuah aplikasi *client/server*. JMeter melakukan pengukuran *response time* dan keseluruhan sumber daya *server* seperti *CPU load*, *memory usage*, dan *resource usage* [EMI-08].

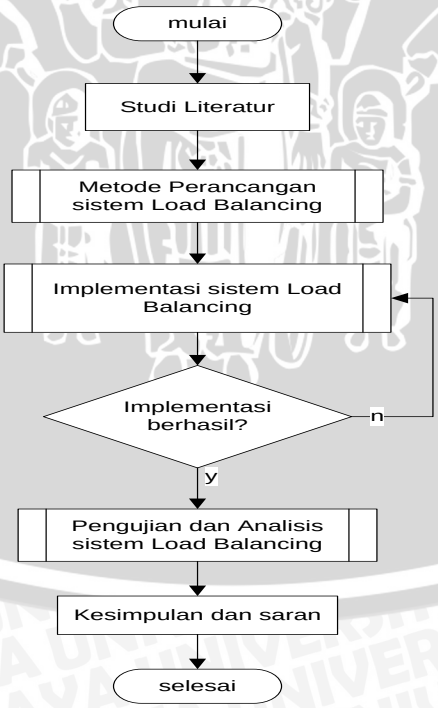
JMeter dapat digunakan untuk menguji kinerja dari sebuah sumber daya statis dan dinamis seperti berkas statis, *servlets*, *server FTP*, objek *java*, *database*, skrip *perl/CGI*, *query*, dan lain-lain. Untuk menguji pengukuran dari sebuah *server HTTP* atau *server FTP*, jaringan, jMeter membutuhkan untuk menyediakan simulasi dari beberapa beban yang berbeda dari sistem ini. JMeter dapat melakukan analisis kinerja yang lebih baik dibawah beban kerja yang berat sekalipun[EMI-08].

BAB III
METODOLOGI PENELITIAN DAN PERANCANGAN

Bab ini terbagi menjadi dua buah sub-bab yaitu metodologi penelitian dan perancangan. Pada sub-bab metodologi penelitian akan membahas langkah-langkah yang akan dilakukan dalam penelitian. Pada sub-bab perancangan akan membahas rancangan sistem yang akan dibangun.

3.1. Metodologi Penelitian

Bagian ini menjelaskan mengenai langkah-langkah yang diambil oleh penulis dalam melaksanakan penelitian. Urutan langkah-langkah tersebut adalah studi literatur, perancangan sistem, implementasi sistem, pengujian sistem, analisis dan pengambilan kesimpulan. Diagram alir keseluruhan pelaksanaan penelitian dapat dilihat pada gambar 3.1.



Gambar 3.1. Diagram alir keseluruhan pelaksanaan penelitian

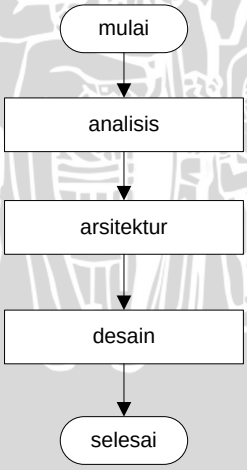
3.1.1. Studi Literatur

Studi literatur yang dilakukan adalah mengenai karakteristik, parameter, serta teori pendukung lain yang menunjang dalam penulisan skripsi ini. Teori-teori pendukung tersebut meliputi:

a. *Load Balancing*
b. Penelitian terkait
c. Pemrograman *PHP*
d. Pemrograman soket
e. *Web-server*
   - Apache *Web-server*

3.1.2. Perancangan Sistem *Load Balancing*

Tahapan perancangan dilakukan untuk dapat membangun sistem *load balancing* yang sesuai dengan tujuan penelitian. Perancangan dilakukan untuk mempermudah saat proses implementasi dilakukan. Diagram alir perancangan sistem *load balancing* dapat dilihat pada gambar 3.2.

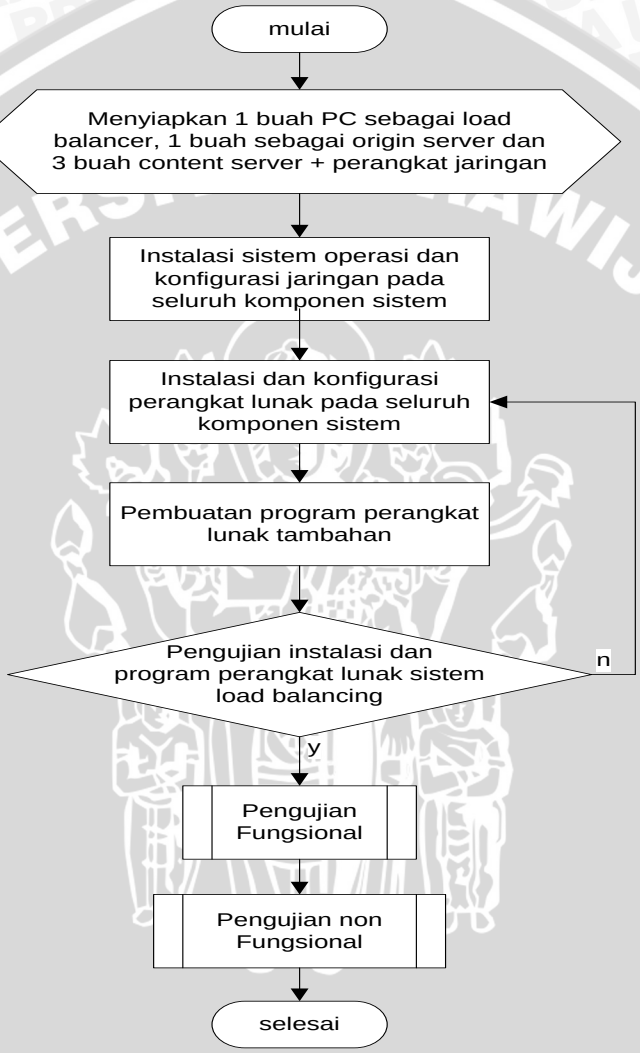


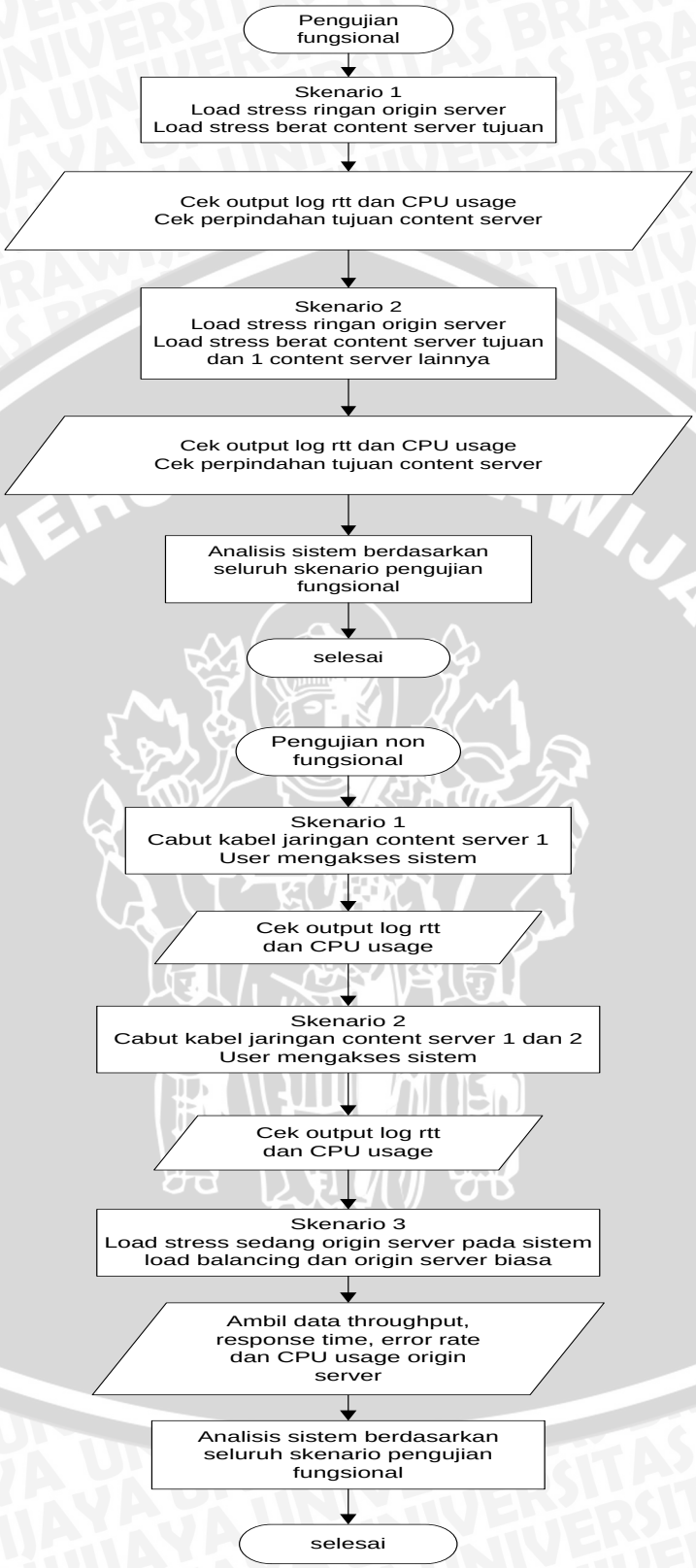
Gambar 3.2. Diagram alir perancangan sistem

3.1.3. Implementasi dan Pengujian Sistem Load Balancing

Proses implementasi dilakukan berdasarkan pada tahapan perancangan yang telah ditentukan. Pada bagian ini, proses dikerjakan bertahap agar

memudahkan penulis dalam mengidentifikasi kesalahan dalam pembuatan sistem *load balancing*. Tahapan pengujian dilakukan dengan tujuan untuk memastikan apakah sistem *load balancing* yang dibangun dapat bekerja secara fungsional maupun non-fungsional. Diagram alir implementasi dan pengujian sistem *load balancing* dapat dilihat pada gambar 3.2.





Gambar 3.3. Diagram alir implementasi dan pengujian sistem

3.1.4. Analisis dan Pembahasan

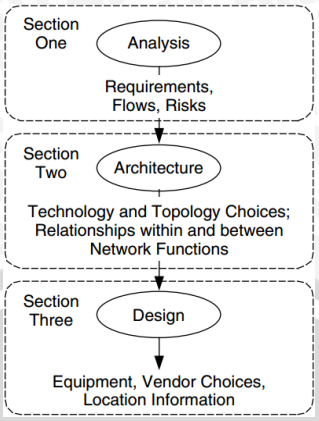
Analisis yang dilakukan adalah apakah sistem *load balancing* yang dibangun memenuhi kebutuhan yang diinginkan dan sesuai dengan tujuan penelitian yang dilakukan. Pada tahapan ini akan dilakukan pembahasan mengenai hasil pengujian sistem *load balancing* yang dilakukan sebelumnya. hasil dari analisis yang dilakukan kemudian digunakan untuk pengambilan kesimpulan.

3.1.5. Pengambilan Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi, pengujian, dan analisis sistem *load balancing* telah selesai dilakukan. Kesimpulan disusun berdasarkan data hasil pengujian dan analisis dari sistem *load balancing* yang telah dibangun. Pada tahap akhir penulisan dimuat saran yang bertujuan memperbaiki dan memberikan beberapa pertimbangan pada sistem yang dibangun untuk pengembangan pada penelitian selanjutnya.

3.2. Perancangan

Bagian perancangan menjelaskan tentang rancangan sistem *load balancing* yang akan dibangun. Perancangan dilakukan agar sistem *load balancing* yang dibangun dapat memenuhi tujuan penelitian yang telah dijelaskan sebelumnya. Untuk dapat membangun sistem yang sesuai dengan kebutuhan user, perlu dilakukan tiga tahapan proses yaitu analisis jaringan, arsitektur, dan desain. Melalui proses-proses ini dapat ditemukan masalah yang akan diatasi pada sistem yang akan dibangun, dapat menentukan layanan dan kinerja yang dibutuhkan untuk mengatasi masalah tersebut dan dapat mendesain sistem yang menyediakan layanan serta kinerja yang dibutuhkan [JAM-07]. Alur hubungan antara analisis, arsitektur, dan desain ditunjukkan pada gambar 3.4. Melalui ketiga proses tersebut akan dapat diketahui masalah yang perlu diselesaikan dengan sistem tersebut, menentukan layanan apa saja yang dapat mengatasi masalah tersebut, serta arsitektur dan desain sistem yang akan dibangun sesuai dengan layanan yang dibutuhkan.



Gambar 3.4. Alur hubungan perancangan sistem [JAM-07]

3.2.1. Analisis

Pada tahap analisis dilakukan identifikasi kebutuhan-kebutuhan yang dibutuhkan oleh sistem *load balancing* yang akan dibangun, sehingga dapat diketahui desain sistem yang sesuai untuk diterapkan pada saat tahapan implementasi dilakukan. Sistem *load balancing* yang dibangun merujuk pada tujuan penelitian yaitu dapat membagi beban pada *origin server* dengan cara menyebar konten pada halaman web ke beberapa *content server*. Pembagian beban dilakukan berdasarkan perhitungan *rtt* antara user dan *content server* dan *CPU usage* pada *content server* yang terkecil.

