ISSN : 2302-3805

IMPLEMENTASI TEKNOLOGI LOAD BALANCER DENGAN WEB SERVER NGINX UNTUK MENGATASI BEBAN SERVER

Effendi Yusuf [1)] , Tengku A Riza [2)] , Tody Ariefianto [3)]
[1,2,3)] *Fak Elektro & Komunikasi IT Telkom Bandung*
Jl. Telekomunikasi No.1 Ters Buah Batu Bandung
Email: effendiyusuf08@gmail.com [1)], tka@ittelkom.ac.id [2)], taw@ittelkom.ac.id [3)]

Abstrak

Kebutuhan akan koneksi Internet semakin meningkat seiring dengan berlimpahnya perangkat yang bisa dipakai untuk menjelajah di ranah maya. Web server merupakan penyedia layanan akses kepada pengguna melalui protokol komunikasi Hypertext Transfer Protocol (HTTP) atas berkas-berkas yang terdapat pada suatu situs web. Banyaknya permintaan pengguna terhadap web server atas berkas-berkas yang terdapat pada suatu situs web, sehingga dibutuhkan load balancer untuk mendistribusikan beban pengaksesan pada dua atau lebih jalur koneksi agar tidak berpusat ke salah satu perangkat jaringan saja.

Teknologi load balancer ini menggunakan web server Nginx. Nginx merupakan salah satu dari sebagian perangkat lunak untuk server. Kelebihan Nginx adalah karena performanya yang tinggi, stabil, memiliki banyak fitur, mudah dikonfigurasi, dan menggunakan sedikit sumberdaya pada server.

Dalam penelitian ini akan diimplementasikan sebuah jaringan yang terdiri dari, load balancer server, web server dan database server, menggunakan Virtual Private Server (VPS). Dengan hasilnya akan didapat sebuah jaringan yang dapat melayani berkas-berkas yang terdapat pada situs web dengan membagi-bagi beban pengaksesan yang datang ke beberapa server, jadi tidak berpusat ke salah satu perangkat jaringan saja.

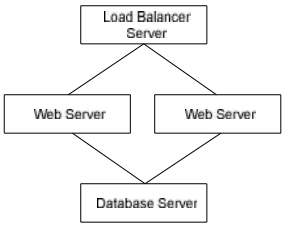
Kata kunci :
Web server, load balancer, Nginx, VPS

1. Pendahuluan

Web server merupakan software yang memberikan layanan data yang berfungsi menerima permintaan *Hypertext Transfer Protocol* (HTTP) dari client yang dikenal dengan *browser web* dan mengirimkan kembali hasilnya dalam bentuk halaman - halaman *website* yang umumnya menampilkan teks, gambar, animasi dan video.

Kebutuhan akan koneksi Internet semakin meningkat seiring dengan berlimpahnya perangkat yang bisa dipakai untuk menjelajah di dunia maya. *Web server* merupakan penyedia layanan akses kepada pengguna melalui protokol komunikasi HTTP atas berkas-berkas yang terdapat pada suatu situs web. Banyaknya permintaan pengguna terhadap *web server* atas berkas-berkas yang terdapat pada suatu situs web. Sehingga membuat *web server* akan bekerja lebih untuk me *load* berkas-berkas yang terdapat pada *website* tersebut. Hal ini akan menyebabkan menurunnya performansi di sisi server, yang mengakibatkan server tersebut tidak dapat melayani permintaan dari *user*. Berdasarkan pertimbangan diatas, penulis tertarik untuk membangun *network load balancing* yang terdapat sebuah *load balancer server* untuk dapat mendistribusikan beban trafik pada dua jalur koneksi yang terhubung dengan dua *web server* dan *database server*.

Network load balancing merupakan sistem yang dapat melayani beban pengaksesan yang besar dengan kemungkinan dapat meminimalisir kegagalan melayani *request* dari *user*. Perangkat lunak yang dipasang pada *load balancer server* tersebut adalah *Nginx*. Gambar 1 menunjukkan blok diagram dari sistem *network load balancing*.



Gambar 1 Diagram blok web server dengan load balancer.

2. Tinjauan Pustaka

2.1 Jaringan Komputer

Jaringan komputer adalah sekumpulan komputer yang terhubung satu dengan lainnya menggunakan protokol komunikasi melalui media komunikasi sehingga dapat menggunakan sumber daya bersama seperti *harddisk*, *printer*, dan sumber informasi lainnya[2].

