Implementasi *Computer Cluster* Berbasis *Open Source* untuk Penyeimbang Beban Sistem dan Jaringan Komputer

Yudhi Arta

Program Studi Teknik Informatika, Fakultas Teknik, Universitas Islam Riau
Jl. Kaharuddin Nasution 113 Perhentian Marpoyan, Pekanbaru 28284, Riau
Telepon: (0761) 72126, Fax: (0761) 674834, Website: http://www.uir.ac.id
yudhiarta@eng.uir.ac.id

Abstract **– The importance of an information currently required a system capable of providing the best service, especially information service with the *website*s media. Increasing the amount of access to the *website* resulting into increased workload *web server* and cannot be solved with a *single server*. This problem can be resolved by applying the *cluster load balance* method. Basically *cluster load balance* work by sharing *web server* workload is distributed to multiple *server* nodes so that the *website*s become balanced. Weight round robin scheduling algorithm can balance the load by determining the amount of weight to each *server* node. With the *load balance* by using a scheduling algorithm is expected to prevent the *overload request*s, workload leveling *web server*, and speed up the response time and throughput of the performance *web server*s.**

Keyword **– *Load balance*, Linux Virtual Service, Weight Round Robin, Respond Time, Throughput.**

I. PENDAHULUAN

Pada saat ini perkembangan teknologi berkembang dengan sangat pesat, hal ini dibuktikan dengan cepatnya perkembangan sebuah informasi. Informasi sudah menjadi keperluan pokok oleh berbagai bidang, baik instansi pemerintahan, institusi pendidikan, maupun perusahaan-perusahaan. Mengingat pentingnya informasi tersebut, maka dituntut agar dapat bisa diakses di mana saja dan kapan saja, yaitu dengan menggunakan jasa internet sebagai sarana penyedia informasi. Jasa internet yang umum digunakan yaitu World Wide Web (WWW) atau sering dikenali dengan web.

Pentingnya sebuah media informasi mengakibatkan meningkatnya jumlah pengunjung yang mengakses *website* setiap harinya, serta masih menggunakan *single server* dalam penyedia layanan tersebut maka faktor inilah menjadi faktor sistem menjadi *overload* dan crash terhadap *request*, serta performansi kinerja *website* menjadi berkurang. Dari dampak tersebut maka mengakibatkan beban pada *server* menjadi meningkat dan waktu respon pada layanan tersebut menjadi lambat. Sehingga dengan adanya permasalahan tersebut, maka perlu diperhatikan cara untuk mengatasi beban *server* tersebut serta peningkatan ketersediaan dan meminimalkan waktu tanggap dari *web server* tersebut.

*Cluster*ing merupakan teknik di mana dua atau lebih *web server* dikelompokkan bersama sebagai sebuah *cluster* yang mengakomodasi peningkatan beban. Dengan adanya permasalahan di atas, maka perlu adanya *cluster*ing untuk mengatasi masalah tersebut, yaitu dengan menerapkan metode *load balancing* pada *web server*, maka beban *server* akan dibagi dengan rata sesuai dengan jumlah *server* yang telah ter*cluster*.[2]

Dengan adanya permasalahan di atas, maka perlu adanya *cluster*ing untuk mengatasi masalah tersebut, yaitu dengan menerapkan metode *load balancing* pada *web server*, maka beban *server* akan dibagi dengan rata sesuai dengan jumlah *server* yang telah ter*cluster*. Pemahaman terhadap *load balance* juga banyak digunakan, yaitu jurnal internasional oleh Srivastava et al tentang *load balancing* menggunakan *high performance computing cluster* programming, dan Mei Lu Chin et al (2012) tentang beban efisiensi berbasis *load balancing* untuk *bursty* lalu lintas aplikasi *web*, serta Jiani Guo et al tentang *load balancing* di *cluster* berbasis *web server* untuk aplikasi multimedia.[13]

II. METODE PENELITIAN

A. Metode Penelitian

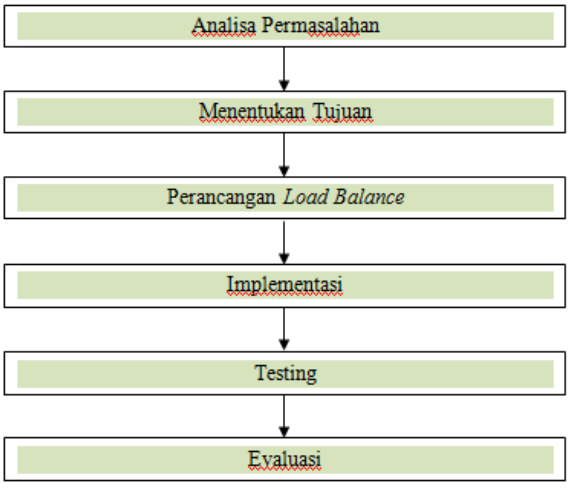
Tahapan kerja penelitian ini akan diuraikan pada kerangka kerja, yang mana kerangka kerja tersebut akan digunakan sebagai acuan atau langkah dalam membangun *load balance cluster* ini. Pada awalnya yang akan dilakukan yaitu melakukan analisis terhadap kebutuhan membangun *load balance* ini. Selanjutnya akan dilakukan dengan merancang dan menguji dari *load balance* itu sendiri serta mengetahui hasil dari pengujian penelitian ini yang nantinya akan dijadikan perbandingan.

Dalam metodologi penelitian ini dapat diketahui bahwa permasalahan yang terjadi pada institusi yang diteliti yaitu masalah terhadap ketidakefisienan *resource* yang ada,

khususnya *server* di dalam pelayanan terhadap informasi kampus dengan media *website*. Dengan adanya masalah tersebut, maka metodologi yang dapat digunakan untuk mengatasi kelemahan tersebut yaitu dengan mengimplementasikan *load balance cluster*. Penggunaan metode *Load balance cluster* ini menggunakan beberapa *node* untuk membantu dalam penyedia tersebut, sehingga kinerja dari node-node *cluster* dibagi secara merata dengan menggunakan algoritma penjadwalan dan tidak tergantung pada *web server* tunggal lagi.