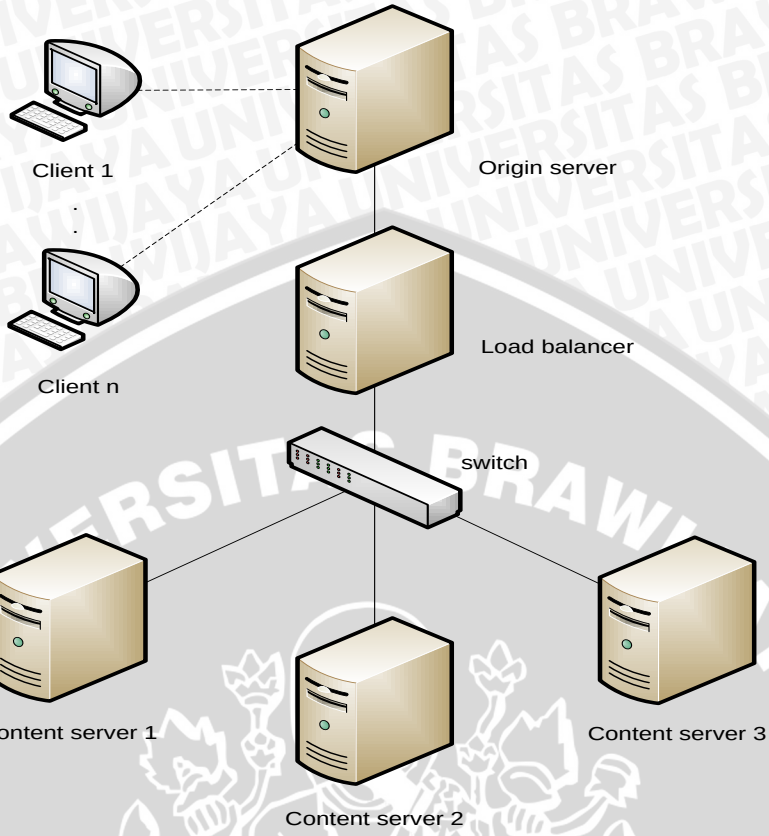
Untuk meneruskan alamat IP user yang didapat oleh *origin server* menuju *content server*, sistem membutuhkan media pengiriman pesan antara *origin server*, *load balancer* dan *content server*. Pengecekan kondisi *CPU usage* pada *content server* dibutuhkan oleh sistem untuk selalu mengetahui apabila *content server* mengalami *overload CPU usage*. User tidak menginginkan waktu proses yang lama saat mengakses layanan pada sistem. Proses penyimpanan alamat IP user dibutuhkan untuk melewati proses pengukuran *rtt* dan *CPU usage* agar user tidak perlu menunggu lebih lama dalam setiap mengakses layanan dalam sistem *load balancing* yang dibuat. Modifikasi sumber konten pada halaman web dibutuhkan untuk mengganti tujuan konten sesuai dari hasil pengukuran sistem. Sistem *load balancing* yang dibangun harus dapat tetap melayani sistem walaupun

tidak semua *content server* dalam keadaan hidup. Dari penjelasan diatas, dapat dilakukan perincian kebutuhan sistem *load balancing* yang akan dibangun:

- Antar komponen sistem *load balancing* dapat saling bertukar informasi.
- *Content server* dapat memberitahu keadaan *CPU usage* kepada *load balancer* untuk dilakukan tindakan lebih lanjut.
- Penyimpanan alamat IP user dibutuhkan untuk menghindari proses pengukuran *rtt* dan *CPU usage* setiap kali mengakses layanan pada sistem.
- Sistem dapat memodifikasi halaman web untuk mengubah sumber konten sesuai hasil pengukuran sistem.
- Sistem tetap dapat melayani user apabila salah satu atau beberapa *content server* dalam keadaan mati.

3.2.2. Arsitektur

Pada tahapan arsitektur yang dilakukan adalah menetapkan teknologi yang akan diterapkan serta merancang topologi jaringan yang akan digunakan oleh sistem *load balancing.* Perancangan jaringan yang dilakukan meliputi instalasi *origin server, content server,* dan *load balancer*. Hal berikutnya yang dilakukan yaitu mengkonfigurasikan setiap bagian tersebut berada dalam segmen jaringan yang sama yang terhubung melalui media penghubung. Alamat jaringan yang ditentukan untuk sistem *load balancing* yang dibangun yaitu 10.10.10.0/24. Perancangan jaringan sistem *load balancing* yang akan dibangun dapat dilihat pada gambar 3.5.



Gambar 3.5. Perancangan jaringan sistem *load balancing*

Untuk perancangan perangkat lunak menjelaskan mengenai perangkat lunak apa saja yang akan digunakan untuk mendukung sistem yang akan dibangun. Perangkat lunak yang digunakan mengacu pada tahap analisis yang telah dijelaskan sebelumnya. Berikut adalah rincian dan kegunaan perangkat lunak yang mendukung sistem *load balancing* yang sesuai dengan tahapan analisis kebutuhan.

- ***PHP* –** Sebagian besar sistem menggunakan perangkat lunak ini. Penggunaan perangkat lunak ini antara lain komunikasi soket serta perbandingan *rtt* dan *CPU usage* dari masing-masing *content server*.
- **DNS *(Domain Name System) server* –** Digunakan untuk mencari alamat IP dari *domain name* pada *content server* sebagai tujuan user.

- ***Load tester*** **–** Digunakan untuk melakukan pengujian. Perangkat lunak ini digunakan untuk memberi request dalam jumlah yang besar.
- ***Web-server*** **–** Perangkat lunak ini diinstal di masing-masing komponen sistem yang berfungsi sebagai media pengujian.
- ***DBMS*** **–** DBMS yang digunakan penulis yaitu *MySQL*. Perangkat lunak ini digunakan untuk menyimpan data-data mengenai alamat IP user, *rtt, CPU usage,* dan alamat IP *content server.* Perangkat lunak ini juga digunakan sebagai media replikasi konten untuk mereplikasi konten yang berada pada *origin server* menuju *content server.*
- ***Program bash shell*** **–** digunakan untuk menjalankan beberapa program yang ditulis dalam shell. Program bash shell ini digunakan untuk beberapa fungsi seperti melakukan pengecekan *CPU usage* secara berkala dan mencatat beberapa log. Beberapa program bash dieksekusi dengan bantuan *linux cron* agar program berjalan pada background proses.

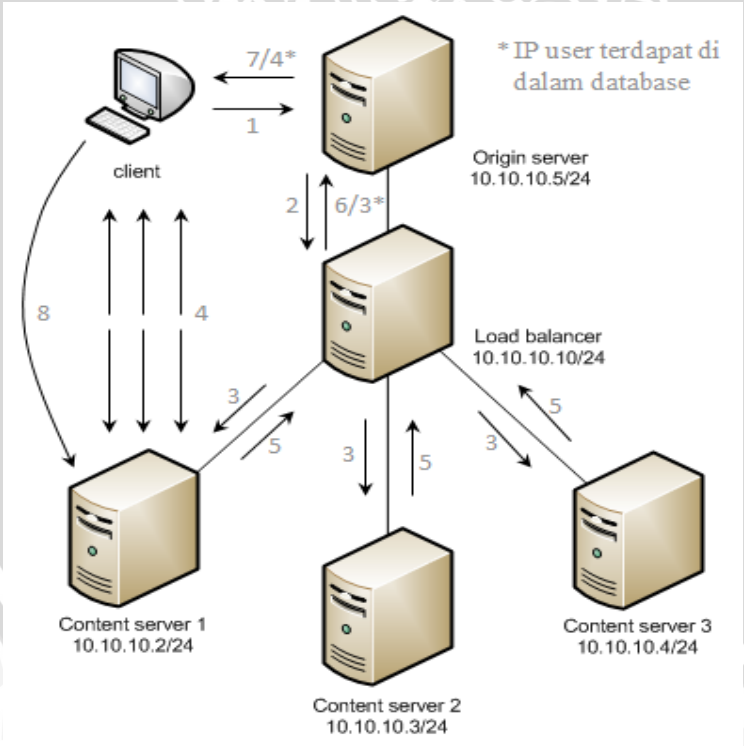
3.2.3. Desain

Pada tahap desain akan dijelaskan mengenai rancangan detil sistem *load balancing* yang akan dibangun mengacu pada tahap arsitektur sebelumnya. Bagaimana sistem *load balancing* dapat bekerja dan proses perbandingan *content server* terbaik secara lebih rinci akan dijelaskan pada tahapan ini. Desain sistem ini yang nantinya akan digunakan pada tahapan implementasi.

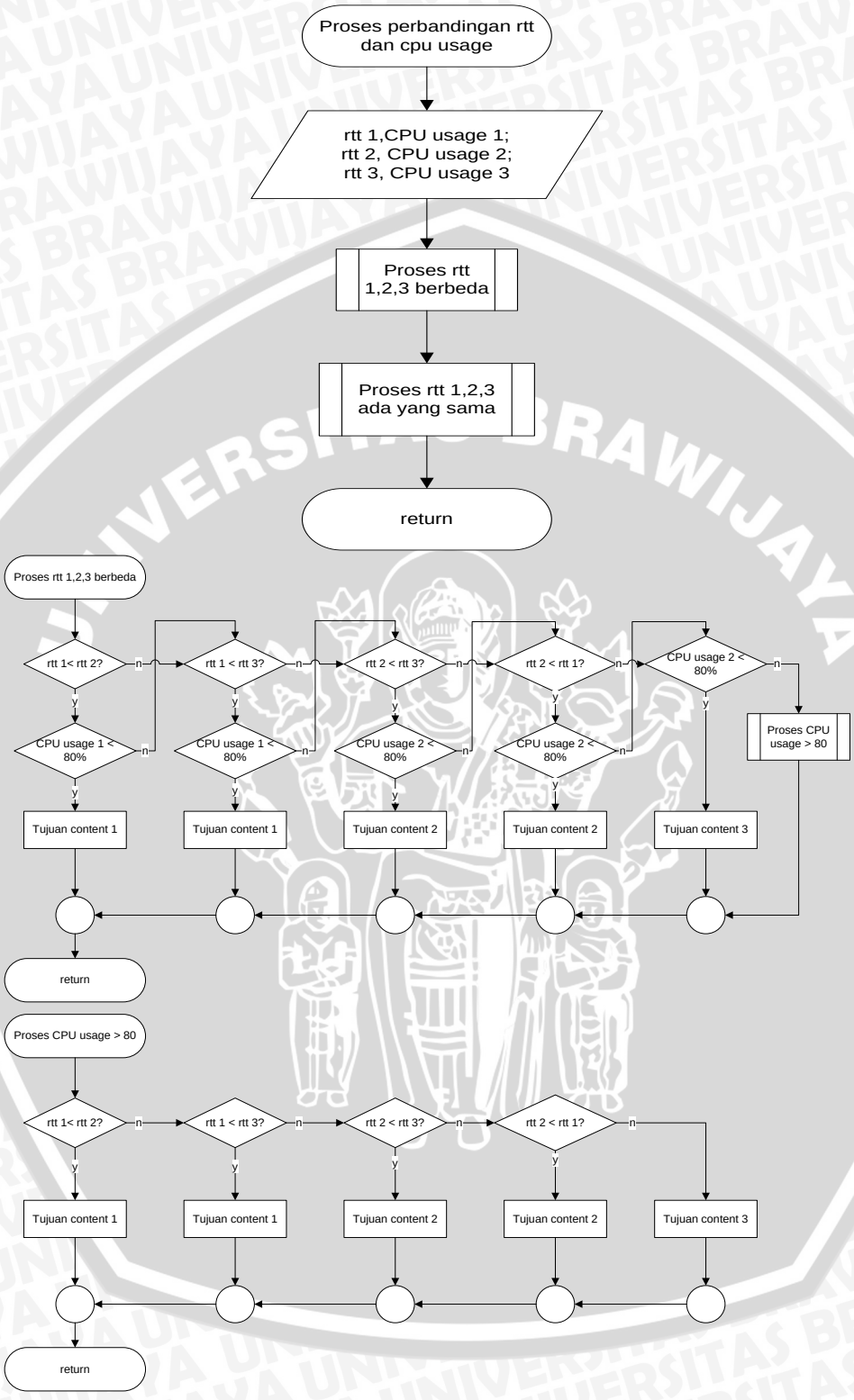
Untuk mendapatkan *rtt* user terhadap *content server*, sistem melakukan *ping* kepada user. Sementara *CPU usage* didapatkan dari masing-masing *content server*. Hasil dari pangukuran *rtt* dan *CPU usage* masing-masing *content server* kemudian akan dikirimkan menuju *load balancer* untuk melakukan perbandingan hasil pengukuran tersebut. Hasil terbaik akan dijadikan sebagai alamat konten yang akan dituju oleh user. Alamat IP user akan disimpan sementara didalam *database*. Hal ini dilakukan agar untuk dalam jangka waktu tertentu user tersebut

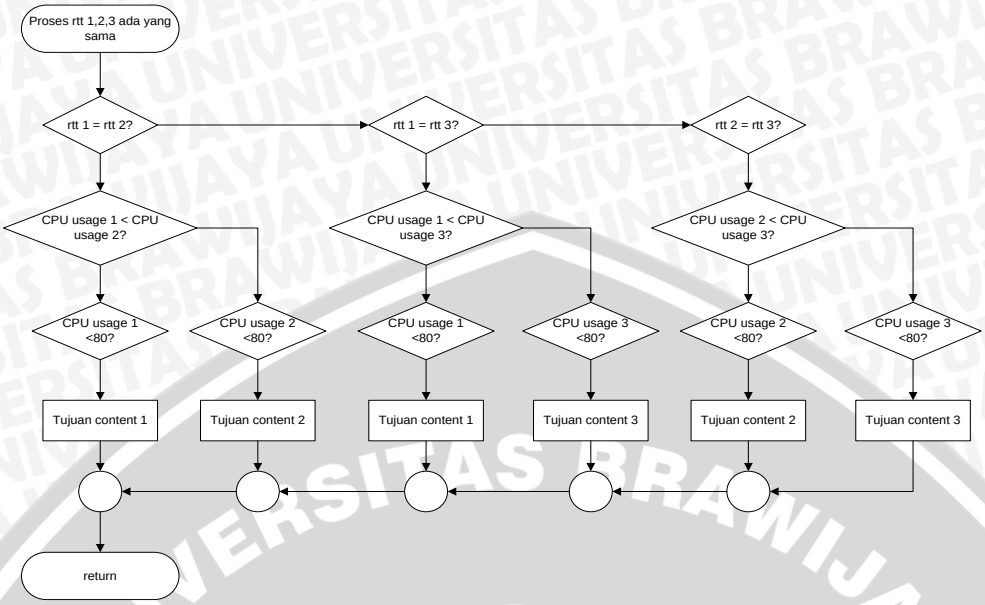
mengakses kembali sistem tidak perlu melakukan pengukuran *rtt* dan *CPU usage* lagi.