Tujuan dibangunnya jaringan komputer adalah membawa suatu informasi secara tepat dan tanpa adanya kesalahan dari sisi pengirim menuju sisi penerima melalui media komunikasi. Sedangkan manfaat yang bisa didapat adalah: *Sharing Resource* yang bertujuan agar sumber daya yang ada dalam jaringan baik berupa program, perangkat keras atau lainnya dapat dimanfaatkan oleh orang yang sedang mengakses jaringan komputer tersebut[2].

Keamanan Data, jaringan komputer memberikan perlindungan bagi *user* untuk menyimpan data sehingga tidak sembarang orang bisa

ISSN : 2302-3805

mengaksesnya demikian juga untuk sumber-sumber daya lainnya[2].

2.1.1 Skala Jaringan Komputer

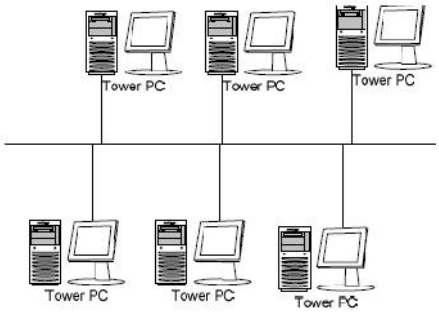
[2]Skala dalam jaringan komputer adalah cakupan layanan jaringan komputer yang ada. Jaringan komputer terdiri dari beberapa cakupan yaitu:

- Lokal ataupun LAN (*Local Area Network*)
- Luas ataupun WAN (*Wide Area Network*)
- Global atapun Internet

2.1.2 Topologi Jaringan Komputer

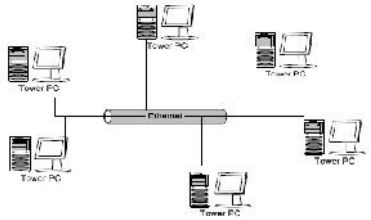
Dalam membentuk jaringan komputer harus dipertimbangkan bagaimana cara melakukan konektivitas antar komputer yang akan tergabung dengan jaringan komputer. Dari sisi dukungan teknologi terdapat beberapa topologi jaringan komputer yang popular yang digunakan dalam membentuk sebuah jaringan komputer yaitu[2]:

A. **Bus**, pada topologi ini tidak terdapat *repeater* setiap *host* melakukan *broadcast* ke *host* lainnya dan masing-masing menyimpan alamat *host* yang bisa dihubungi[2]. Gambar 2 merupakan contoh topologi jaringan bus



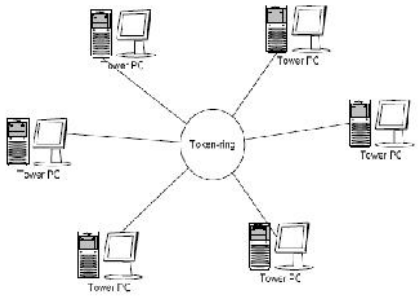
Gambar 2 Topologi jaringan bus[2]

B. **Star**, pada topologi ini terdapat sebuah *repeater* yang juga berfungsi untuk mengatur lalu lintas data antar *host* dan alamat setiap *host* disimpan dalam *repeater* ini setelah setiap *host* melakukan *broadcast* ke *repeater* untuk mengenalkan diri[2]. Gambar 3 merupakan contoh topologi star.



Gambar 3 Topologi jaringan star[2]

C. **Ring**, pada topologi ini struktur pengkabelan bersifat tertutup dan didalamnya tidak terdapat sebuah repeater, setiap host melakukan broadcast ke jaringan untuk mengenalkan diri dan setiap host yang akan melakukan koneksi dengan host lain mengunjungi setiap titik host yang dilewati sampai host yang di tuju di temukan[2]. Gambar 4 merupakan contoh penggunaan topologi jaringan ring.