Pada bab ini, metode penelitian yang akan digunakan pada tesis ini adalah studi pustaka, yang mana dengan melakukan penelitian dengan cara mencari sumber literatur, baik bersumber dari jurnal-jurnal, buku-buku, maupun artikel-artikel yang berhubungan dengan masalah serta perancangan dan implementasi pada bahan yang dibahas sesuai dengan yang akan diteliti pada penelitian ini. Pada penelitian ini juga akan menerapkan studi lapangan, yang mana akan dilakukan pengambilan data pada objek penelitian yang nantinya akan dipadukan serta dibandingkan dengan konsep *Load balancing* yang dipelajari dalam studi pustaka.

Untuk hasil akhir dari penelitian yang didapat yaitu mengimplementasikan *load balance* terhadap kinerja dan performansi dari *web server*, khususnya pada tempat objek penelitian yang akan diteliti. Pada implementasi ini ditujukan agar resource yang ada mampu mengatasi kelemahan pelayanan informasi dengan media *online* atau *website*. Selain itu dapat mengetahui bagaimana sistem *load balance* ini bekerja dengan baik, terutama menggunakan algoritma Weight Round Robin (WRR). Dengan algoritma Weight Round Robin, nantinya dapat diketahui apakah pembagian beban sesuai dengan parameter yang diberikan. Setelah itu kita akan mendapatkan hasil sebagai nilai waktu respon dan *throughput* perbandingan apakah kinerja *cluster load balance* lebih baik dari kinerja dengan *server* tunggal.



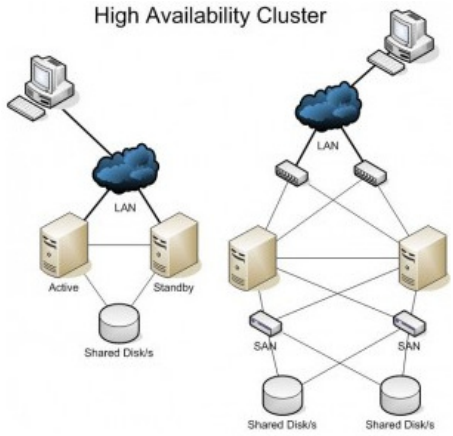
Gambar 1. Kerangka Kerja Penelitian

B. *High Availability Cluster*

High Availability Cluster (lebih dikenal sebagai HA *Cluster* atau Failover *Cluster*) adalah *cluster* komputer yang diimplementasikan dengan tujuan utama untuk meningkatkan availabilitas layanan yang disediakan *cluster* tersebut (Muhammad Taufik Saenal, 2010). High availability *cluster* digunakan untuk meningkatkan ketersediaan layanan yang disediakan oleh *cluster* tersebut.

High availability cluster menggunakan banyak komputer, yang mana komputer tersebut digunakan sebagai penyedia layanan ketika sistem pada salah satu komputer sedang mengalami *crash*. Ukuran yang paling kecil untuk membangun *high availability* ini yaitu 2 *node*, yang mana *node* tersebut akan digunakan untuk melakukan redundansi nantinya menghilangkan kegagalan di satu titik (*single point of failure*). [3]

Prinsip kerja *high availability cluster* ketika *server* utama gagal menyediakan layanan *service*, maka *server* yang lain yang sudah ter-*cluster* akan menggantikan tugas dari *server* utama tersebut secara otomatis. Dengan demikian, dengan adanya *high avalibilty* maka akan mengatasi permasalahan tersebut, khususnya di dalam meningkatkan ketersediaan data[3].

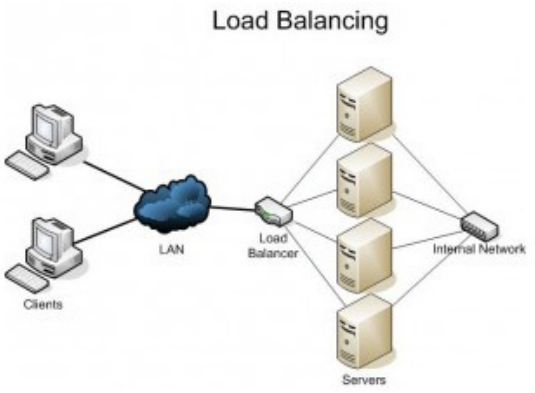


Gambar 2. HA *Cluster*

C. *Load balance Cluster*

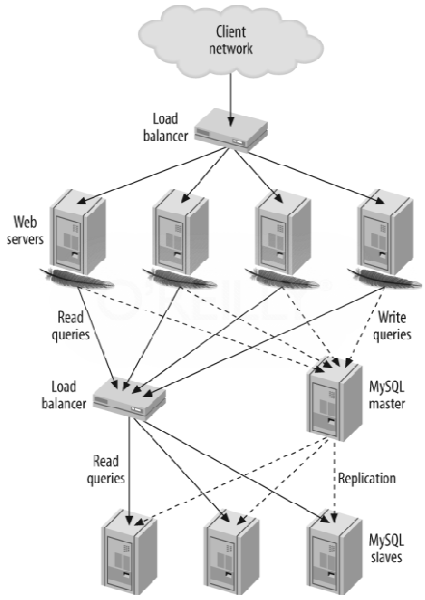
Load balance cluster bekerja dengan mengirimkan layanan-layanan di dalam sebuah jaringan ke node-node yang telah di-*cluster* untuk menyeimbangkan beban permintaan layanan di antara node *cluster*. Prinsip kerja *load balance cluster*, ketika node pada *cluster load balance* tidak bekerja maka aplikasi *load balance* akan mendeteksi kegagalan dan meneruskan permintaan ke node *cluster* lainnya.[15]

Pada dasarnya mekanisme penyeimbang beban *load balance* yaitu setiap beban yang masuk ke *load balance*r akan dialihkan ke masing-masing *server*, berdasarkan bobot yang *server* dan koneksi yang masuk. Sehingga setiap *request* yang masuk akan dibagi secara merata atau seimbang kepada masing-masing real *server* [4].