Alamat konten yang dituju oleh user adalah *content server* yang sesuai dengan hasil pengukuran sebelumnya. Jika jangka waktu untuk menyimpan IP user sudah habis maka sistem akan melakukan pengukuran *rtt* dan *CPU usage* kembali apabila user tersebut kembali melakukan akses terhadap halaman web tersebut. Selain itu pada *content server* dilakukan proses pengecekan *CPU usage* secara terus menerus, apabila *CPU usage* menunjukkan angka lebih besar dari 80% maka *content server* akan mengirimkan pesan kepada *load balancer* untuk menghapus seluruh data alamat IP user yang pernah mengakses pada database. Hal ini dilakukan agar sistem melakukan pengukuran kembali ketika user berikutnya mengakses layanan walaupun alamat IP user tersebut pernah mengakses sebelumnya. Penjelasan skema sistem dijelaskan dengan diagram alir pada gambar 3.6 dan diagram alir perbandingan *content server* pada gambar 3.7.



Gambar 3.6. Alur sistem *load balancing*


Proses perbandingan rtt dan cpu usage
rtt 1,CPU usage 1;
rtt 2, CPU usage 2;
rtt 3, CPU usage 3
Proses rtt 1,2,3 berbeda
Proses rtt 1,2,3 ada yang sama
return
Proses rtt 1,2,3 berbeda
rtt 1< rtt 2?
rtt 1 < rtt 3?
rtt 2 < rtt 3?
rtt 2 < rtt 1?
CPU usage 2 < 80%
CPU usage 1 < 80%
CPU usage 1 < 80%
CPU usage 2 < 80%
CPU usage 2 < 80%
Proses CPU usage > 80
Tujuan content 1
Tujuan content 1
Tujuan content 2
Tujuan content 2
Tujuan content 3
return
Proses CPU usage > 80
rtt 1< rtt 2?
rtt 1 < rtt 3?
rtt 2 < rtt 3?
rtt 2 < rtt 1?
Tujuan content 1
Tujuan content 1
Tujuan content 2
Tujuan content 2
Tujuan content 3
return



Gambar 3.7. Algoritma pengukuran metrik pemilihan *server*

Penjelasan alur sistem:

- Proses awal yang dilakukan user adalah mengakses *origin server.*
- *Origin server* mengambil alamat IP user dan mengirimkannya ke *load balancer.*
- *Load balancer* mengecek apakah IP user ada di dalam *database*.
    - Jika ada maka *load balancer* mengirim pesan kepada *origin server* bahwa alamat IP tersebut pernah mengakses.
        - alamat konten yang ada di dalam halaman web akan diganti dengan alamat *content server* sesuai isi dari *database*.
    - Jika tidak ada *load balancer* melakukan ping terhadap seluruh *content server* untuk mengecek ketersediaan *server*.
    - *Load balancer* meneruskan alamat IP user tersebut ke seluruh *content server.*

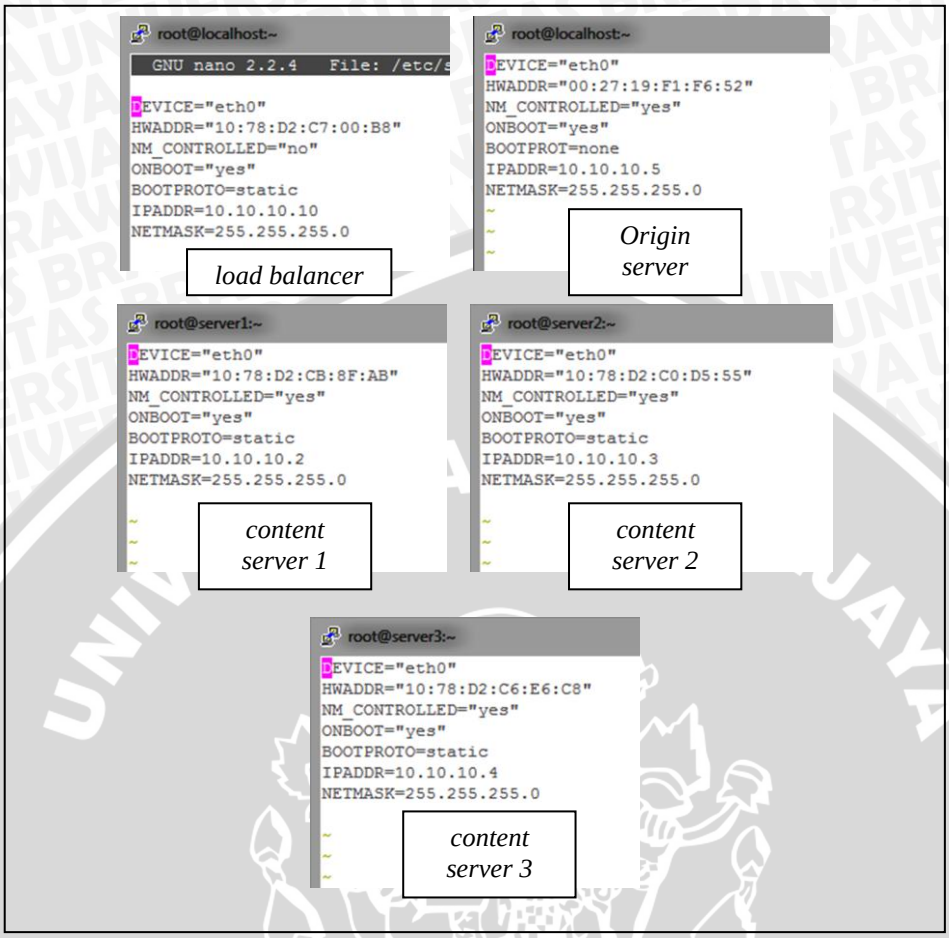
- Masing-masing *content server* melakukan ping terhadap user dan mengambil data *rtt* rata-rata dari hasil ping tersebut, kemudian menghitung *CPU usage* dari masing-masing *content server* yang sedang terjadi. Hasil *rtt* dan *CPU usage* yang diperoleh diteruskan menuju *load balancer*.
- *Load balancer* melakukan perbandingan dari data hasil masing-masing *content server* dan mencatat hasilnya ke dalam *database* antara lain *data* IP user, *CPU usage*, *rtt*, IP *content server*, dan *domain name content server*.
- *Load balancer* mengirimkan *domain name* dari *content server* tujuan sesuai dengan hasil pengukuran dalam *database* kepada *origin server*.
- Alamat konten yang ada di dalam halaman web akan diganti dengan *domain name content server* yang diperoleh dari *load balancer*.

BAB IV
IMPLEMENTASI

Pada bab ini akan membahas mengenai langkah-langkah dalam pembuatan sistem *load balancing*. Tahapan pembuatan sistem mengacu pada tahapan perancangan yang terdiri dari sebuah komputer sebagai *origin server*, sebuah komputer sebagai *load balancer*, dan tiga buah komputer sebagai *content server*. Langkah-langkah yang dilakukan penulis meliputi implementasi sistem operasi dan jaringan, konfigurasi perangkat lunak dan pembuatan berkas-berkas yang dibutuhkan dalam pembuatan sistem *load balancing*. Pada tahapan pengujian mengacu pada analisis kebutuhan sistem yang akan dilakukan dengan beberapa skenario pengujian fungsional maupun non-fungsional.

4.1. Implementasi Sistem Operasi dan Jaringan Sistem

Langkah awal yang perlu dilakukan pada seluruh komponen sistem *load balancing* yang akan dibangun yaitu menghubungkan seluruh komponen sistem yang ada menggunakan kabel UTP dan melalui switch. Khusus komponen *load balancer* menggunakan sistem operasi Fedora 14-x86_64 dengan fitur *graphical desktop*. Untuk komponen sistem lainnya menggunakan sistem operasi Fedora 14-x86_64 dengan fitur minimalis. Setelah instalasi sistem operasi pada seluruh komponen sistem selesai dilakukan berikutnya adalah melakukan konfigurasi pada jaringan *ethernet* masing–masing komponen sistem. Mengacu pada topologi sistem yang telah dijelaskan pada bagian perancangan, komponen sistem menggunakan segmen IP 10.10.10.0/24. Setelah konfigurasi pada seluruh komponen sistem selesai dilakukan, berikutnya adalah melakukan pengujian jaringan sistem dengan melakukan ping satu sama lain. Konfigurasi jaringan *ethernet* dapat dilihat pada gambar 4.1.



Gambar 4.1. Konfigurasi jaringan *ethernet* komponen sistem *load balancing*

4.1. Implementasi Load Balancer

Setelah proses implementasi jaringan selesai dilakukan, langkah selanjutnya adalah konfigurasi perangkat lunak dan pembuatan beberapa berkas pada load balancer. Hal-hal yang dilakukan antara lain konfigurasi DNS server, konfigurasi MySQL, dan pembuatan beberapa berkas yang dibutuhkan dalam membangun sistem load balancing.

4.1.1. Konfigurasi DNS Server

Konfigurasi yang pertama dilakukan adalah DNS *server* dengan menggunakan *Bind9*. Berkas konfigurasi DNS *server* terletak pada */etc/named.conf*. Pada berkas ini yang dilakukan adalah menambah zona dan

mengubah *allow-query {127.0.0.1;}* menjadi *allow-query {any;}* agar seluruh alamat IP dapat mencari *domain name* melalui DNS *server* ini. Zona yang ditambahkan pada berkas ini yaitu cao.com dan nugro.com. Zona cao.com dimiliki oleh *origin server*, sedangkan zona nugro.com dimiliki oleh masing-masing *content server*. Masing-masing zona yang ditambahkan merujuk pada berkas yang terletak pada */var/named/*. Pada masing-masing berkas zona berisi keterangan mengenai zona tersebut seperti alamat IP dan nama lain dari zona tersebut. Konfigurasi DNS *server* dan zona DNS dapat dilihat pada gambar 4.2 dan 4.3.

```
options {
      listen-on port 53 { any; };
      listen-on-v6 port 53 { any; };
      directory   "/var/named";
      dump-file   "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file
"/var/named/data/named_mem_stats.txt";
      allow-query     { any; };
      recursion yes;

      dnssec-enable yes;
      dnssec-validation yes;
      dnssec-lookaside auto;

      /* Path to ISC DLV key */
      bindkeys-file "/etc/named.iscdlv.key";
      managed-keys-directory "/var/named/dynamic";
};
logging {
        channel default_debug {
                file "data/named.run";
                severity dynamic;
        };
};

zone "cao.com." IN {
      type master;
      file "cao.com.db";
};
zone "." IN {
      type hint;
      file "named.ca";
};
include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
```

Gambar 4.2. Konfigurasi DNS *server*

```
Berkas "cao.com.db"
$TTL 1H
@       SOA     @       root.cao.com. (   4      3H     1H     1W     1H )
        NS      @
        IN      1H      A       10.10.10.5
        IN      1H      NS      s1
        IN      1H      NS      s2
        IN      1H      NS      s3
        NS      @
s1      IN      1H      A       10.10.10.2
s2      IN      1H      A       10.10.10.3
s3      IN      1H      A       10.10.10.4
```

Gambar 4.3. Konfigurasi zona DNS

Agar masing-masing komponen dapat melakukan pencarian *domain name* pada DNS *server* yang telah dikonfigurasi perlu ditambahkan alamat IP DNS *server* tersebut pada masing-masing komponen. Pada *load balancer* alamat IP DNS *server* yang ditambahkan adalah alamat *localhost.* Untuk menambahkan alamat IP DNS *server* dapat dilakukan pada berkas *resolv.conf* yang terletak pada */etc/resolv.conf.*

```
#pada load balancer
nameserver 127.0.0.1

#pada content server dan origin server
Nameserver 10.10.10.10
```

Gambar 4.4. Konfigurasi *resolv.conf*

4.1.2. Konfigurasi MySQL

Langkah berikutnya yaitu membuat *database* untuk menampung data hasil pengukuran dengan *MySQL*. *Database* ini memiliki sebuah tabel yang berisi data IP user, *CPU usage*, *rtt content server* terhadap user, IP *content server*, zona DNS dan waktu input.

```
CREATE DATABASE lobal;
CREATE TABLE param(ip_src VARCHAR(15), cpu_load float(7,2), rtt
float(10,3), ip_dst VARCHAR(15), zone VARCHAR(25), time
VARCHAR(45), PRIMARY KEY(ip_src));
```

Field ip_src menampung alamat IP user yang pernah mengunjungi layanan *HTTP* sebelumnya. *Field cpu_load* menampung *CPU usage* pada *content server* yang terkecil. *Field rtt* menampung data *rtt content server* yang terkecil.

Field ip_dst menampung alamat IP *content server* yang menjadi tempat konten tujuan user berada. *Field zones* menampung *domain name content server* tujuan user. *Field time* menampung waktu ketika suatu data diinput.

4.1.3. Pembuatan Berkas dan Program

Setelah seluruh instalasi dan konfigurasi selesai dilakukan, proses selanjutnya adalah pembuatan beberapa berkas *php*, *bash shell*, dan teks. Daftar berkas berikut yang berada pada *load balancer* dan penjelasan singkat dari masing-masing berkas dapat dilihat pada tabel 4.1.