Gambar 4 Topologi jaringan ring[2]

2.2 Web Server

Web Server adalah software server yang menjadi tulang belakang dari *World Wide Web* (WWW)[6]. *Web server* menunggu permintaan dari client yang menggunakan *browser* seperti *Netscape Navigator*, *Internet Explorer*, *Mozilla Firefox*, dan program *browser* lainnya[6]. Jika ada permintaan dari *browser*, maka *web server* akan memproses permintaan itu dan kemudian memberikan hasil prosesnya berupa data yang diinginkan kembali ke *browser*.

Web server, untuk berkomunikasi dengan *client*nya (*web browser*) mempunyai protokol sendiri yaitu HTTP. Dengan protokol ini, komunikasi antar *web server* dengan *client*nya (*browser*) dapat saling dimengerti dan lebih mudah.

2.2.1 Nginx Web Server

Nginx (baca: engine x) adalah server *HTTP* dan *reverse proxy* gratis berbasis open-source, yang dapat juga digunakan sebagai proxy IMAP/POP3. Perangkat lunak ini diciptakan oleh Igor Sysoev pada tahun 2002, dan dirilis untuk pertama kalinya secara umum pada tahun 2004. Saat ini Nginx digunakan oleh 7.65% (22.8juta) nama domain di seluruh dunia[1].

Nginx terkenal karena performanya yang tinggi, stabil, memiliki banyak fitur, mudah dikonfigurasi, dan menggunakan sedikit sumber daya pada server.

Nginx adalah salah satu dari sebagian perangkat lunak untuk server yang diciptakan untuk mengatasi masalah C10K. Dalam masalah ini, C10K adalah ketika *web server* diminta untuk menangani sepuluh ribu *client* secara bersamaan. Tidak seperti perangkat lunak *server* yang lainnya, Nginx tidak bergantung kepada *thread* untuk melayani *client*. Sebaliknya, Nginx menggunakan arsitektur *asynkronus* yang lebih stabil. Arsitektur ini membutuhkan lebih sedikit memori, dan yang lebih penting, dapat diperkirakan.

Bahkan jika anda tidak mengharapkan *server* anda untuk mengatasi ribuan koneksi pada saat yang bersamaan, anda masih dapat diuntungkan dengan pemakaian memori yang sedikit namun berkemampuan tinggi. Nginx dapat digunakan dalam semua skala: mulai

ISSN : 2302-3805

dari VPS kecil sampai dengan *cluster server* dalam jumlah besar.

Nginx digunakan oleh beberapa website ternama seperti: *WordPress, Hulu, Github, Ohloh, SourceForge dan TorrentReactor*.

2.3 Load balancing

Load balancing adalah suatu metode untuk mendistribusikan beban kepada beberapa *server* sehingga beban kerja di sisi *server* menjadi lebih ringan. Agar waktu rata-rata pengerjaan tugas akan rendah dan ketersediaan layanan yang sangat tinggi[2].

Pada saat sebuah *single server* melayani permintaan dari *user*, maka sebenarnya server tersebut sedang terbebani karena harus melakukan proses permintaan dari *user*. Jadi jika user banyak yang melakukan permintaan maka proses yang terjadi disisi *server* akan sangat tinggi. Jika satu *server* saja terbebani, tentu server tersebut tidak dapat melakukan proses permintaan dari user karena *server* melakukan proses ada batasnya.

Solusi yang paling ideal menggunakan *load balancer*. *Load balancer* adalah perangkat jaringan yang dipasang diantara *client* dan *server. Load balancer* dapat menggunakan beberapa metode penjadwalan yang menentukan permintaan *user* akan diteruskan ke server mana.

2.4 Virtual Private Server

Virtual Private Server (VPS) adalah teknologi virtualisasi dimana anda bisa memiliki sebuah *server virtual* yang resource *Central processing unit* (CPU) , *Random-access memory* (RAM), dan *Storage*nya dialokasikan secara pasti tanpa harus memiliki *server* secara fisik. Teknologi ini memungkinkan Anda untuk memiliki akses *root* dan meng *custom server* sesuai dengan kebutuhan anda. Tentu dengan biaya yang jauh lebih murah dibandingkan jika anda menyewa *dedicated server*[7].

Dalam layanan VPS ini, satu server induk dengan spesifikasi yang tinggi (*Dual Processor – Multiple Core*) dibagi-bagi resourcenya menjadi beberapa *server virtual* (*Virtual Private Server* / VPS) dimana antar *server virtual* bekerja secara bersamaan, bahkan dengan *Operating system* (OS) yang berbeda beda tanpa adanya kemungkinan untuk saling mengganggu satu sama lain.