Gambar 3. *Load balancing*

Load balancing merupakan kemampuan untuk menyebarkan beban dari proses untuk sebuah aplikasi kepada beberapa sistem yang berbeda untuk meningkatkan kemampuan pemrosesan pada permintaan yang datang. Sehingga dengan *load balancing* akan mengirimkan beberapa porsi pemrosesan dari permintaan ke sebuah sistem kepada sistem independen lain yang akan ditangani secara bersamaan



Gambar 4. Arsitektur Load balance

Di dalam implementasi *load balance cluster*, terdapat beberapa cara untuk merancang solusi *load balance* tersebut. Pembuatan *load balance cluster* bisa dibuat dengan sebuah sistem operasi saja, dibuat oleh sebuah perangkat saja, dan bisa juga saja dibangun dengan kombinasi dari perangkat lunak dan perangkat keras. Secara garis besar pembuatan sistem *load balance* terdapat tiga kategori, yaitu pembuatan sistem *load balancing* dengan penerapan DNS round robin, Integrated *Load balancing*, dan Dedicated *Load balancing*. Dari masing-masing penerapan pembuatan *load balancing* tersebut mempunyai sistem kerja yang unik dan berbeda-beda, namun mempunyai tujuan hasil akhir yang sama.

1) Load balance dengan penerapan DNS Round Robin

Load balance DNS *round robin* ini merupakan metode paling sederhana di dalam membuat sebuah sistem *load balance*. Metode ini merupakan sebuah *fitur* dari aplikasi BIND (Barkeley Internet Name Domain). DNS *round robin* menggabungkan teknik penginputan penamaan yang teratur dan rapi dengan sistem perputaran *round robin*.[16]

Dalam metode ini, *record* DNS memberikan beberapa penamaan/ *domain name* yang lain untuk diwakili oleh sebuah nama domain utama, yang mana setiap penamaan memiliki masing-masing *record* dan mewakili alamat IP *Address* yang berbeda. Sehingga ketika domain utama diakses, maka DNS *server* akan mencari *record* tersebut. Ketika di dalam DNS *record* terdapat nama domain lain yang berhubungan dengan nama utama, maka kondisi inilah *server* akan menjalankan sistem perputaran *round robin* untuk menentukan nama domain mana yang akan dilanjutkan kepada pengakses.[6]

2) Integrated Load balancing

Integrated Load balancing merupakan sebuah fitur tambahan yang dimiliki oleh sebuah sistem operasi yang memiliki kemampuan sebagai *server*. Pada metode ini, *load balancing* bukanlah sebagai fungsi utama, yang mana performansi serta kemampuan yang sederhana maka metode ini digunakan untuk sistem yang berskala kecil menengah.[16]

3) Dedicated Load balancing

Metode Dedicated *Load balancing* ini murni diperuntukkan untuk memproses *load balancing* terhadap suatu *server*. Metode ini yang mempunyai tujuan utama untuk melakukan proses *load balancing*, sehingga metode dianggap sebagai metode *load balance* yang sesungguhnya [16].

D. High Performance Cluster

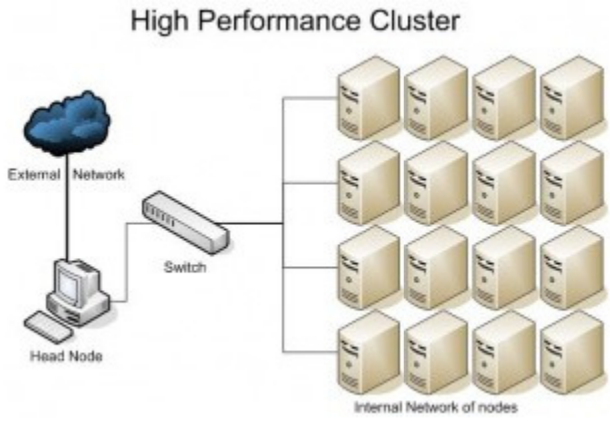
High performance cluster menggunakan sekumpulan *node-node* secara bersama dan dengan waktu bersama pula untuk mengerjakan dan menyelesaikan suatu tugas tertentu, yang mana biasanya dalam bentuk perhitungan yang berat dan rumit. Sistem kerja dari *high performance cluster* membagi pekerjaan menjadi beberapa bagian pekerjaan yang kecil sehingga waktu yang diperlukan untuk kebutuhan tersebut dapat diminimalisirkan. High Performance *Cluster* biasanya memungkinkan aplikasi bekerja secara paralel untuk meningkatkan kinerja aplikasi tersebut. Penerapan high performance *cluster* yaitu seperti me-render film animasi, simulasi, dan lain-lain.[7]

Cluster jenis ini merupakan sistem dengan performa dan skalabilitas tinggi, menggunakan infrastruktur jaringan *private* dan sistem operasi *open source* seperti Linux.

Kinerja dapat ditingkatkan dengan menambahkan mesin ke dalam suatu sistem. Hardware mesin yang digunakan sangat bervariasi, sebanyak yang dapat ditemukan di pasaran, mulai dari 2 (dua) *node* PC *stand alone* dengan Linux dan pemakaian *file* sistem bersama, sampai 1024 *node* di atas jaringan *low lantency*, berkecepatan sangat tinggi.

Cluster yang didedikasikan untuk melakukan proses komputasi secara paralel sehingga dapat menghasilkan performa komputasi yang tinggi dengan kecepatan yang tinggi pula. Karena kemampuannya dalam melakukan komputasi sangat cepat maka *cluster* jenis ini kebanyakan digunakan untuk tujuan ilmiah.