Tabel 4.1. Daftar berkas pada *load balancer*

No	Nama Berkas	Deskripsi Singkat
1	portal.php	Berkas ini berfungsi untuk melakukan pengecekan alamat IP user pada *database* user dan mengeksekusi berkas measure.php untuk melakukan pengukuran metrik
2	measure.php	Berkas ini berfungsi untuk mengeksekusi fungsi pengukuran *rtt* dan *CPU usage* pada *content server* dan melakukan perbandingan untuk mendapatkan hasil yang terbaik.
3	hsl[ip].txt	Berkas ini berfungsi untuk menyimpan hasil pengukuran *rtt* dan *CPU usage* dari masing-masing user sementara.
4	log[ip].txt	Berkas ini berfungsi untuk menyimpan hasil pengukuran *rtt* dan *CPU usage* dari masing-masing user sementara.
5	ip.txt	Berkas ini menampung alamat IP user sementara yang ditangkap oleh berkas

		portal.php
6	update.php	Berkas ini berfungsi untuk menghapus data hasil pengukuran pada *database* yang dieksekusi dalam jangka waktu tertentu dengan bantuan Linux *cron*.
7	cekload.php	Berkas ini berfungsi untuk mengeksekusi clean.php untuk menghapus data *content server* pada *database* sistem ketika *CPU usage content server* tersebut melebihi batas *overload server*.
8	clean.php	Berkas ini berfungsi untuk menghapus data IP user pada *database* sistem. Berkas ini dieksekusi oleh berkas cekload.php

- **portal.php**

Berkas ini berfungsi untuk menerima IP user yang diteruskan oleh *origin server* dan melakukan pengecekan apakah IP tersebut berada di dalam *database* atau tidak. Jika ada *domain name* dari *content server* tujuan dikirimkan ke *origin server*. Jika tidak ada, berkas *measure.php* dieksekusi untuk mendapatkan *domain name* dari *content server* tujuan dan mengirimkannya ke *origin server*.

```
$result = socket_listen($socket, 3) or die("Could not set up
socket listener\n");

$spawn = socket_accept($socket) or die("Could not accept
incoming connection\n");

$input = socket_read($spawn, 1024) or die("Could not read
input\n");
.
.
.
$sql_tran = "SET TRANSACTION ISOLATION LEVEL READ
```

```
UNCOMMITTED;";

$sql =  "SELECT zones FROM param where ip_src='$input';";

mysql_select_db('lobal');

mysql_query( $sql_tran, $conn );
$retval=mysql_query( $sql, $conn );

if(!$retval)
  die('SQL ERROR: '.mysql_error());

$num_rows = mysql_num_rows($retval);
if($num_rows==0){
      shell_exec('php measure.php');
      $retval=mysql_query( $sql, $conn );
      $addr=mysql_result($retval, 0);
      $output=$addr;
      socket_write($spawn, $output, strlen ($output)) or
die("Could not write output\n");
}

else if($num_rows>0) {
      $addr=mysql_result($retval, 0);
      $output=$addr;
      socket_write($spawn, $output, strlen ($output)) or
die("Could not write output\n");
}
```

- **measure.php**

Berkas ini menjalankan fungsi untuk melakukan perbandingan hasil pengukuran *rtt* dan *CPU usage* dari seluruh *content server* dan memasukkan hasil perbandingan tersebut ke dalam *database* sistem. Hal pertama yang dilakukan adalah memanggil berkas *ip.txt* untuk mendapatkan IP user yang kemudian dikirim melalui soket menuju seluruh *content server*.

Proses pengiriman dijalankan melalui proses background secara bersamaan dengan bantuan *php pcntl_fork*. Sebelum *load balancer* mengirim alamat IP user ke masing-masing *content server*, *load balancer* mengecek masing-masing *content server* apakah terhubung ke dalam jaringan atau tidak dengan melakukan ping kepada masing-masing *content server*. *Content server* dianggap mati apabila *packet loss* mencapai 100% saat dilakukan ping. Nilai maksimal *rtt* dan *CPU load* pada *content server* tersebut akan diberikan

sehingga *load balancer* tidak akan memilih *content server* tersebut sebagai tujuan user. Data hasil pengukuran dari seluruh *content server* disimpan dalam sebuah berkas berdasarkan IP user.

```
for($i=0;$i<3;$i++) {
      $pid=pcntl_fork();
      if (!$pid) {
         $cekhost=shell_exec("ping -c 2 ".$host[$i]." | grep
loss | awk '{print $8+0}'");
         echo $cekhost;
         if($cekhost==100) {
               $re=PHP_EOL.$ip."|99|99999|10.10.10.10|load
balancer";
               $hasil[$i]=$re;
               echo $hasil[$i];
         }
         else if($cekhost!=100){
               $sock= socket_create(AF_INET, SOCK_STREAM, 0)
or die("Could not create socket\n");
               $co = socket_connect($sock, $host[$i], $port)
or die("Could not connect data to server\n");
               socket_write($sock, $ip, strlen($ip)) or
die("Could not send data to server\n");
               $re = socket_read($sock, 1024) or die("Could
not read server response\n");
               $hasil[$i]=$re;
               echo $hasil[$i];
         }
         $fh=fopen($file_hsl,'a') or die("can't open file");
         fwrite($fh, $hasil[$i]);
         fclose($fh);
         $fh=fopen($file_log,'a') or die("can't open file");
         fwrite($fh, $hasil[$i].$date);
         fclose($fh);
         die();
      }
}
```

Setelah mendapatkan hasil pengukuran dari masing-masing *content server*, berikutnya adalah memecah kembali data-data hasil pengukuran dari masing-masing *content server* untuk dimasukkan ke dalam sebuah fungsi yang digunakan untuk membandingkan hasil pengukuran tersebut. Dalam fungsi perbandingan tersebut data yang dibandingkan adalah *rtt* dan *CPU usage.* Berikut adalah contoh kode untuk memecah data menjadi tiga bagian

berdasarkan jumlah *content server* dan fungsi perbandingan untuk mendapatkan *rtt* dan *CPU usage* terbaik dari ketiga *content server*.

```
function sel_server($s_ser1, $s_ser2, $s_ser3)
{
      list($ip1, $l1, $r1)=explode("|", $s_ser1);
      list($ip2, $l2, $r2)=explode("|", $s_ser2);
      list($ip3, $l3, $r3)=explode("|", $s_ser3);

      $fl1=floatval($l1);
      $fr1=floatval($r1);

      $fl2=floatval($l2);
      $fr2=floatval($r2);

      $fl3=floatval($l3);
      $fr3=floatval($r3);

      $tmp0=null;
      $tmp1=null;

      if($fr1 != $fr2 || $fr1 != $fr3 || $fr2 != $fr3){
            if($fr1 < $fr2 || $fr1 < $fr3) {if ($fl1 < 80)
$tmp0=$s_ser1;}
            else if($fr2 < $fr3 || $fr2 < $fr1) {if ($fl2 <
80) $tmp0=$s_ser2;}
            else if($fl3 < 80) {$tmp0=$s_ser3;}
            else if($fl1>80 && $fl2>80 && $fl3>80){
                  if($fr1 < $fr2 || $fr1 < $fr3)
$tmp0=$s_ser1;
                  else if($fr2 < $fr3 || $fr2 < $fr1)
$tmp0=$s_ser2;
                  else $tmp0=$s_ser3;
            }
      }

      if($fr1 == $fr2 || $fr1 == $fr3 || $fr2 == $fr3){

            if($fr1 == $fr2){
                  if($fl1 < $fl2 && $fl1 < 80) $tmp0=$s_ser1;
                  else if($fl2 < 80) $tmp0=$s_ser2;
            }

            else if($fr1 == $fr3){
                  if($fl1 < $fl3 && $fl1 < 80) $tmp0=$s_ser1;
                  else if($fl3 < 80) $tmp0=$s_ser3;
            }

            else if($fr2 == $fr3){
                  if($fl2 < $fl3 && $fl2 < 80) $tmp0=$s_ser2;
                  else if($fl3 < 80) $tmp0=$s_ser3;
            }
```

```
            else if($fl1>80 && $fl2>80 && $fl3>80){
                    if($fl1 < $fl2 || $fl1 < $fl3)
$tmp0=$s_ser1;
                    else if($fl2 < $fl3 || $fl2 < $fl1)
$tmp0=$s_ser2;
                    else $tmp0=$s_ser3;
            }

      }
      return $tmp0;
}
```

Setelah data hasil perbandingan diperoleh, berikutnya adalah memecah data tersebut menjadi bagian yang lebih kecil. Tujuannya adalah untuk memudahkan dalam memasukkan data tersebut ke dalam *database* sistem. Berikut adalah kode untuk memecah data hasil perbandingan dan memasukkannya ke dalam *database.*

```
list($ip_s, $cpu, $rtt, $ip_d, $zone)=explode("|", $select);
echo PHP_EOL.$ip_s."|".$cpu."|".$rtt."|".$ip_d."|".$zone;

.
.
.

if(! $conn)
  die('could not connect: '.mysql_error());

$sql =  "INSERT INTO param ".
      "(ip_src, cpu_load, rtt, ip_dst, zones, time) ".
      "VALUES ( '$ip_s', '$cpu', '$rtt', '$ip_d', '$zone',
now() )";

mysql_select_db('lobal');
$retval=mysql_query( $sql, $conn );
if(! $retval)
  die('could not enter data: '.mysql_error());

echo "</br>entered data successfully\n";
mysql_close($conn);
```

- **hsl[ip].txt**

Berkas ini merupakan output dari pengukuran *rtt* dan *CPU usage* yang dilakukan oleh masing-masing *content server* terhadap user. Berkas ini disimpan berdasarkan alamat IP user yang mengakses layanan *HTTP* sistem

load balancing. Masing-masing *content server* memberikan output alamat IP user, *CPU usage*, *rtt*, alamat IP *content server* dan zona DNS. Data yang tersimpan pada berkas ini merupakan data yang terakhir diperoleh. Berikut contoh salah satu berkas dengan alamat IP user 10.10.10.32.

```
IP Client | CPU usage | RTT | IP server | Zone

10.10.10.32|76|89.350|10.10.10.2|s1.cao.com
10.10.10.32|86.4|77.130|10.10.10.3|s2.cao.com
10.10.10.32|58.2|131.560|10.10.10.4|s3.cao.com
```

Gambar 4.5. Berkas hsl10.10.10.32.txt

- **log[ip].txt**

Berkas ini hampir serupa dengan berkas hsl[ip].txt hanya saja perbedaannya adalah berkas ini memiliki format waktu untuk mengetahui kapan hasil pengukuran yang terjadi dan data hasil pengukuran tidak ditimpa dengan data yang terbaru seperti pada berkas hsl[ip].txt. berikut contoh salah satu berkas dengan alamat IP user 10.10.10.32.

```
===============
12:37:41

IP Client | CPU usage | RTT | IP server | Zone

10.10.10.32|76|89.350|10.10.10.2|s1.cao.com
10.10.10.32|86.4|77.130|10.10.10.3|s2.cao.com
10.10.10.32|58.2|131.560|10.10.10.4|s3.cao.com
```

Gambar 4.6. Berkas log10.10.10.32.txt

- **ip.txt**

Berkas ini berfungsi sebagai media penyimpanan sementara alamat IP user. Setiap user yang mengakses layanan *HTTP* sistem *load balancing* akan diambil alamat IP-nya dan menyimpannya sementara didalam berkas ini.

- **update.php**

Berkas ini berfungsi untuk menghapus data user didalam *database* sistem *load balancing* yang terletak pada *load balancer*. Terdapat

pengecualian apabila data user di dalam *database* belum mencapai 15 menit, maka data tersebut tidak akan dihapus. Berkas ini dieksekusi setiap 15 menit sekali dengan bantuan *Linux cron*. Berikut *SQL command* untuk menghapus data yang diinput lebih dari 15 menit. Waktu 15 menit yang ditentukan dapat berubah-ubah sesuai dengan kondisi di lapangan sesuai kebijakan yang ada.

```
DELETE FROM param WHERE time < NOW() - INTERVAL 15 MIN
```

- **cekload.php**

Berkas ini berfungsi untuk memeriksa kondisi *CPU usage* pada *content server* dan akan mengeksekusi berkas *clean.php* apabila salah satu *CPU usage* pada *content server* melewati batas penggunaan 80%. *Content server* akan mengirim pesan kepada *load balancer* apabila *CPU usage* pada *content server* tersebut melewati batas penggunaan 80%. Alamat IP *content server* yang megalami *overload* disimpan ke dalam file sementara untuk digunakan pada berkas *clean.php*.

```
while(true){
      $result = socket_listen($socket, 3) or die("Could not
set up socket listener\n");
      $spawn = socket_accept($socket) or die("Could not
accept incoming connection\n");
      $input = socket_read($spawn, 1024) or die("Could not
read input\n");

      echo PHP_EOL."IP: ".$input." overload";
      $fh=fopen($ip_ov,'w') or die("can't open file");
      fwrite($fh, $input);
      fclose($fh);
      shell_exec("php clean.php");
}
```

- **clean.php**

Berkas ini menghapus alamat IP user yang ditujukan kepada *content server* yang mengalami *overload CPU usage* dalam *database* sehingga apabila alamat IP user tersebut mengakses pada waktu berikutnya sistem *load balancing* akan melakukan pengukuran kembali. Berkas dieksekusi oleh berkas *cekload.php* apabila *content server* mengalami *overload CPU usage*.

```
$input = socket_read($spawn, 1024) or die("Could not read
input\n");
$sql = "DELETE FROM param WHERE ip_src='$input'";

mysql_select_db('lobal');
$retval=mysql_query( $sql, $conn );
```

4.2. Implementasi *Content Server*

Pada tahapan implementasi *content server* yang dilakukan adalah membuat beberapa berkas yang akan digunakan untuk melakukan pengukuran *rtt* dan *CPU usage*. Selain membuat beberapa berkas, penulis menggunakan *Apache Web Server* sebagai *web server* agar konten yang terdapat pada *content server* dapat diakses oleh user. Daftar berkas yang berada pada *content server* beserta penjelasan singkat dapat dilihat pada tabel 4.2.

Tabel 4.2. Daftar berkas pada *content server*

No	Nama Berkas	Deskripsi Singkat
1	server.php	Berkas ini berfungsi untuk mengeksekusi fungsi pengukuran *rtt* dan *CPU usage* dan mengirimkan hasilnya menuju load balancer.
2	load.sh	Berkas ini berfungsi untuk mengambil data *CPU usage* yang diperoleh dengan menggunakan *command top*.
3	scanload.sh	Berkas ini berfungsi untuk melakukan pengecekan *CPU usage* secara terus-menerus yang dieksekusi oleh berkas *scanload.php*.
4	scanload.php	Berkas ini mengirimkan pesan kepada *load balancer* apabila *CPU usage* pada *content server* melebihi batas overload.