3. Metode Penelitian

1. Studi literatur

Pada tahap ini akan dilakukan studi literatur terhadap materi-materi yang terkait dengan topik penelitian melalui referensi yang berhubungan dengan *load balancer* menggunakan *web server Nginx.*

2. Konsultasi dan diskusi

Selain studi literatur, penulis juga berkonsultasi dan berdiskusi dengan orang yang ahli dalam bidang sistem jaringan komputer.

3. Perancangan

Perancangan yang dimaksudkan untuk memperoleh hasil yang baik.

4. Pengujian

Melakukan pengujian dan pengukuran performansi *network load balancing.*

5. Penyusunan Laporan

Setelah dilakukan konfigurasi dan realisasi, hasil pengukuran performansi *load balancer* yang didapat ditulis dalam bentuk laporan.

4. Hasil dan Pembahasan

Pada implementasi penelitian ini dilakukan pengujian sistem pada saat *web server* dalam kondisi gangguan dan pengukuran parameter yang diukur pada implementasi penelitian ini meliputi waktu respon, *throughput, request loss* dan jumlah *request* per detik. Pada pengukuran ini menggunakan perangkat lunak httperf, untuk membangkitkan *request* secara simultan.

4.1 Pengukuran Performansi Sistem

4.1.1 Pengukuran Performansi Single Server

Dalam pengukuran performansi dengan *single server* dilakukan pengamatan menggunakan satu VPS (*virtual private server*) yang menyediakan layanan *website* yang telah dipasang perangkat lunak *web server Apache* dan perangkat lunak *MySQL* sebagai *database*. Dengan menggunakan perangkat lunak *httperf* disisi *client*, *request* seolah dibangkitkan oleh beberapa *client* secara simultan, dimana jumlah request yang dibangkitkan mencapai 3000 *request.* Tabel 1 dibawah ini merupakan hasil pengukuran yang meliputi paramater-paramater yang telah dilakukan pengukuran sebanyak 10 kali pengambilan data.

Tabel 1 Hasil pengukuran dengan Single Server

Request yang dibangkitkan	Request yang dilayani per detik	Waktu Respon (ms)	Throughput (Kbps)	Request Loss (%)
500	6,25	18613,83	111,74	2,72
1000	2,37	58808,28	36,99	33,1
1500	3,59	59591,65	40,54	56,6
2000	1,63	216250,2	16,98	68,1
2500	2,6	98866,6	20,51	70,7
3000	3,25	99598,33	27,76	71,8

4.1.2 Pengukuran Performansi *Network Load Balancing*

Pada pengujian ini dilakukan implementasi *network load balancing* dengan menggunakan dua buah *web server*, dimana terdapat *load balancer server* diantara *client* dan *web server*. Dimana *load balancer server* bertugas meneruskan *request* dari *client* kepada

ISSN : 2302-3805

kedua *web server* tersebut. Pembangkitan *request* dilakukan dengan cara yang sama pada pengukuran *single server* yang dibangkitkan mencapai 3000 *request* secara bertahap dapat dilihat pada tabel 1 diatas. Tabel 2 berikut merupakan hasil pengukuran dari 10 kali pengukuran.

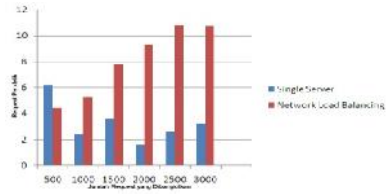
Tabel 2 Hasil pengukuran dengan *Network Load Balancing*

Request yang dibangkitkan	Request yang dilayani per detik	Waktu Respon (ms)	Throughput (Kbps)	Request Loss (%)
500	4,43	27394,02	64,89	8,2
1000	5,27	33716,95	43,21	14,3
1500	7,82	29699,97	34,64	11,8
2000	9,33	25299,79	34,64	14,6
2500	10,82	21273,28	37,94	18,4
3000	10,73	20588	38,47	24,4

4.2 Perbandingan Antara Sistem *Single Server* dan *Network Load Balancing*

4.2.1 Jumlah *Request* Perdetik

Jumlah *request* perdetik adalah kemampuan *server* untuk melayani beberapa *client* dalam satu detik. Tujuan pengukuran yaitu untuk mengetahui berapa jumlah *request* yang dapat dilayani sistem dalam satu detik.