Mekanisme yang digunakan untuk melakukan pemrosesan secara paralel adalah dengan menggunakan MPI (*Message Passing Interface*). MPI adalah sebuah standar pemrograman yang memungkinkan pemrogram untuk membuat sebuah aplikasi yang dapat dijalankan secara paralel. Proses yang dijalankan oleh sebuah aplikasi dapat dibagi untuk dikirimkan ke masing-masing *node* yang kemudian masing-masing *node* tersebut mengolah dan mengembalikan hasilnya ke komputer *head node*.[9]



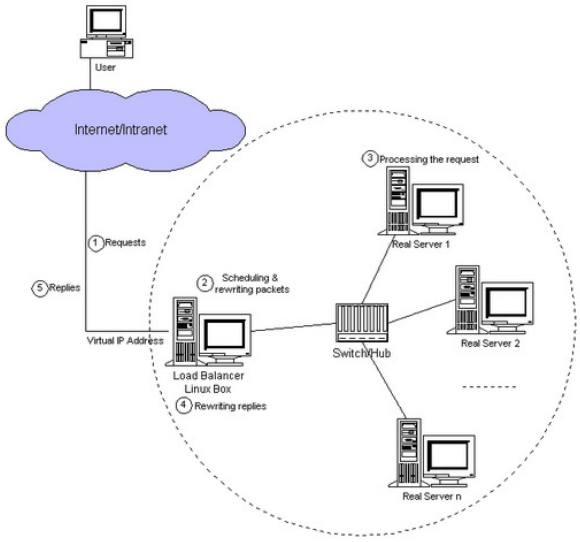
Gambar 5. *High Performance Cluster*

E. Linux Virtual Server

Linux Virtul *Server* yaitu suatu *server* yang mempunyai skalabilitas dan mempunyai *high availability* yang dibangun di atas sebuah *cluster* dengan beberapa real *server* yang berada di belakangnya. Pada Linux Virtual *Server* terdapat 3 metode, yaitu Linux Virtual *Server* via NAT (LVS-NAT), Linux Virtual *Server* via Direct-Routing (LVS-Direct Routing), dan Linux Virtual *Server* via Tunneling).

1) Linux Virtual Server via NAT (LVS-NAT)

Network Address Translation (NAT) merupakan mekanisme di mana memetakan sebuah alamat IP publik menjadi beberapa alamat lokal, yang mana walaupun banyaknya perangkat yang ada di area lokal tersebut maka juga dianggap sebagai satu alamat publik. Pada metode LVS-NAT ini *header* dari paket-paket tersebut ditulis ulang oleh *director*. LVS *director* menyamar menjadi *server* dan hal ini menciptakan anggapan bahwa *client* mengakses langsung *server-server* tersebut. *Director* harus dikonfigurasi sebagai *default gateway* dari *server-server* tersebut.[13]



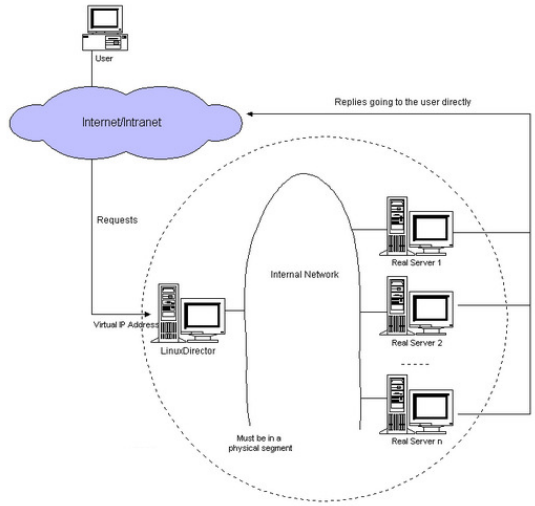
Gambar 6. Topologi *Load balance* Dengan LVS-NAT

Pada penelitian ini, *load balance* dibangun dengan menggunakan LVS-NAT. Adapun justifikasi dalam penerapan *load balance* dengan menggunakan LVS-NAT yaitu adanya keterbatasan IP *address* pada objek penelitian. LVS-NAT tidak menggunakan banyak IP publik di setiap *node*, penggunaan ip publik hanya pada di *node director* saja dan setiap *node cluster* menggunakan IP *private* sebagai penghubung ke *director* saja. Sehingga penggunaan IP publik lainnya dapat digunakan pada penggunaan yang lain.

LVS-NAT lebih aman sangat aman digunakan, yang mana *director* tidak berperan sebagai *web server*, melainkan *node-node* cluster yang berperan sebagai *web server*, dan *node cluster* tidak berhubungan langsung dengan *client* secara langsung. Setelah semua *request* diproses, maka *request* akan dikembalikan lagi kepada *director*.

2) Linux Virtual Server via Direct Routing (LVS-Direct Routing)

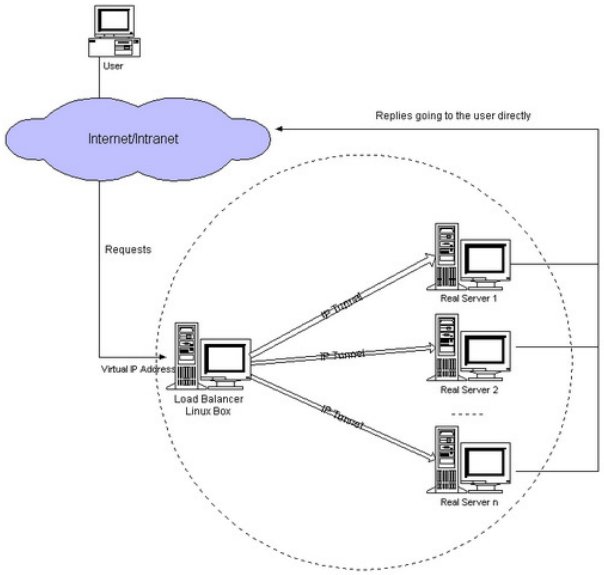
Dimana paket-paket yang ditujukan pada virtual *server* dilampirkan pada paket lain kemudian diarahkan ke salah satu node *cluster server*. Ketika *user* mengakses layanan virtual yang telah disediakan *cluster server*, maka *load balance* akan meneruskan paket dan memeriksa alamat dan tujuan dan port node *cluster server*. Node *cluster server* dipilih dari *cluster* dengan menggunakan algoritma penjadwalan, dan koneksi tersebut akan ditambahkan ke dalam *hash* yang merekam koneksi. Pada LVS-*direct routing* ini, *load balance server* bisa terletak di jaringan yang berbeda dengan *node cluster server*.[14]



Gambar 7. Topologi *Load balance* Dengan LVS-Direct Routing

3) Linux Virtual Server via Tunneling (LVS-Tunelling)

Seperti namanya, tiap-tiap *server* memiliki routing masing-masing ketika menjawab permintaan client. Jalur routing dari *server* terpisah dari director.