- **server.php**

Berkas ini berfungsi untuk melakukan pengukuran *rtt* dan *CPU usage* pada masing-masing *content server*. Berkas ini selalu berjalan menunggu koneksi soket dari *load balancer* yang membawa pesan berisi alamat IP user. Kemudian *content server* melakukan ping terhadap IP user tersebut dan mengambil rata-rata *rtt* dari hasil ping tersebut. Setelah mendapatkan rata-rata *rtt* hasil ping kemudian melakukan pengukuran *CPU usage* dengan mengeksekusi berkas *bash load.sh*. Hasil kedua pengukuran tersebut kemudian dikirim kembali menuju *load balancer* dan disimpan didalam berkas sebagai data mentah. Hasil yang dikirim dan disimpan tidak hanya *rtt* dan *CPU usage*, tetapi juga alamat IP user, alamat IP *content server* dan zona DNS. Berikut adalah salah satu contoh berkas *server.php* pada *content server* satu.

```
$ping = trim(shell_exec("ping -c 3 ".$input." | tail -1| awk
'{print $4}' | cut -d '/' -f 2"));

$load = trim(shell_exec("./load.sh"));

$output =
PHP_EOL.$input."|".$load."|".$ping."|"."10.10.10.2"."|"."s1.n
ugro.com";

echo $output;
socket_write($spawn, $output, strlen ($output)) or die("Could
not write output\n");
```

- **load.sh**

Berkas ini berfungsi untuk mendapatkan total penggunaan *CPU* yang sedang digunakan dengan menggunakan *command top*. *Command* ini dilakukan sebanyak dua iterasi untuk mendapatkan hasil penggunaan *CPU* yang akurat. Berkas ini dieksekusi oleh berkas *server.php* yang hasilnya kemudian dikirim menuju *load balancer* bersamaan dengan *rtt* untuk dilakukan pengukuran.

```
#!/bin/sh
top -n2 | awk '/Cpu\(s\):/ {print $2+$3+$4+$6+$7+$8+$9}'
>load.txt
awk 'FNR==2 {print}' load.txt
```

- **scanload.sh**

Berkas ini berfungsi untuk melakukan pengecekan *CPU usage* secara terus-menerus yang dieksekusi oleh berkas *scanload.php.*

```
top=$(top -d 1 -n 10 | awk '/Cpu\(s\):/ {print
$2+$3+$4+$6+$7+$8+$9}' >load1.txt)
max=$(awk '$1 > max {max=$1} END{ print max}' load1.txt)
load=`echo "$max" | bc`
over=`echo "80" | bc`

if [[ "$(echo $load '>' $over | bc -l)" -eq 1 ]]
then echo "over"
else echo "not yet"
fi
```

- **scanload.php**

Berkas ini berfungsi untuk mengirimkan pesan kepada *load balancer* apabila *content server* mengalami overload *CPU usage* melalui komunikasi soket. Apabila *CPU usage* melabihi batas overload berkas *scanload.sh* memberi pesan “over”.

```
while(true){
$ret  = shell_exec("/var/www/html/scanload.sh");
if($ret=="over\n"){
      $sock = socket_create(AF_INET, SOCK_STREAM, 0) or
die("Could not create socket\n");
      $co   = socket_connect($sock, $host, $port) or
die("Could not connect data to server\n");
      socket_write($sock, $load, strlen($load)) or
die("Could not send data to server\n");
      echo $load.PHP_EOL;
      socket_close($sock);
}
```

4.3. Implementasi *Origin Server*

Pada tahapan implementasi *origin server* yang dilakukan hanya memasang *Apache Web Server*, karena komponen ini hanya melayani layanan *HTTP*. Halaman *web* yang akan diakses oleh user disimpan pada komponen ini. Penulis hanya membuat sebuah halaman *web* sebagai media pengujian sistem.

- **index.php**

Konten yang ada pada halaman *web* ini terletak di seluruh *content server*. Alamat IP user akan diambil kemudian diteruskan menuju *load balancer* melalui koneksi soket untuk dilakukan pengecekan apakah alamat IP tersebut pernah mengakses sebelumnya. *Url* dari seluruh konten pada halaman ini akan menyesuaikan dengan *content server* yang diperoleh dari hasil pengukuran sistem.

```
$ip=$_SERVER['REMOTE_ADDR'];
$sock= socket_create(AF_INET, SOCK_STREAM, 0) or die("Could
not create socket\n");
$co = socket_connect($sock, $shost, $port) or die("Could not
connect data to server\n");
socket_write($sock, $ip, strlen($ip)) or die("Could not send
data to server\n");

$re = socket_read($sock, 1024) or die("Could not read server
response\n");

$host="http://".$re;
.
.
<img src="<?php echo $host; ?>/home_files/u4_normal.jpg"
class="raw_image"></div>
```

BAB V
PENGUJIAN DAN ANALISIS

Bab V pengujian dan analisis berisi tentang langkah-langkah untuk menguji sistem *load balancing* yang dibangun serta menganalisis mengenai hasil pengujian yang dilakukan.

5.1. Pengujian

Dalam penelitian ini, pengujian dilakukan dalam dua tahapan pengujian yaitu pengujian fungsional dan pengujian non fungsional. Pada pengujian fungsional dilakukan dua buah skenario untuk menguji sistem dapat melakukan perpindahan tujuan konten yang diakses oleh user. Pada pengujian non fungsional dilakukan tiga buah skenario untuk menguji sistem dapat tetap mengakses konten apabila salah satu atau beberapa *content server* sedang *down(availability)* dan untuk melihat perbandingan *CPU usage* pada *origin server* ketika seluruh konten terletak pada *origin server* dan konten tersebar pada tiga buah *content server*.

Untuk melakukan pengujian, penulis menggunakan perangkat lunak Apache JMeter. Apache Jmeter merupakan perangkat lunak berbasis Java yang didesain untuk memberikan beban pada perilaku fungsional dan melakukan pengukuran peforma pada sebuah web server. Apache JMeter mensimulasikan beban tinggi pada sebuah server, jaringan, atau sebuah objek untuk mengetahui kemampuannya atau menganalisa keseluruhan peforma pada kedaan beban yang berbeda.

Untuk melakukan pengujian berupa request http terhadap sebuah web server perlu diperhatikan 3 buah komponen dari apache Jmeter yaitu Group Thread, Controller berupa HTTP sampler, dan listerner didalam HTTP sampler. Group Thread merupakan tahap awal dan yang terpenting dikarenakan seluruh controller dan listener berada didalam group thread. Group thread mengatur

jumlah thread yang akan dibuat selama pengujian dilakukan, ramp up atau waktu yang diperlukan selama pengujian serta mengatur perulangan yang dilakukan oleh setiap thread ketika pengujian.

Thread mewakili jumlah user yang akan disimulasikan mengakses web server yang diuji. Setiap thread akan berjalan secara independen saat mengeksekusi keseluruhan pengujian. Ramp-up menentukan waktu eksekusi yang dilakukan oleh Apache JMeter dengan jumlah thread yang telah ditentukan. Jika digunakan 10 thread dengan ramp-up 10 detik dan perulangan 1 kali makan setiap detik akan ada 1 thread yang akan melakukan pengujian.

5.1.1. Pengujian Fungsional

Pada tahapan pengujian fungsional yang dilakukan adalah melihat perilaku sistem *load balancing* ketika *CPU usage* pada *content server* tujuan user melebihi batas *overload* saat user sedang mengakses konten. Skenario yang dilakukan yaitu user melakukan pengaksesan untuk mendapatkan *content server* tujuan user mengakses konten secara terus-menerus dalam tingkat permintaan yang normal dengan bantuan aplikasi Apache Jmeter. Selama user melakukan *stress* normal pada sistem, *content server* tujuan di *stress* agar *CPU usage* pada *server* tersebut meningkat hingga melewati batas *overload* yang telah ditentukan yaitu 80%. Batas 80% ditentukan sebagai tindakan untuk mengantisipasi peningkatan utilisasi *CPU* yang signifikan pada penggunaan *CPU usage* antara 80% sampai dengan 100%. Pengujian ini dilakukan dengan dua buah skenario *stress load.* Proses pengujian *stress load* mengacu pada tahapan pengujian yang telah dijelaskan sebelumnya pada bab III. *Stress load* pada *content server* juga dilakukan dengan bantuan aplikasi Apache Jmeter. Konfigurasi aplikasi Apache Jmeter untuk *stress* normal adalah sebagai berikut:

- Jumlah Thread (klien) = 50
- Periode Ramp-Up (dalam detik) = 1

- Jumlah Perulangan = Selamanya
- Mengambil konten halaman html = True

Konfigurasi diatas ditentukan oleh penulis berdasarkan kemampuan sumber daya user untuk membuat permintaan *HTTP* yang ideal dengan perulangan selamanya. Pengujian ini dibatasi dalam waktu dua menit karena sumber daya komputer untuk melakukan *stress load* untuk dapat menigkatkan *CPU usage* pada *content server* hingga melebihi batas overload yang ditentukan hanya selama dua menit. Berikut konfigurasi Apache Jmeter untuk *stress load* pada *content server* tujuan:

- Jumlah Thread (klien) = 3000
- Periode Ramp-Up (dalam detik) = 10
- Jumlah Perulangan = 10

Konfigurasi diatas ditentukan berdasarkan kemampuan sumber daya komputer untuk melakukan permintaan *HTTP* dalam jumlah besar. Pengujian yang diharapkan adalah sistem *load balancing* dapat melakukan pengukuran metrik ulang ketika *CPU usage* pada *content server* melebihi batas *overload* dan lokasi konten yang dituju user dapat berpindah menuju *content server* yang tidak memiliki *CPU usage* melebihi batas *overload*. Seluruh data hasil pengujian fungsional ini diambil dari *CPU usage* dari seluruh *content server* dan *summary report* dari aplikasi Apache Jmeter. Masing-masing skenario dilakukan pengujian sebanyak lima kali percobaan untuk mendapatkan data yang lebih akurat.

- **Skenario Satu**

Pada skenario satu, *stress load* yang dilakukan hanya ditujukan pada satu *content server* tujuan user mengakses konten. Tujuan pengujian fungsional skenario satu adalah untuk mengetahui perilaku sistem ketika *CPU usage* pada *content server* tujuan user melebihi batas *overload* yang telah

ditentukan saat user mengakses *content server* tersebut. Pengujian fungsional skenario satu dilakukan dengan memberi permintaan dengan jumlah yang telah ditentukan sebelumnya untuk mendapatkan lokasi *content server* tujuan. Setelah lokasi *content server* diperoleh, *stress load* untuk menaikkan *CPU usage* hingga melebihi batas *overload* dilakukan kepada *content server* tujuan. Hasil dari pengujian fungsional skenario satu dapat dilihat pada tabel 5.1.

Tabel 5.1. Hasil pengujian fungsional skenario satu

Per-co-baan	SERVER 1				SERVER 2				SERVER 3				Server Tujuan	
	sebelum stress load		sesudah stress load		sebelum stress load		sesudah stress load		sebelum stress load		sesudah stress load			
	RTT (ms)	CPU usage (%)	RTT (ms)	CPU usage (%)	RTT (ms)	CPU usage (%)	RTT (ms)	CPU usage (%)	RTT (ms)	CPU usage (%)	RTT (ms)	CPU usage (%)	sebelum stress load	sesudah stress load
1	0.425	0.3	0.266	0.3	0.586	0.7	0.283	0.9	0.412	0.2	0.278	**95.2**	server 3	server 1
2	0.49	0.6	0.442	0.3	0.386	0.3	0.547	**85**	0.391	0.3	0.492	0.3	server 2	server 1
3	0.335	0.5	0.224	**92.9**	0.377	0.8	0.442	0.3	0.351	0.3	0.375	0.2	server 1	server 3
4	0.392	0.5	0.29	0.4	0.486	0.6	0.39	0.4	0.382	0.4	0.274	**94.4**	server 3	server 1
5	0.427	0.4	12.538	0.3	0.392	0.3	7.895	**90.5**	0.446	4.4	12.655	0.3	server 2	server 1

- **Skenario Dua**

Pada skenario dua, *stress load* yang dilakukan ditujukan pada *content server* tujuan user mengakses konten dan sebuah *content server* lainnya. Tujuan pengujian fungsional skenario dua adalah untuk mengetahui perilaku sistem ketika *CPU usage* pada *content server* tujuan user dan sebuah *content server* lainnya melebihi batas *overload* yang telah ditentukan saat user mengakses *content server* tujuan. Pengujian fungsional skenario dua sama seperti pada skenario satu yaitu memberi permintaan dengan jumlah yang telah ditentukan untuk mendapatkan lokasi *content server* tujuan. Setelah lokasi *content server* diperoleh, *stress load* untuk menaikkan *CPU usage* hingga melebihi batas *overload* dilakukan kepada *content server* tujuan dan sebuah *content server* lainnya. Hasil dari pengujian fungsional skenario dua dapat dilihat pada tabel 5.2.

Tabel 5.2. Hasil pengujian fungsional skenario dua

Per-co-baan	SERVER 1				SERVER 2				SERVER 3				Server Tujuan	
	sebelum stress load		sesudah stress load		sebelum stress load		sesudah stress load		sebelum stress load		sesudah stress load			
	RTT (ms)	CPU usage (%)	RTT (ms)	CPU usage (%)	RTT (ms)	CPU usage (%)	RTT (ms)	CPU usage (%)	RTT (ms)	CPU usage (%)	RTT (ms)	CPU usage (%)	sebelum stress load	sesudah stress load
1	0.356	0.7	0.509	**86.2**	0.367	0.4	0.488	0.3	0.381	0.6	0.543	**89.6**	server 1	server 2
2	0.396	0.5	0.472	**89.2**	0.485	0.4	0.384	0.4	0.528	0.5	0.546	**81.1**	server 1	server 2
3	0.497	1.8	0.262	**83.8**	0.508	2	0.191	5.7	0.421	0.4	0.277	**90.5**	server 3	server 2
4	0.455	0.7	0.351	**85.9**	0.445	5	0.206	**92.8**	0.45	0.9	0.329	0.7	server 2	server 3
5	0.426	0.5	0.363	**86.7**	0.458	9.2	0.374	**80.2**	0.43	0.2	0.325	0.3	server 1	server 3

5.1.2. Pengujian Non Fungsional

Pada tahapan pengujian non fungsional yang dilakukan adalah menguji ketersediaan*(availability)* sistem *load balancing* serta membandingkan kinerja *CPU* pada *origin server* dengan konten terpusat dan konten tersebar pada tiga buah *content server*. Untuk menguji ketersediaan sistem *load balancing* yang dibangun dilakukan skenario pengujian dengan mencabut kabel jaringan pada *content server*. Pada skenario ini diharapkan user dapat tetap mengakses konten walaupun *content server* tidak seluruhnya dalam kondisi hidup. Untuk menguji kinerja *CPU* pada *origin server* dengan konten terpusat dan konten tersebar dilakukan skenario pengujian dengan mengakses layanan *HTTP* pada *origin server*. User mengakses halaman web yang berbeda tapi memiliki konten yang sama. Pada pengujian non-fungsional ini dilakukan tiga skenario pengujian, dua skenario pengujian untuk menguji ketersediaan sistem *load balancing* dan satu skenario untuk menguji kinerja *CPU* pada *origin server*.