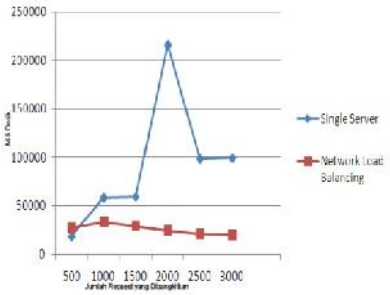
Gambar 5 Perbandingan *Single Server* dan *Network Load Balancing* Berdasarkan Parameter Jumlah *Request* Perdetik

Dari Gambar 5 diatas dapat diketahui saat jumlah request dari *client* 500 *request*, *single server* lebih banyak melayani *request* dari *client* dari pada *network load balancing* karena hanya menggunakan satu *node*, jadi jarak tempuh paket dari *client* ke *web server* semakin dekat. Namun saat jumlah request dari client melebihi 1000 *request*, *network load balancing* lebih banyak melayani request dibandingkan *single server*. Hal ini disebabkan karena kemampuan *single server* mulai berkurang ketika 1000 *request* dibangkitkan. Sedangkan pada *network load balancing* semakin banyak request yang dibangkitkan, semakin banyak jumlah *request* dari *client* yang dapat dilayani dalam satu detik.

4.2.2 Waktu respon

Waktu respon adalah lama waktu yang dibutuhkan *server* dalam melayani setiap paket yang datang dengan satuan mili detik. Tujuan pengukuran waktu respon yaitu untuk mengetahui kecepatan sistem dalam melayani setiap paket yang datang dari *client*. Pada gambar 6 menunjukkan hasil pengukuran waktu respon

.



Gambar 6 Perbandingan *Single Server* dan *Network Load Balancing* berdasarkan parameter waktu respon

Berdasarkan gambar 6 diatas, data hasil pengukuran waktu respon yang didapat, diketahui ketika *request* dari *client* 500 *request*, *single server* lebih cepat dalam melayani setiap paket yang datang dibandingkan dengan *network load balancing*. Hal ini disebabkan karena single server tidak menggunakan load balancer. Jadi request dari *client* langsung dilayani oleh *single server*, oleh karena itu waktu respon yang didapat sangat cepat. Namun ketika jumlah *request* yang dibangkitkan melebihi 500 *request* waktu respon *network load balancing* lebih cepat dibandingkan *single server*. Hal ini disebabkan karena kemampuan *single server* untuk melayani *request* lebih dari 500 *request* mulai berkurang. Selain itu waktu respon yang lama juga disebabkan karena terjadi antrian paket data pada *single server* sedangkan pada *network load balancing* hal itu tidak terjadi karena adanya pembagian tugas untuk melayani *request* oleh *load balancer server*.

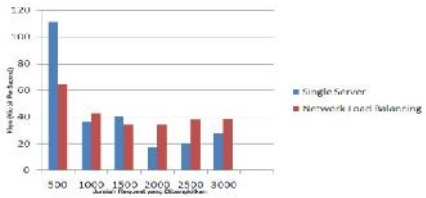
Kelebihan *network load balancing* dalam melayani *request* yang datang dipengaruhi adanya *load balancer server*. *Load balancer* memiliki algoritma penjadwalan *Round Robin* sederhana sehingga membagi beban secara bergiliran dan berurutan dari *web server* ke *web server* lain. Pembagian beban secara bergiliran dan berurutan menyebabkan *request* yang datang tidak menumpuk pada satu bagian *web server* saja sehingga antrian paket pada *server* dapat diminimalisir.

Kedua *web server* bekerja sama untuk menyediakan layanan *website* sehingga permintaan yang datang dari *client* bisa cepat dilayani dan kerja sistem tidak berat.

4.2.3 *Throughput*

Untuk mengetahui kemampuan sistem dalam memberikan layanan secara benar dari *client* saat meminta layanan yang datang secara bersamaan perlu dilakukan pengukuran *throughput* pada sistem.