Gambar 8. Topologi *Load balance* Dengan LVS-Tunneling

F. Algoritma Penjadwalan

Algoritma Penjadwalan yaitu metode yang digunakan untuk membagikan beban kerja ke masing-masing *real server* yang berada di belakang *director*. Jenis-jenis algoritma penjadwalan yang umum digunakan yaitu algoritma *round robin*, *weighted round robin*, *ratio*, *fastest*, dan *least-connection*.

G. Iptables

Iptables merupakan sebuah aplikasi pada sebuah sistem operasi yang berfungsi sebagai alat *filtering* terhadap lalu lintas data. Pada iptables ini mengatur semua lalu lintas pada jaringan, baik yang masuk maupun keluar atau hanya sekedar melewati saja. Iptables dapat mengatur semua kegiatan dalam sistem komputer baik besar data yang boleh lewat, jenis paket data yang dapat diterima, mengatur trafik sesuai dengan asal dan tujuan data, *forwarding*, NAT, *redirecting*, pengelolaan port, dan *firewall*.[6]

III. HASIL DAN PEMBAHASAN

A. Analisa Kebutuhan

Load balance adalah gabungan dari beberapa *server* untuk menyelesaikan permasalahan keseimbangan beban dari suatu proses komputing. *Load balance* menggunakan sejumlah node-node yang akan dibagi secara merata.

Di dalam penerapan *load balance* tersebut diperlukan beberapa *hardware* dan *software* sebagai pendukung. Adapun kebutuhan *hardware* dan *software* dalam penelitian yaitu sebagai berikut:

1) Kebutuhan Perangkat Keras / Hardware

Didalam membangun *load balance* yang akan dibuat, diperlukan beberapa spesifikasi hardware yang dibutuhkan sebagai berikut:

- Processor: Core 2 Duo 2,1 GHz atau lebih.
- Hardisk: 20 GB atau lebih
- Memori: 1 GB atau lebih
- Ethernet card dan Kabel UTP
- Switch 8 Port

2) Kebutuhan Perangkat Lunak / Software

Kebutuhan *software* yang akan digunakan untuk keperluan implementasi dan pengujian *load balance* tersebut adalah sebagai berikut:

- *Load balance Server*/ director
- Sistem Operasi: Ubuntu *Server*
- Ipvsadm
- Httperf: aplikasi benchmark
- Paket-paket dependensi yang dibutuhkan untuk menjalankan director *server*.
- Node *Cluster Server*
- Apache2
- Mysql
- Php5
- Paket-paket dependensi yang dibutuhkan untuk menjalankan *node cluster server*

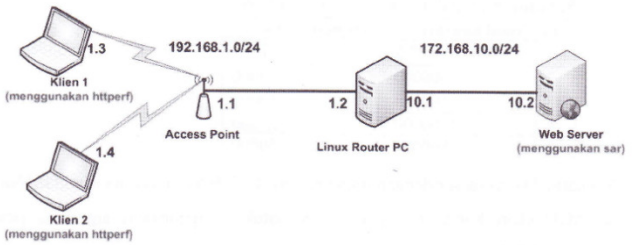
3) Perancangan Load balance

Setelah tahap analisa kebutuhan selesai, maka tahap selanjutnya yaitu tahap perancangan. Tahap perancangan ini akan menjelaskan perancangan model topologi yang akan dibangun dalam penelitian ini.

B. Perancangan Topologi

Pada perancangan topologi ini akan digunakan 2 model topologi, yaitu dengan menggunakan topologi *web server*

dengan *single server* dan menggunakan topologi *web server* dengan menggunakan penerapan *load balance*. Pada topologi *web server* dengan *single server*, semuanya hanya dikerjakan dan dibebankan pada sebuah *web server* saja.

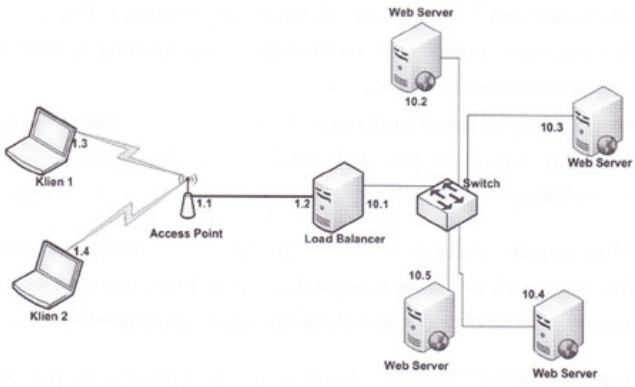


Gambar 9. Topologi *Web server* Dengan *Single Server*

Penjelasan topologi di atas yaitu:

- Topologi di atas hanya menggunakan sebuah node tunggal sebagai *web server*, yang mana bertindak sebagai master sekaligus *client server*.
- *Router* dan *web server* dihubungkan oleh switch
- User mengakses *web server* dengan media transmisi kabel maupun radio wireless.

Pada perancangan topologi dengan penerapan *load balance* ini, mempunyai sebuah *node* yang bertindak sebagai master / *director* dan beberapa *node* yang akan bertugas sebagai *client*. Dengan adanya penerapan *load balance* ini, semua beban kerja akan dikerjakan oleh node-node client secara merata yang telah diatur oleh Pada perancangan topologi dengan penerapan *load balance* ini, mempunyai sebuah *node* yang bertindak sebagai master / *director* dan beberapa *node* yang akan bertugas sebagai *client*. Dengan adanya penerapan *load balance* ini, semua beban kerja akan dikerjakan oleh *node-node client* secara merata yang telah diatur oleh master *cluster*.