Pada pengujian non fungsional ini masing-masing skenario dilakukan lima kali percobaan. Data yang diambil dari skenario satu dan dua adalah log hasil perhitungan sistem *load balancing*. Untuk skenario pengujian tiga dilakukan permintaan *HTTP* secara berulang untuk melihat peningkatan *CPU usage* pada *origin server* dengan bantuan aplikasi Apache Jmeter. Sebelum menentukan konfigurasi Apache jMeter yang digunakan untuk pengujian ini penulis melakukan percobaan untuk mengetahui kemampuan sistem *single server* untuk melayani permintaan user. Dari hasil percobaan yang dilakukan menunjukkan sistem *single server* masih dapat mengungguli sistem *load balancing* dengan konfigurasi Apache jMeter sebagai berikut:

- Jumlah Thread (klien) = 10
- Periode Ramp-Up (dalam detik) = 1
- Jumlah Perulangan = 5
- Mengambil konten halaman html = True

Hasil percobaan yang diperoleh dapat dilihat pada tabel 5.3.

Tabel 5.3. Hasil Percobaan Batas Kemampuan Sistem *Single Server*

Sistem	Jumlah Permintaan	Response Time (ms)	Throughput (Permintaan/s)	Throughput (KB/s)
Single Server	29	14591	24.7	10396.2
Load Balancing	31	19806	21	8524.23

Untuk konfigurasi pengujian nonfungsional yang akan dilakukan diperlukan jumlah permintaan yang lebih besar agar dapat memberatkan *server* sehingga *server* dapat tidak melayani permintaan dengan baik. Berikut konfigurasi kontrol grup pada Apache Jmeter untuk pengujian nonfungsional:

- Jumlah Thread (klien) = 100
- Periode Ramp-Up (dalam detik) = 1
- Jumlah Perulangan = 5
- Mengambil konten halaman html = True

Konfigurasi diatas ditentukan oleh penulis berdasarkan kemampuan sumber daya komputer untuk membuat permintaan *HTTP* yang dapat meningkatkan beban *CPU* pada *origin server*. Data yang diambil pada pengujian ini adalah *CPU usage* pada *origin server* dan *summary report* pada Apache Jmeter.

- **Skenario satu**

  Pada skenario satu, untuk mensimulasikan keadaan *content server* yang sedang mati, kabel jaringan pada satu *content server* dicabut. Pengujian skenario satu ini dilakukan dengan tujuan melihat perilaku sistem apabila salah satu *content server* tidak terhubung kedalam jaringan. Pada skenario ini

pengujian dilakukan dengan mengakses halaman web yang disediakan melalui web browser. Data hasil pengujian skenario satu dapat dilihat pada tabel 5.4.

Tabel 5.4. Hasil pengujian non-fungsional skenario satu

Per-co-baan	SERVER 1		SERVER 2		SERVER 3		Server Tujuan
	RTT (ms)	CPU usage (%)	RTT (ms)	CPU usage (%)	RTT (ms)	CPU usage (%)	
1	0.405	0.3	**99999**	**99**	0.412	0.2	server 1
2	0.296	0.3	**99999**	**99**	0.278	0.2	server 3
3	0.43	0.6	**99999**	**99**	0.381	0.3	server 3
4	0.303	0.3	**99999**	**99**	0.322	0.3	server 1
5	0.375	0.5	**99999**	**99**	0.401	0.3	server 1

- **Skenario dua**

Pengujian yang dilakukan pada skenario dua hampir serupa dengan skenario satu, hanya saja kabel jaringan yang dicabut ada dua sehingga hanya satu *content server* yang terhubung ke dalam jaringan. Tujuan dari skenario dua yaitu untuk melihat apakah user dapat tetap mengakses layanan dengan hanya satu *content server* yang terhubung ke dalam jaringan sistem. Data hasil pengujian skenario dua dapat dilihat pada tabel 5.5.

Tabel 5.5. Hasil pengujian non-fungsional skenario dua

Per-co-baan	SERVER 1		SERVER 2		SERVER 3		Server Tujuan
	RTT (ms)	CPU usage (%)	RTT (ms)	CPU usage (%)	RTT (ms)	CPU usage (%)	
1	0.315	0.3	**99999**	**99**	**99999**	**99**	server 1
2	0.275	0.3	**99999**	**99**	**99999**	**99**	server 1
3	0.424	0.2	**99999**	**99**	**99999**	**99**	server 1
4	0.421	0.3	**99999**	**99**	**99999**	**99**	server 1
5	0.315	0.5	**99999**	**99**	**99999**	**99**	server 1

- **Skenario tiga**

Tujuan dari pengujian skenario tiga adalah membandingkan beban kerja *CPU* pada *origin server* dengan sistem *single server* dan sistem *load*

balancing yang dibangun, *throughput* dan *response time* yang dialami oleh user. Pengujian yang dilakukan pada skenario tiga adalah memberi permintaan *HTTP* dengan konfigurasi yang telah dijelaskan sebelumnya pada *origin server*. Pada skenario tiga menggunakan tiga buah komputer user untuk membebani *server* dikarenakan jumlah permintaan yang dapat dilakukan oleh satu komputer user sesuai pada konfigurasi yang telah dijelaskan sebelumnya pada skenario ini. Data hasil pengujian skenario tiga dapat dilihat pada tabel 5.6 hingga 5.9.

Tabel 5.6. Hasil Pengujian *Throughput* dan *Response Time* Skenario Tiga Sistem *Single Server*

Perco-baan	Jumlah Permintaan Layanan HTTP					Throughput Rata-Rata(KB/s)					Response Time Rata-Rata(ms)				
	client 1	client 2	client 3	client 4	client 5	client 1	client 2	client 3	client 4	client 5	client 1	client 2	client 3	client 4	client 5
1	49	31	25	29	28	1320.55	1188.93	1201.1	1214.34	1276.95	91168	128985	150584	136246	133841
2	29	27	29	29	29	1342.25	1244.8	1175.21	1234.93	1335.79	139429	146414	134950	129292	135388
3	25	28	25	25	29	1445.64	1616.98	1471.95	1154.31	1682.79	122341	105733	121780	156347	107194
4	30	29	29	26	29	1338.42	1292.07	1320.23	1166.4	1294.76	147901	141629	141707	154047	141216
5	25	29	33	29	29	1127.71	1305.62	1322.89	1312.01	1299.74	154977	140257	126483	142467	141238
6	32	26	29	25	29	1327.5	1191.91	1303.85	1154.22	1332.22	128396	159144	133892	154390	138898
7	32	29	28	35	29	1305.92	1310.35	1150.68	1313.5	1307.26	130230	139901	139482	121609	139368
8	29	30	34	33	29	1299.14	1337.03	1169.36	1348.23	1294.19	141876	147199	118103	137874	140823
9	30	25	32	25	29	1272.31	1132.55	1421.72	1137.68	1307.7	126341	153085	136024	155845	139813
10	36	29	29	25	30	1306.09	1304.16	1328.74	1130.72	1349.22	125295	140940	140857	157038	146587

Tabel 5.7. Hasil Pengujian *Error Rate*, jumlah permintaan/menit dan *CPU Usage* Skenario Tiga Sistem *Single Server*

Perco-baan	Error Rate (%)					Throughput (Req/min)					CPU Usage Rata-Rata(%)
	client 1	client 2	client 3	client 4	client 5	client 1	client 2	client 3	client 4	client 5	origin server
1	48.98	22.58	4	10.34	3.57	5.3	3.4	2.9	3.2	3	1.197
2	3.45	11.11	17.24	31.03	20.69	3.2	3	3.2	3.6	3.2	1.189
3	0	0	0	0	0	3.4	3.8	3.5	2.7	4	1.171
4	3.33	10.34	3.45	3.85	3.45	3.2	3.1	3.1	2.8	3.1	1.166
5	0	3.45	12.12	0	13.79	2.7	3.1	3.6	3.1	3.1	1.118
6	15.62	7.69	10.34	0	3.45	3.5	2.8	3.3	2.7	3.2	1.147
7	15.62	3.45	10.71	20	3.45	3.4	3.1	3.1	3.8	3.1	1.157
8	6.9	13.33	26.47	21.21	10.34	3.1	3.2	3.8	3.5	3.1	1.139
9	6.67	4	6.25	4	13.79	3.2	2.7	3.6	2.7	3.1	1.156
10	22.22	3.45	10.34	0	3.33	3.9	3.1	3.2	2.7	3.2	1.171

Tabel 5.8. Hasil Pengujian *Throughput* dan *Response Time* Skenario Tiga Sistem *Load Balancing*

Perco-baan	Jumlah Permintaan Layanan HTTP					Throughput Rata-Rata(KB/s)					Response Time Rata-Rata(ms)				
	client 1	client 2	client 3	client 4	client 5	client 1	client 2	client 3	client 4	client 5	client 1	client 2	client 3	client 4	client 5
1	29	25	29	25	30	3362.02	5331.93	3363.37	2966.01	3449.52	55004	34136	54342	59413	58159
2	25	29	29	27	26	2920.81	5945.95	3402.32	3344.55	3804.42	60851	31204	52993	55244	45455
3	29	29	29	25	29	3344.34	6093.81	3345.08	3102.34	3272.12	55841	30183	54679	57619	56018
4	29	29	24	25	29	3327.29	5947.11	2832.16	3071.69	3387.82	56149	31141	56639	58732	55006
5	25	29	29	25	29	2975.24	5930.28	3472.04	3140.15	3269.4	60403	31227	53125	57375	56155
6	25	29	25	25	28	3018.48	5889.25	2968.59	3063.74	3331.01	57583	31687	58957	59515	55543
7	25	25	25	25	29	3119.44	5417.76	2967.64	2912.51	3463.76	56069	33391	59188	62496	54090
8	25	29	25	25	26	3359.93	5955.29	3058.8	2748.62	3100.82	51818	31014	57977	66895	58026
9	27	29	25	25	28	3580.38	6014.66	2967.25	2638.79	3282.86	49657	87461	59053	70071	56356
10	24	29	25	25	29	3338.42	5895.28	3008.21	2589	3378.85	48237	31545	58511	71320	54806

Tabel 5.9. Hasil Pengujian *Error Rate*, jumlah permintaan/menit dan *CPU Usage* Skenario Tiga Sistem *Load Balancing*

Perco-baan	Error Rate (%)					Throughput (Req/min)					CPU Usage Rata-Rata(%)			
	client 1	client 2	client 3	client 4	client 5	client 1	client 2	client 3	client 4	client 5	origin	server1	server2	server3
1	0	0	0	0	0	8	12.7	8	7.1	8.2	0.304	0.856	0.919	14.507
2	0	0	0	0	0	7	14.1	8.1	8	9.1	0.337	0.869	15.421	13.689
3	0	0	0	0	0	8	14.5	8	7.4	7.8	0.324	0.817	15.275	13.753
4	0	0	0	0	0	7.9	14.2	6.7	7.3	8.1	0.306	0.840	15.207	13.768
5	0	0	0	0	0	7.1	14.1	8.3	7.5	7.8	0.318	0.834	0.955	13.513
6	0	0	0	0	0	7.2	14	7.1	7.3	7.9	0.336	0.834	0.883	15.422
7	0	0	0	0	0	7.4	12.9	7.1	6.9	8.2	0.309	0.848	0.929	0.824
8	0	0	0	0	0	8	14.2	7.3	6.5	7.4	0.308	0.860	0.894	0.840
9	0	1.85	0	0	0	8.5	14.3	7.1	6.3	7.8	0.321	0.820	0.844	1.000
10	0	0	0	0	0	7.9	14	7.2	6.2	8	0.330	0.834	15.022	13.990

5.2. Analisis

Analisis dilakukan berdasarkan hasil pengujian pada bab implementasi dan pengujian. Analisis dibagi berdasarkan jenis pengujian yang dilakukan, yaitu pengujian fungsional dan pengujian non-fungsional.

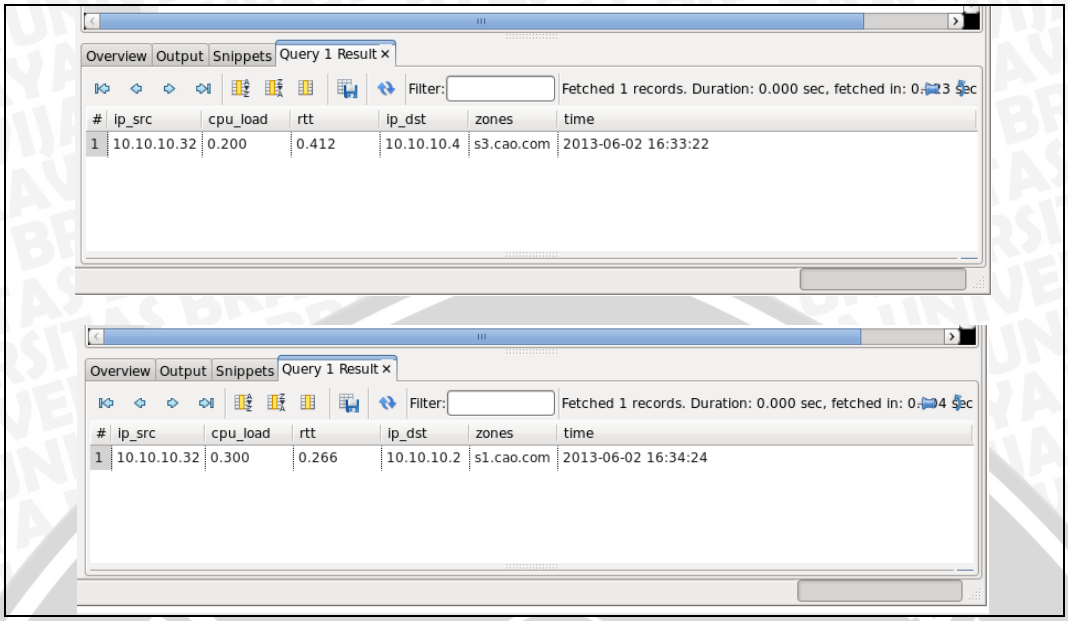
5.2.1. Analisis Pengujian Fungsional

Berdasarkan hasil pengujian fungsional yang dilakukan pada skenario satu dan skenario dua, data menunjukkan bahwa sistem *load balancing* yang dibangun dapat bekerja secara fungsional. Ketika *content server* tujuan menanggung beban kerja hingga melebihi batas penggunaan *CPU* yang ditentukan yaitu pada 80%, maka secara otomatis sistem dapat melakukan pengukuran ulang untuk menentukan tujuan *content server* kembali sehingga user akan berpindah tujuan menuju *content server* yang tidak menanggung beban kerja hingga melebihi batas penggunaan *CPU* yang ditentukan.