ISSN : 2302-3805



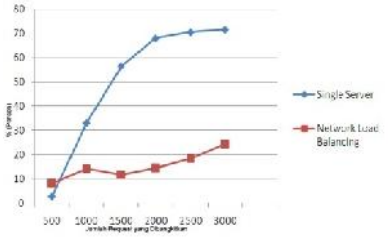
Gambar 7 Perbandingan Single Server dan Network Load Balancing Berdasarkan Parameter Throughput

Dari Gambar 7, secara keseluruhan nilai *throughput* pada *network load balancing* lebih baik dibandingkan *single server*. Hal ini dikarenakan *request* yang datang akan dikerjakan oleh dua *web server* sehingga dapat mengurangi terjadinya *bottleneck* disisi *server.*

Dari Gambar 7 juga dapat diketahui bahwa dengan *single server* dan *network load balancing* kemampuan sistem hanya terbatas pada saat jumlah request yang dibangkitkan 500 *request*. Kemudian lebih dari 500 *request* nilai *throughput* akan turun naik sehingga grafik tidak linear. Hal ini 500 *request* adalah batas maksimal dari kemampuan yang dimiliki sistem untuk memberi layanan secara benar terhadap *request* yang datang.

4.2.4 Request Loss

Tujuan pengukuran *request loss* adalah untuk mengetahui besar *request* yang hilang ketika dikirimkan kepada sistem. Dalam hasil pengukuran, semakin kecil *request loss* maka kehandalan sistem semakin tinggi.



Gambar 8 Perbandingan *Single Server* dan *Network Load Balancing* Berdasarkan *Parameter Request Loss*

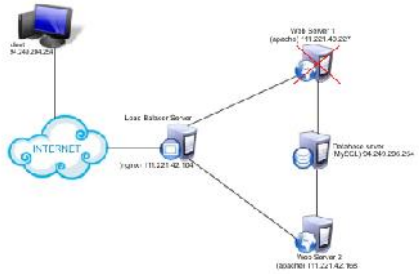
Dari gambar 8 menunjukkan bahwa ketika *request* yang dibangkitkan dari *client* semakin banyak, tidak sebanding dengan dengan kemampuan *server* dalam memberikan layanan menyebabkan terjadinya antrian paket disisi *server*, sehingga paket dalam antrian yang belum dilayani dan melebihi *time out* yang ditentukan akan dianggap gagal terkirim. *Single server* akan memiliki prosentase *request loss* yang lebih besar jika dibandingkan dengan *network load balancing*. Hal ini disebabkan kemampuan *single server* sangat terbatas karena semua *request* untuk melayani *client* dikerjakan dengan sendiri. Berbeda dengan *network load balancing*, karena *network load balancing* membagi beban yang datang kepada dua *web server*.

Jumlah request yang hilang pada sistem *network load balancing* lebih kecil dibandingkan *single server* yaitu 8,2 – 24,4 %. Yaitu ketika jumlah *request* yang dibangkitkan 500 *request*, mempunyai jumlah *request* yang hilang paling sedikit. Dan setelah itu nilai *request loss* naik turun dan tidak linear karena ketika jumlah *request* dibangkitkan lebih dari 500 *request* melebihi batas kemampuan sistem dalam memberikan layanan.

Pada pengamatan hasil pengukuran *single server*, saat *request* yang dibangkitkan lebih dari 500 *request*, maka jumlah *request* yang hilang pada grafik semakin naik secara linear. Hal ini disebabkan karena *single server* mengalami *overload* yang menyebabkan munculnya antrian yang sangat panjang sehingga *request* yang tidak terlayani semakin banyak.

4.3 Pengujian Saat *Web Server* dengan Simulasi Gangguan

Pada pengujian ini dilakukan ketika salah satu *web server* mati atau *link* putus, sistem masih dapat melayani *request* dari *client.* Pengujian ini dilakukan dengan cara mematikan *service apache* salah satu *web server*.Pada gambar 9 menunjukkan skenario pengujian gangguan yang akan diuji.