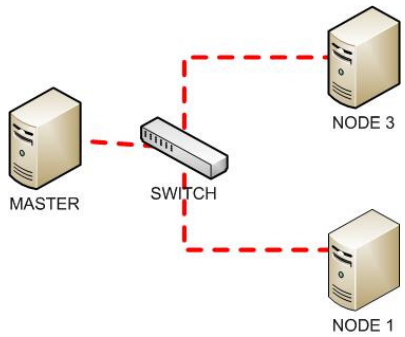


Gambar 10. Topologi Web server Load balance

Pada bagian hasil dan pembahasan, penulis akan menampilkan hasil perbandingan pengujian *load balance* dari respon time dan thorughput pada masing-masing arsitektur, baik dengan *single server* maupun *load balance*.

Gambar 10 menjelaskan bagaimana *client* mengakses sebuah *website* dan akan diterima serta dieksekusi terlebih dahulu oleh *load balance*r sebelum diserahkan ke masing-masing node client *web server* secara merata.

Perancangan model topologi dengan *load balance* yang akan dipakai dalam membuat *cluster* ini adalah sebagai berikut:



Gambar 11. Topologi *Load balance* Dengan 3 Node *Cluster*

Penjelasan topologi di atas:

- Pada topologi ini terdapat 1 *node* master dan 2 *node client* (node2 dan node3).
- Master berfungsi sebagai *director* untuk melakukan proses kerja *web server* biasa. Pada master ini, IP *Address* diset dalam bentuk *static*.
- *Node* 1 berfungsi sebagai *host*, yang mana *resource* pada *node* tersebut digunakan sebagai membantu dalam mengatasi beban kerja *web server*. Pada *node* ini, IP *Address* diset juga dalam bentuk *static*.
- *Node* 2 fungsinya sama dengan *node* 1 dan IP *Address node* 2 juga diset dalam bentuk *static*.
- Kedua *node* terhubung dengan menggunakan *switch*.

C. Perancangan Load balance

Pada tahapan ini menjelaskan uraian proses data dan juga proses dalam membangun sistem *load balance* serta juga tahapan hasil pengujian *load balance* nantinya. Gambar 11 memperlihatkan uraian proses data pada penerapan *load balance* pada objek penelitian, penjelasannya dapat diuraikan sebagai berikut:

- Menentukan spesifikasi hardware yang akan digunakan sebagai *server load balance* maupun node *cluster*.
- Ada beberapa langkah dalam instalasi software, yang mana langkah pertama yaitu menginstal sistem operasi yang digunakan pada *server load balance*. Dalam pemilihan sistem operasi perlu dipastikan sistem operasi mendukung paket IP virtual *server*. Langkah kedua yaitu menginstal paket-paket yang dibutuhkan membangun *server load balance*, dan yang ketiga menginstal aplikasi pengujian yang akan digunakan sebagai pengujian nantinya.
- Menjalankan paket-paket yang telah terinstal, dan kemudian mengkonfigurasi apa yang akan dibutuhkan dalam membangun *server load balance*.

- Hal-hal yang perlu dikonfigurasi yaitu menentukan Ethernet card sebagai penghubung gateway internet dan ke masing-masing node.
- Memeriksa dukungan IP virtual service pada sistem operasi
- Mengkonfigurasi ipvsadm untuk menentukan jumlah node yang akan digunakan sebagai anggota *cluster*.
- Pada *node cluster* perlu ditentukan spesifikasi yang dibutuhkan. Pada spesifikasi ini semua node *cluster* mempunyai spesifikasi yang sama.
- Langkah selanjutnya yaitu langkah instalasi dan konfigurasi pada *node cluster*. Pada langkah ini sama dengan langkah instalasi *server load balance*, namun hanya saja pada *node cluster* ini hanya menginstal paket-paket untuk kebutuhan *web server* biasa saja dan mengkonfigurasinya pada masing-masing node. Setelah itu konfigurasi yang perlu dilakukan yaitu menentukan Ethernet card yang akan digunakan sebagai penghubung ke *server* director.
- Setelah *node cluster* telah terinstal dan dikonfigurasi, maka selanjutnya kita sudah dapat melakukan pengujian dengan mengukur kinerja dari *load balance* tersebut. Yang mana hasil dari analisa kinerja *load balance* tersebut berupa kecepatan *web server* dari jumlah *request* yang dikirimkan ke alamat *address* dengan menggunakan aplikasi *benchmarking web server* yaitu httperf. Untuk mendapatkan hasil pengukuran, perlu dilakukan beberapa kali pengujian agar mendapatkan hasil yang jelas.
- Setelah melakukan analisa kinerja *load balance*, maka kita dapat mengambil hasil kesimpulan dari hasil kinerja *load balance* tersebut. Hasil dari kecepatan akses pada *web server* akan dijadikan sebagai perbandingan serta hasil akhir pada penelitian ini. Untuk selanjutnya kita akan mencari faktor-faktor apa saja yang menjadi pengaruh dalam *load balance* ini.

D. Konfigurasi Node Cluster Server

Pada tahapan ini hampir sama apa yang dilakukan pada tahapan sebelumnya, yaitu menentukan port Ethernet dan melakukan konfigurasi IP Address masing-masing node *cluster*, dan menginstalasi paket-paket yang dibutuhkan untuk node *cluster server*. Pada node *cluster* ini hanya membutuhkan 1 buah ethernet card yang berfungsi sebagai penghubung ke *server load balance*.