Pencatatan log akan dilakukan setiap alamat IP user yang mengakses layanan tidak terdapat didalam *database*. Sistem secara otomatis menghapus data alamat IP pada *database* yang menuju *content server* yang sedang mengalami *overload* penggunaan *CPU*, sehingga sistem akan melakukan pengukuran kembali untuk menentukan *content server* tujuan. Log pengukuran sistem dapat dilihat pada gambar 5.1 dan database sistem pada gambar 5.2.

```
===============
16:33:15

IP Client | CPU usage | RTT | IP server | Zone

10.10.10.32|0.2|0.412|10.10.10.4|s3.cao.com
10.10.10.32|0.3|0.425|10.10.10.2|s1.cao.com
10.10.10.32|0.7|0.586|10.10.10.3|s2.cao.com

===============
16:34:10

IP Client | CPU usage | RTT | IP server | Zone

10.10.10.32|0.3|0.266|10.10.10.2|s1.cao.com
10.10.10.32|0.9|0.283|10.10.10.3|s2.cao.com
10.10.10.32|95.2|0.278|10.10.10.4|s3.cao.com
```

Gambar 5.1. Log Pengukuran Sistem Skenario Satu Percobaan Satu



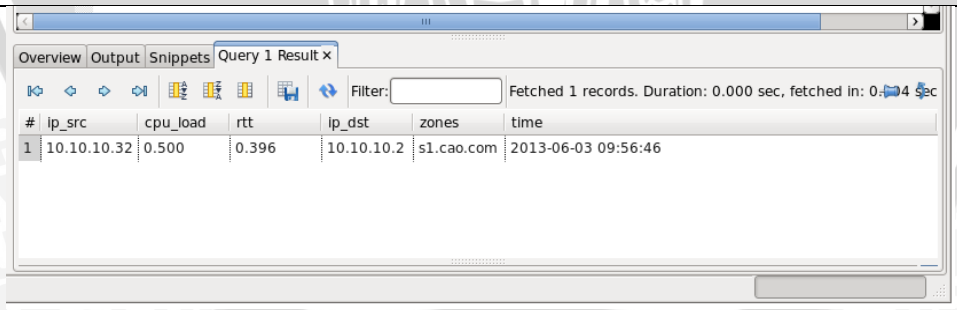
Gambar 5.2. *Database* Sistem Skenario Satu Percobaan Satu

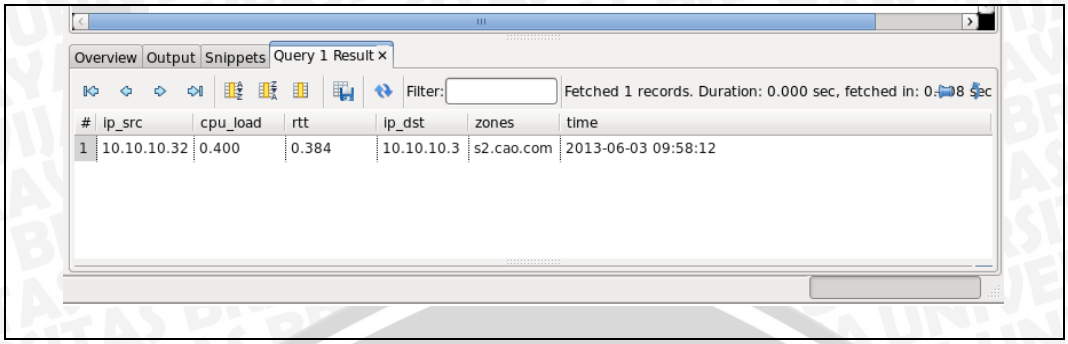
Terkadang sistem tetap memilih *content server* yang sedang mengalami *overload* penggunaan *CPU* saat pengukuran ulang kembali dilakukan. Hal itu dikarenakan saat program yang dijalankan untuk menghitung *CPU usage* tidak mencapai 80% dan memperoleh *rtt* yang lebih kecil dibandingkan *content server* lainnya, sedangkan program lain yang dijalankan untuk selalu menghitung *CPU usage* mendapatkan nilai diatas 80%. *Content server* akan tetap mengirim pesan kepada *load balancer* bahwa *content server* ini sedang mengalami *overload* penggunaan *CPU,* sehingga sistem akan tetap menghapus data alamat IP user yang mengakses pada *content server tersebut* dalam *database* sistem sehingga pengukuran ulang akan terus dilakukan hingga user diarahkan menuju *content server* yang benar. Kejadian ini terjadi pada saat skenario dua dilakukan, sistem tetap mengarahkan user menuju *content server* yang sedang mengalami *overload* penggunaan *CPU.* Log proses yang dihasilkan pada skenario ini dapat dilihat pada gambar 5.3.

```
===============
09:56:38

IP Client | CPU usage | RTT | IP server | Zone

10.10.10.32|0.5|0.528|10.10.10.4|s3.cao.com
10.10.10.32|0.5|0.396|10.10.10.2|s1.cao.com
10.10.10.32|0.4|0.485|10.10.10.3|s2.cao.com
```

```
===============
09:57:28

IP Client | CPU usage | RTT | IP server | Zone

10.10.10.32|2|0.327|10.10.10.3|s2.cao.com
10.10.10.32|72.5|0.291|10.10.10.2|s1.cao.com
10.10.10.32|96.5|0.257|10.10.10.4|s3.cao.com

===============
09:57:39

IP Client | CPU usage | RTT | IP server | Zone

10.10.10.32|0.6|0.436|10.10.10.3|s2.cao.com
10.10.10.32|71.6|0.298|10.10.10.2|s1.cao.com
10.10.10.32|90.2|0.186|10.10.10.4|s3.cao.com

===============
09:57:52

IP Client | CPU usage | RTT | IP server | Zone

10.10.10.32|0.4|0.298|10.10.10.3|s2.cao.com
10.10.10.32|78.4|0.242|10.10.10.2|s1.cao.com
10.10.10.32|99.9|0.251|10.10.10.4|s3.cao.com

===============
09:58:05

IP Client | CPU usage | RTT | IP server | Zone

10.10.10.32|0.4|0.384|10.10.10.3|s2.cao.com
10.10.10.32|81.1|0.546|10.10.10.4|s3.cao.com
10.10.10.32|89.2|0.472|10.10.10.2|s1.cao.com
```

Gambar 5.3. Log Pengukuran Sistem Skenario Dua Percobaan Dua





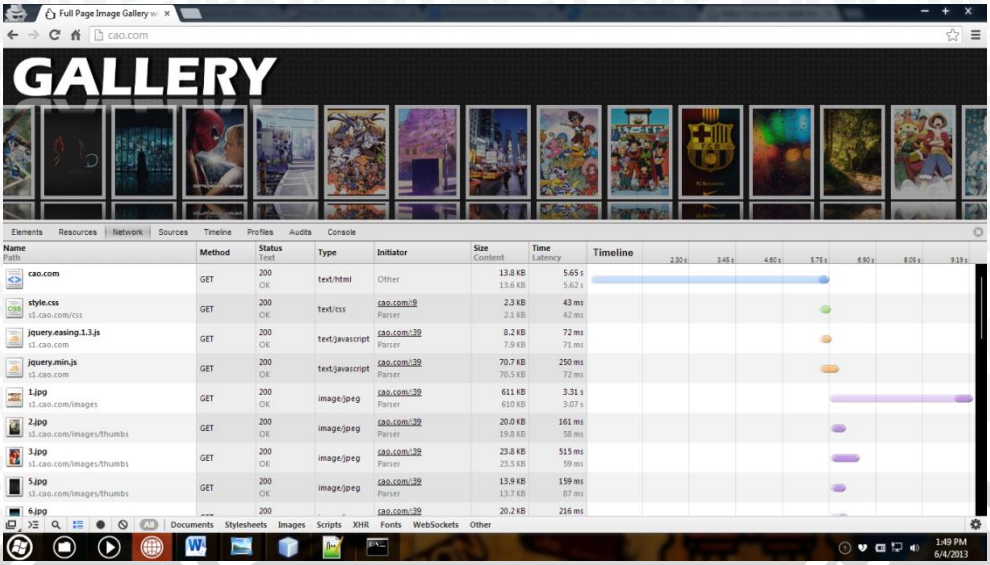
Gambar 5.4. *Database* Sistem Skenario Dua Percobaan Dua

5.2.2. Analisis Pengujian Non-fungsional

Hasil pengujian non-fungsional yang dilakukan yaitu untuk menguji ketersediaan *(availability)* dan kinerja penggunaan *CPU* pada *origin server*. Data yang diperoleh dari skenario satu dan dua untuk menguji ketersediaan sistem menunjukkan bahwa sistem dapat tetap melayani user apabila salah satu atau beberapa *content server* dalam keadaan mati atau tidak terhubung ke dalam jaringan. *Content server* yang tidak terhubung ke dalam jaringan akan diberi nilai yang besar oleh sistem agar sistem tidak akan mengarahkan user kepada *content server* tersebut. Log pengukuran sistem untuk pengujian non-fungsional skenario satu dan dua dapat dilihat pada gambar 5.5 dan 5.7. Hasil yang diperoleh oleh user pada *web browser* dapat dilihat pada gambar 5.6 dan 5.8.

```
===============
13:49:16

IP Client | CPU usage | RTT | IP server | Zone

10.10.10.32|99|99999|10.10.10.3|s2.cao.com
10.10.10.32|0.2|0.412|10.10.10.4|s3.cao.com
10.10.10.32|0.3|0.405|10.10.10.2|s1.cao.com
```

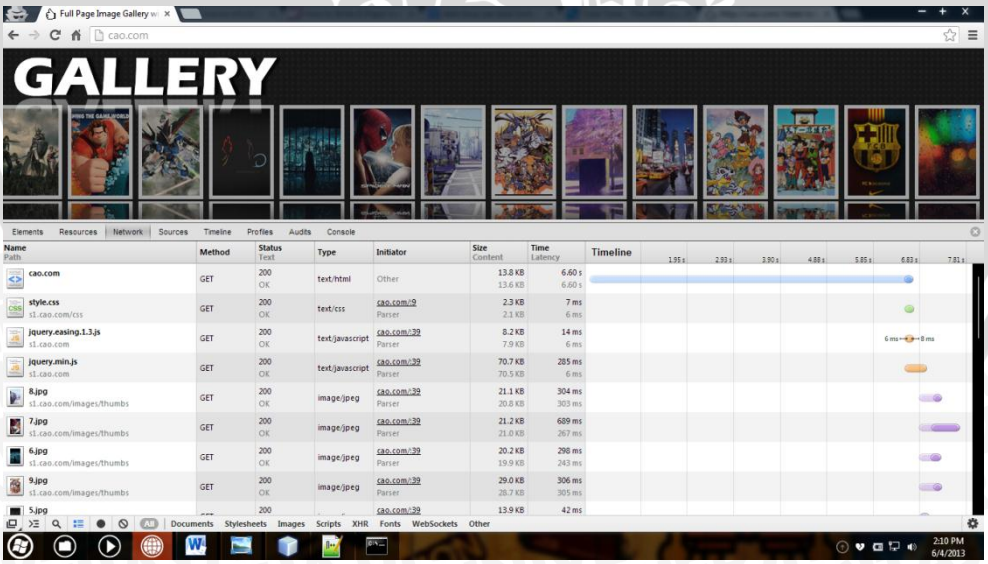
Gambar 5.5. Log Pengukuran Sistem Skenario Satu Percobaan Satu



Gambar 5.6. Tampilan *Web Browser* Skenario Satu Percobaan Satu

```
===============
14:10:12

IP Client | CPU usage | RTT | IP server | Zone

10.10.10.32|99|99999|10.10.10.3|s2.cao.com
10.10.10.32|99|99999|10.10.10.4|s3.cao.com
10.10.10.32|0.3|0.315|10.10.10.2|s1.cao.com
```

Gambar 5.7. Log Pengukuran Sistem Skenario Dua Percobaan Satu



Gambar 5.8. Tampilan *Web Browser* Skenario Dua Percobaan Satu

Pada pengujian skenario tiga dilakukan untuk membandingkan kinerja *origin server* dengan sistem *load balancing* dan *single server*. Parameter yang

dibandingkan adalah *CPU usage, Throughput,* dan *Response Time.* Untuk menguji *origin server* dengan sistem *single server,* konten yang diakses oleh user seluruhnya ditujukan kepada *origin server.*

Data *CPU usage* pada *origin server* dengan sistem *single server* menunjukkan angka yang lebih tinggi dibandingkan sistem *load balancing* yang dibangun. Hal ini disebabkan *origin server* dengan sistem *single server* memproses seluruh objek yang terdapat di dalam halaman web yang diakses oleh user. Untuk *origin server* dengan sistem *load balancing* hanya menanggung beban kerja pemrosesan teks pada halaman web tersebut karena konten lainnya tersebar di ketiga *content server.* Karena beban kerja *origin server* dengan sistem *single server* lebih berat, *throughput* yang dihasilkan user lebih kecil dibandingkan dengan sistem *load balancing* yang dibangun. Dengan *throughput* yang lebih besar maka *response time* yang dihasilkan akan lebih kecil karena user lebih cepat terlayani oleh sistem.