Gambar 9 Pengujian Jika Salah Satu *Real Server* Mengalami Gangguan

Load balancer server yang berperan untuk meneruskan *request* layanan dari *client* ke *web server*, akan mengarahkan ke *web server* mana *request* dari *client* akan dilayani. Sehingga ketika salah satu *web server* mengalami kegagalan maka *load balacer server* akan melimpahkan *request* pelayanan kepada *web server* yang masih aktif. Dimana di halaman index.php pada *web server* digunakan untuk melihat, *website* yang sekarang diakses dilayani oleh *web server* mana.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Dari hasil implementasi penelitian ini serta pengambilan data pengukuran dan pengujian mengenai implementasi *network load balancing*, maka dapat diambil kesimpulan sebagai berikut :

1. Kemampuan sistem *network load balancing* dalam melayani *request* lebih besar dibandingkan dengan *single server*. Sistem *network load balancing* mampu melayani *request* maksimal 10,82 per detik sedangkan pada *single server* hanya mampu melayani maksimal 6,25 *request* per detik sehingga pada sistem *network load balancing* memiliki *throughput* yang lebih bagus dibandingkan *single server*.

ISSN : 2302-3805

2. Waktu *respon sistem network load balancing* lebih cepat saat jumlah *request* dibangkitkan 1000 adalah 33716,95 mili detik sedangkan waktu respon untuk sistem *single server* yaitu 58808,28 mili detik sehingga antrian *request* yang terjadi pada sistem *network load balancing* tidak mengalamai antrian panjang dibandingkan *single server*.
3. Faktor-faktor penyebab *request loss* dapat diminimalisir oleh sistem *network load balancing*. Jumlah *request* yang hilang pada sistem *network load balancing* lebih kecil dibandingkan *single server* yaitu 8,2 – 24,4 %.
4. Ketika *request* yang dibangkitkan dari *client* semakin banyak, tidak sebanding dengan dengan kemampuan *server* dalam memberikan layanan menyebabkan terjadinya antrian paket disisi *server*, sehingga paket dalam antrian yang belum dilayani dan melebihi time out yang ditentukan akan dianggap gagal terkirim.
5. Menggunakan sistem *network load balancing* mempunyai ketersediaan layanan yang tinggi dibandingkan dengan *single server*.
6. Pada sistem *network load balancing*, kedua *web server* bekerja sama untuk menyediakan layanan *website* sehingga *request* yang datang dari *client* bisa cepat dilayani dan kerja sistem tidak berat.

5.2 Saran

Beberapa saran yang dapat diberikan guna pengembangan lebih lanjut antara lain:

1. Perlu dilakukan implementasi dan penelitian lebih lanjut dengan konfigurasi jaringan menggunakan alamat *IPv6*
2. Perlu adanya sebuah implementasi dan penelitian keamanan sistem terhadap network load balancing.

Daftar Pustaka

[1] *http://www.sherin.co.in/openxcluster121/* tanggal terakhir mengakses 30 April 2012.
[2] *Rijayana, Iwan (2005). Teknologi Load Balancing Untuk Mengatasi Beban Server. From http://journal.uii.ac.id/index.php/Snati/article/view/1385/1165, 10 Oktober 2011.*
[3] *Nedelcu, Clément. 2010. Nginx HTTP Server. Birmingham: Packt Publishing Ltd*
[4] *http://linux.die.net/man/1/httperf.* Tanggal terakhir mengakses 31 April 2012.
[5] Andrew S. Tanenbaum, Computer Network, 3rd Edition, Prentice Hall, New Jersey, 1996
[6] Anita Sesar Ria, Pengenalan PHP, *form http://my.cic.ac.id/stmik/files/blogfile/2006104002-090422084513.pdf*. Tanggal terkahir mengakses 25 Desember 2012.
[7] http://www.rumahweb.com/vps-indonesia. Tanggal tanggal terakhir mengakses 25 Desember 2012

Biodata Penulis

Tengku A Riza, memperoleh gelar Sarjana Teknik (S.T), Program Studi Teknik Elektro USU, lulus tahun 2002. Tahun 2008 memperoleh gelar Magister Teknik Elektro (M.T) dari Program Teknik Elektro Telekomunikasi IT Telkom. Saat ini sebagai Staf Pengajar program D3 Teknik Telekomunikasi (D3-TT) IT Telkom, Bandung.

Tody A, memperoleh gelar Sarjana Teknik (S.T), Program Studi Teknik Telekomunikasi ST Telkom, lulus tahun 2006. Tahun 2009 memperoleh gelar Magister Teknik Elektro (M.T) dari Program Teknik Elektro ITB. Saat ini sebagai Staf Pengajar program S1 Teknik Telekomunikasi (S1-TT) IT Telkom, Bandung