Pada tahapan instalasi dan konfigurasi paket yang dibutuhkan untuk node *cluster server*, paket yang diperlukan yaitu apache, mysql, dan php. Paket tersebut sama dengan paket dalam membangun sebuah *web server*, yang mana pada node *cluster server* hanya berperan sebagai *web server* saja. Perintah instalasi di terminal / console:

```
# apt-get install apache2
# apt-get install php5
# apt-get install mysql-server mysql-client
```

Pada tahapan konfigurasi node *cluster server* ini, kita terlebih dahulu mempersiapkan database yang akan digunakan pada *website* nantinya. Berikut perintah pada terminal / console:

```
# mysql –u root –panggi                > Masuk ke database
#Mysql>create database webuir;         > Membuat database
#Mysql>use webuir;
#Mysql>Source /home/anggi/webuir.sql
```

Langkah selanjutnya yaitu melakukan konfigurasi VirtualHost untuk mengatur konfigurasi URL pada *server*. Pada VirtualHost kita akan melakukan konfigurasi nama domain serta peletakan *directory* web. Berikut perintah pada terminal/ *console*:

```
$ sudo su            > masuk ke super user
# nano /etc/apache2/site-enabled/000-default >  konfigurasi virtualhost
```

Setelah itu kita akan meng-*copy file web* yang akan digunakan ke dalam *directory* /var/www dan membuat kepemilikan *file* serta memberikan *permission file* pada *file* web tersebut. Berikut perintah pada terminal / *console*:

```
$ sudo su                                  > masuk super user
# cp –R /home/anggi/uir /var/www/ > copy directory web
# chown –R www-data:www-data uir > merubah kepemilikan file
# chmod –R 777 uir > merubah permisision file
```

Setelah melakukan semua tahapan konfigurasi di atas, maka langkah selanjutnya mengecek semua *service* node *cluster* tersebut. Berikut perintah pada terminal/ *console*:

```
$ sudo su                                  > masuk ke super user
# /etc/init.d/apache2 status               > mengecek status apache
# /etc/init.d/apache2 restart              > merestart service apache
```

E. Hasil Load balance Untuk Perbandingan Waktu Respon

Dalam pengujian ini dilakukan pengukuran kecepatan waktu respon dari *server* tunggal maupun *server load balance* LVS-NAT. Pengukuran waktu respon ini digunakan untuk mengetahui bagaimana kemampuan *server* dalam memberikan pelayanan informasi terhadap *request* yang datang secara bersamaan. Berikut tabel hasil perhitungan *respons time* dari *server* tunggal dan *server load balance*:

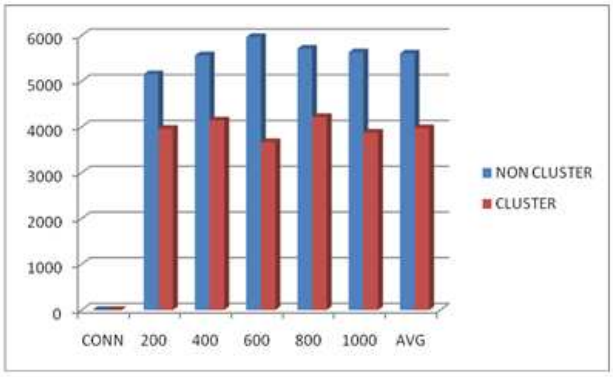
TABEL I
HASIL PERBANDINGAN *RESPON TIME SINGLE SERVER* DAN *LOAD BALANCE*

Total Connection	***Single Server***	***Server Load balance***
200	5155,6	3595,1
400	5566,3	4142,7
600	5967,1	3672
800	5714	4219,7
1000	5632,5	3876,4
Rata-rata	5607,1	3973,98

Tabel I menunjukkan hasil yang diperoleh dari hasil tes pengukuran kecepatan waktu respon dengan penerapan *load balance* LVS-NAT. Untuk waktu respon dapat dilihat pada *reply time* pada kategori *reply section* yang menampilkan nilai waktu respon *server* tersebut.

Untuk mengetahui perbandingan pengukuran waktu respon yang baik dapat dilihat nilai yang paling kecil dari perbandingan antara *server* tunggal dengan *server* dengan LVS-NAT. Hasil pengujian yang ditunjukkan pada tabel di atas dapat diketahui bahwa hasil pengujian dengan mengirimkan *request* sebanyak 200 sampai dengan 1000 *request* dengan 1000 koneksi, maka *respons time* yang dihasilkan oleh *server* tunggal membutuhkan waktu rata-rata 5607,1 ms dan sedangkan *respons time* yang dihasilkan oleh *server load balance* LVS-NAT membutuhkan waktu rata-rata 3876,4 ms. Pada tabel I di atas menunjukkan bahwa nilai rata-rata *respon time* pada *server load balance* lebih kecil dibandingkan dengan *server* tunggal.

Kecepatan *respons time* yang dipengaruhi dengan keberadaan *server load balance* dan 2 buah node *cluster* yang berada di dalamnya. Pada *server load balance* memiliki algoritma penjadwalan weight round robin yang dapat meneruskan *request* yang diberikan oleh *director* kepada anggota node *cluster* sesuai dengan beban yang diberikan ke masing-masing node *cluster*. Yang mana dengan adanya algoritma penjadwalan tersebut maka kerja sistem menjadi tidak berat dan *request* yang datang bisa dilayani dengan cepat.