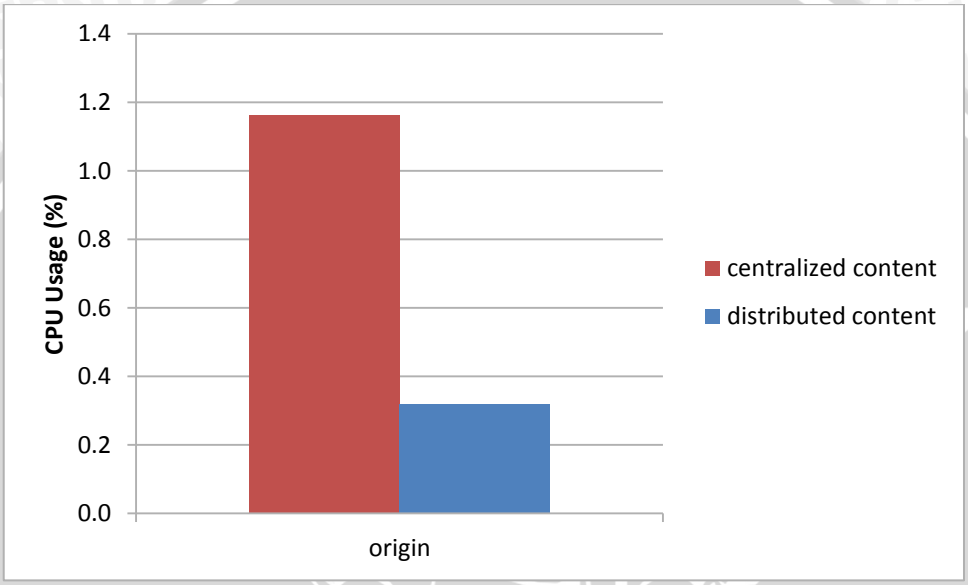
Pada sistem *single server* masih terdapat *error rate* yang bervariasi tidak seperti pada sistem *load balancing* hampir tidak terdapat *error.* Jumlah permintaan/menit pada sistem *load balancing* juga menunjukkan hasil yang lebih baik dengan sistem single server. Dengan jumlah permintaan rata-rata pada setiap user sebanyak 2.7/menit pada sistem *single server* masih terdapat *error* yang bervariasi, sedangkan dengan sistem *load balancing* yang dibangun mampu melayani jumlah permintaan sebanyak 8.8/menit dengan hampir tidak memiliki *error* sama sekali. Hal ini disebabkan *origin server* tidak mampu melayani seluruh permintaan user dengan baik karena beban yang ditanggung lebih banyak dibandingkan dengan sistem *load balancing* yang dibangun. Grafik perbandingan *CPU usage, throughput,* dan *response time* dapat dilihat pada gambar 5.10 hingga 5.14.

Overview | Output | Snippets | Query 1 Result ×

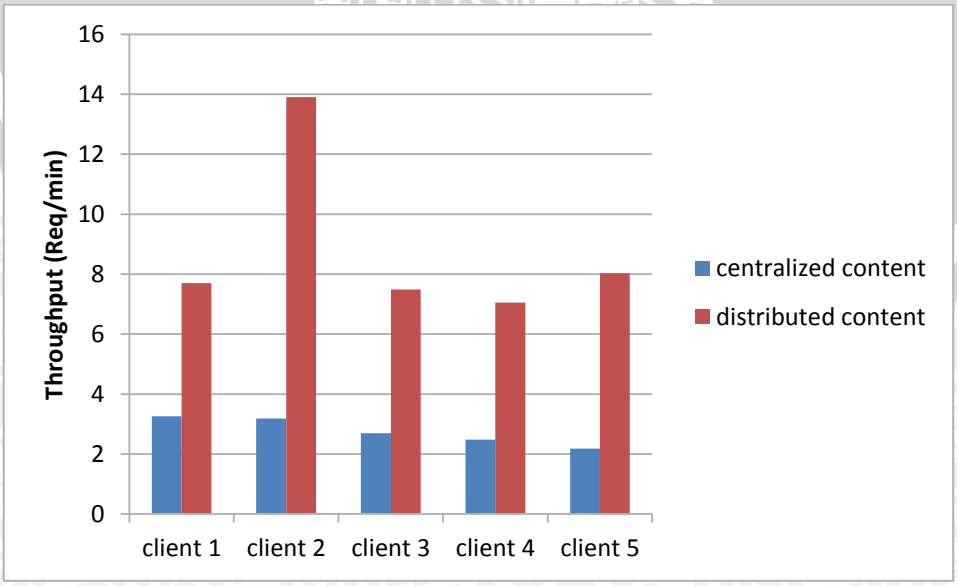
Filter: Fetched 5 records. Duration: 0.000 sec, fetched in: 0.006 sec

#	ip_src	cpu_load	rtt	ip_dst	zones	time
1	10.10.10.30	9.300	0.896	10.10.10.2	s1.cao.com	2013-07-01 21:27:47
2	10.10.10.34	0.700	0.799	10.10.10.3	s2.cao.com	2013-07-01 21:27:46
3	10.10.10.31	0.700	0.723	10.10.10.4	s3.cao.com	2013-07-01 21:27:47
4	10.10.10.32	0.700	1.096	10.10.10.3	s2.cao.com	2013-07-01 21:27:42
5	10.10.10.33	0.600	1.029	10.10.10.2	s1.cao.com	2013-07-01 21:27:47

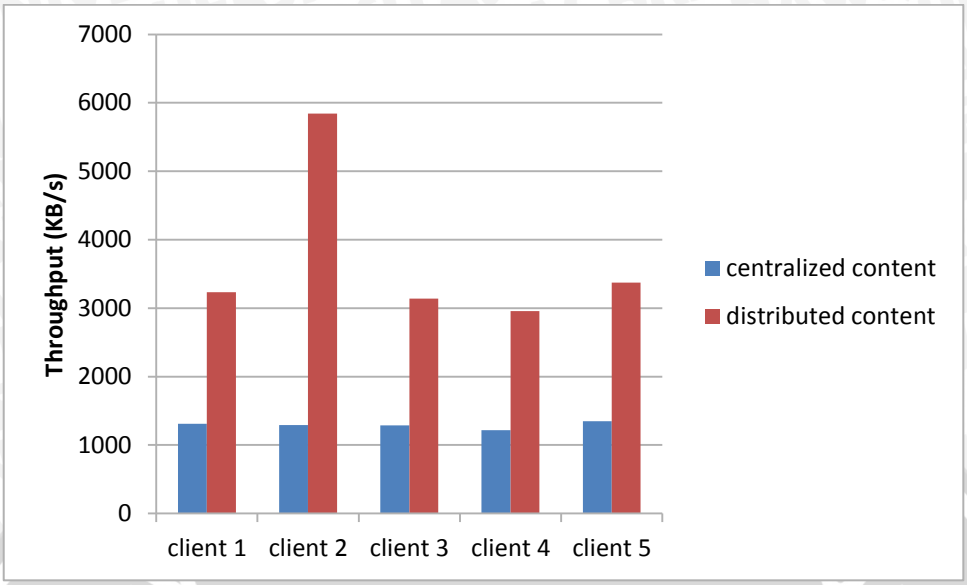
Gambar 5.9. Gambar Tujuan *Content Server* Masing-Masing User



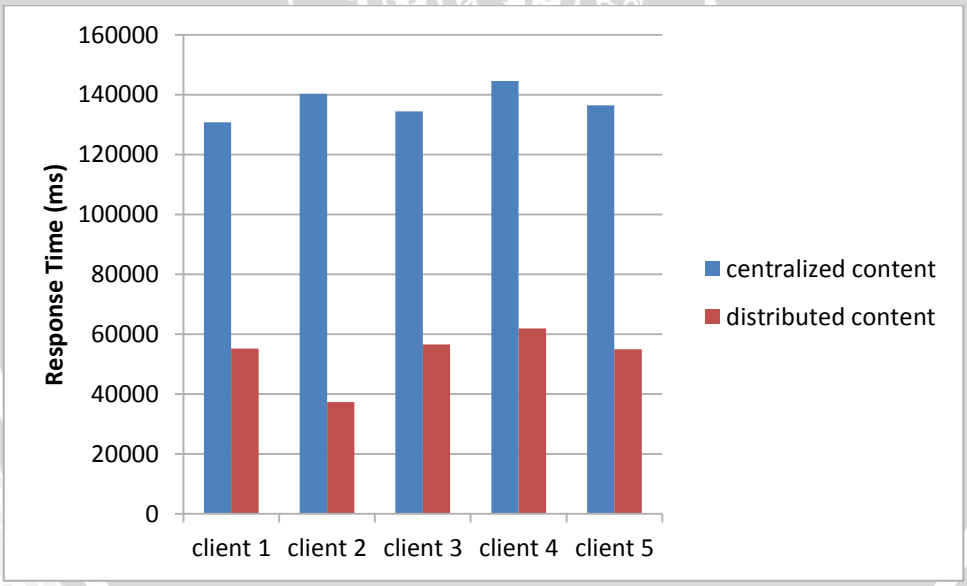
Gambar 5.10. Grafik Perbandingan *CPU usage origin server*



Gambar 5.11. Grafik Perbandingan jumlah permintaan/menit user



Gambar 5.12. Grafik Perbandingan *Throughput* User



Gambar 5.13. Grafik Perbandingan *Response Time* User

BAB VI
PENUTUP

Bab VI penutup berisi kesimpulan dari implementasi dan analisis mengenai sistem yang dibangun serta saran untuk pengembangan sistem ke depannya.

6.1. Kesimpulan

Dari hasil implementasi, pengujian dan analisis dari sistem *load balancing* yang dibangun, dapat disimpulkan bahwa:

- Sistem *load balancing* yang dibangun dapat melakukan pemilihan *server* tujuan berdasarkan perbandingan metrik waktu *ICMP request/reply* dan *CPU usage*. Dan mampu melakukan perhitungan metrik ulang terhadap user yang sedang mengakses server tujuan yang sedang mengalami *"overload"* secara otomatis sehingga user tidak akan diarahkan menuju *server* tersebut.
- Sistem *load balancing* yang dibangun dapat tetap melayani user apabila salah satu atau dua *content server* sedang *down* atau tidak terhubung ke dalam jaringan.
- Sistem *load balancing* yang dibangun dapat melakukan optimasi *CPU usage* pada *origin server* sebesar 72.6% terhadap layanan HTTP yang digunakan. Dengan optimasi *CPU usage* tersebut, sistem *load balancing* yang dibangun dapat memberikan layanan yang lebih cepat dan meminimalisasi *error* yang terjadi. Hal ini dibuktikan pada pengujian sistem *load balancing* yang dilakukan menunjukkan grafik *throughput* dan *response time* serta *error rate* yang lebih baik dibandingkan dengan sistem *single server*.

6.2. Saran

Saran yang dapat disampaikan penulis untuk pengembangan sistem *load balancing* dengan distribusi load pada konten halaman web adalah:

- Perlu melakukan kombinasi dengan algoritma *load balancing* yang sudah ada seperti *rr*, *wrr*, *lc* dan *wlc* pada *origin server* untuk mendapatkan hasil pembagian beban yang lebih seimbang.
- Perlunya mempertimbangkan metrik lain seperti *memory usage*, *disk usage*, dan *current process number* untuk menentukan *server* tujuan serta melakukan pelacakan terhadap permintaan koneksi user sehingga sistem tidak hanya melihat kondisi *server-server* dalam sistem saja, tetapi juga melihat kondisi koneksi user.

DAFTAR PUSTAKA

[BOU-01] Bourke, Tony.2001, "*Server* Load Balancing", O'Reilly Media, California.

[TZU-04] Tzung-Shi Chen dan Kuo-Lian Chen.2004."Balancing workload based on content types for scalable Web server clusters", *Advanced Information Networking and Applications*, Vol. 2, hal. 321-325.

[KON-07] Kontogiannis, Valsamidis, dkk.2007."An adaptive load balancing algorithm for cluster-based web systems".

[DTO-97] D. Andresen, T. Yang, dan O. H. Ibarra.1997."Toward A Scalable Distributed WWW Server On Workstation Clusters", *Journal of Parallel and Distributed Computing*, 42(1), hal. 91-100.

[HEO-00] H. Bryhni, E. Klovning, and O. Kure.2000, "A Comparison of Load Balancing Techniques For Scalable Web Servers", *Network, IEEE 14.4*, hal. 58-64.

[ZHA-03] Zhang, Wensong. 2003, "Linux Virtual *Server* for Scalable Network Services", *Ottawa Linux Symposium*, Vol. 2000.

[JOH-02] John Dilley, dkk. 2002, "Globally Distributed Content Delivery", *Internet Computing, IEEE* 6.5, hal. 50-58.

[HAD-04] Haddad, Ibrahim., dan Butler, Greg.2004, ”Experimental Studies of Scalability in Clustered Web Systems”, *Parallel and Distributed Processing Symposium,* hal. 185.

[JAM-07] James D. M. 2007, “Network Analysis, Architecture, and Design”, Morgan Kaufmann, Burlington.

[ANO-08] Anonim. 2008, “Choosing a Content Delivery Method”, http://www.webtorials.com/main/resource/papers/att/paper4/Choosing-Content-Delivery.pdf, diakses tanggal 23 Juni 2013.

[GER-09] Adam, Gerhard. 2009, “CPU Utilization: The Misunderstood Metric”, http://enterprisesystemsmedia.com/article/cpu-utilization-the-misunderstood-metric, diakses tanggal 21 Juli 2013.

[DOU-95] Douglas E. Comer. 1995, “Internetworking With TCPI/IP Vol I: Principles, Protocols, and Architecture”, Prentice Hall, New Jersey.

[EMI-08] Emily H. Halili. 2008, “Apache JMeter”, Packt Publishing, Birmingham.

[VSZ-11] Vijay K. A. 2011, “Where Do You ‘Tube’? Uncovering YouTube Server Selection Strategy”, *Computer Communications and Networks (ICCCN),* hal. 1-6.