Gambar 12. Grafik Perbandingan Respon Time *Single Server* dan *Load balance*

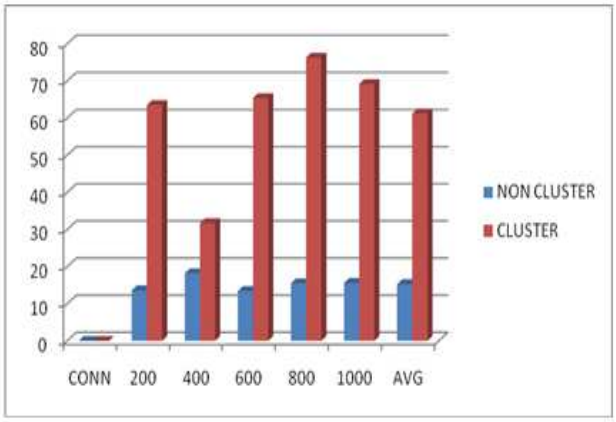
Pengujian *throughput* ini dilakukan untuk mengetahui kecepatan yang dihasilkan oleh *server* tunggal dan *server load balance* di dalam memberikan layanan yang datang secara bersamaan. Pada pengujian *throughput* ini juga menggunakan simulasi yang sama pada pengujian *respons time*, yang mana menggunakan simulasi sebanyak 1000 koneksi dengan mengirimkan *request* secara bertahap dengan 200, 400, 600, 800, dan 1000 *request* per detiknya. Berikut tabel hasil pengujian *load balance throughput* dari *server* tunggal dan *server load balance*:

TABEL III
HASIL PERBANDINGAN *THROUGHPUT SINGLE SERVER* DAN *LOAD BALANCE*

Total Connection	*Single Server*	*Server Load balance*
200	13,6	63,5
400	18,2	31,7
600	13,4	65,4
800	15,5	76,3
1000	15,6	69,2
Rata-rata	15,26	61,22

Tabel II menunjukkan hasil dari perhitungan kecepatan *throughput* yang digunakan dari penerapan *server* tunggal dan *server load balance* LVS-NAT. Untuk dapat mengetahui nilai *throughput* dapat dilihat pada Net I/O pada kategori *miscellaneous section* yang menampilkan nilai *throughput* tersebut.

Untuk mengetahui perbandingan pengukuran *throughput* yang benar dapat dilihat nilai yang paling besar dari perbandingan antara *server* tunggal dengan *server load balance* LVS-NAT. Hasil pengujian yang ditunjukkan pada tabel di atas dapat diketahui bahwa hasil pengujian dengan mengirimkan *request* sebanyak 200 sampai dengan 1000 *request* dengan 1000 koneksi, maka *throughput* yang dihasilkan oleh *server* tunggal menghasilkan waktu rata-rata 15,26 KB/s dan sedangkan *throughput* yang dihasilkan oleh *server load balance* dengan penerapan LVS-NAT menghasilkan waktu rata-rata 61,22 KB/s. Pada tabel 5.3.2 di atas menunjukkan bahwa nilai rata-rata *throughput* pada *server load balance* LVS-NAT lebih besar dibandingkan dengan *server* tunggal.



Gambar 13. Grafik Perbandingan *Throughput Single Server* dan *Load balance*

IV. SIMPULAN

Simpulan dari pengujian ini adalah:

1. *Load balance* menghasilkan nilai *respon time* yang lebih sedikit dan nilai *throughput* yang lebih besar dibandingkan dengan arsitektur *single server*.
2. Dengan algoritma penjadwalan *weighted round robin*, beban *web server* dapat didistribusikan dengan baik

kepada masing-masing *server* dengan ketentuan bobot yang diberikan.

3. Pada pengujian *web server* harus diberikan waktu proses pengujian (*timeout*) untuk mendapatkan proses *stress load* yang tidak terlalu lama. Semakin lama proses pengujian maka hasil pengujian tidak menghasilkan nilai yang baik untuk dijadikan sebagai perbandingan.
4. *Load balance* dapat membantu dalam mempercepat kinerja proses terhadap penyediaan layanan *web server*.
5. *Load balance* dapat memberikan ketersediaan informasi yang tinggi (*high availability*) terhadap *website*.
6. Dengan adanya *load balance server* kita dapat membagi beban sistem yang akan dikirimkan ke masing-masing *node* dengan menggunakan algoritma penjadwalan Weighted Round Robin.
7. Dengan adanya *load balance* dengan penerapan NAT, kita dapat menghemat jumlah IP yang diberikan oleh provider untuk dimanfaatkan semaksimal mungkin untuk perangkat yang lain.
8. *Load balance* dapat digunakan dalam berbagai hal yaitu pelayanan terhadap *website*, *learning management system*, *Mail*, pembagian beban *bandwith*, dan lain-lain.

DAFTAR PUSTAKA

[1] A. B. M. Moniruzzaman and Syed Akther Hossain (2014), "A Low Cost Two-Tier Architectur Model for High Availability *Cluster*s Application *Load balancing*", Volume 7, Number 1.

[2] Ambia Rachman Haryadi (2010), "Load *balancing* Menggunakan Linux Virtual *Server* (LVS) Via Network Address Translation (NAT) Pada PT. Bank Syariah".

[3] Andy Purnama Nurhatta, Adian Fatchur Rochim and R.Rizal Isnanto (2012), " Sistem Penyeimbang Beban Web *Server* Dengan Iptables", Volume 1, Number 3.

[4] Burhanuddin and Yusep Rosmansyah (2008)," Studi Mengenai Kinerja *Web server* Berbasiskan Linux Menggunakan Teknologi *Load balancing*".

[5] B. K. Gupta, S.C. Sharma and Jyothi Sethi (2013), "Performance Evaluation of ATM Networks with Round Robin and Weighted Round Robin Algorithm", Volume 5, issue 5.

[6] Desy Lukitasari and Ahmad Fali Oklilas (2010), Jurnal Generic, "Analisis Perbandingan *Load balancing* Web *Server* Tunggal Dengan Web *Server* Custer Menggunakan Linux Virtual *Server*", Volume 5, Number 2, 31.

[7] Imam Maghribi Mursal (2011), "Desain dan Implementasi *Load balancing* Jaringan Lokal Pada CV. Sukses Makmur Mandiri Palembang".

[8] Jefty Alvonsius Rabu, Joko Purwadi and Willy S. Raharjo (2012), "Implementasi *Load balancing* Web *Server* Menggunakan Metode LVS-NAT", Volume 8, Number 2.

[9] Jiani Guo and Laxmi Narayan Bhuyan (2006), "*Load balancing* In a *Cluster*-Based Web *Server* for Multimedia Application", Volume 17, Number 1.

[10] Mei Lu Chin, Chong Eng Tan and Mohammad Imran Bandan (2012), "Efficient DNS Based *Load balancing* For Bursty Web Application Traffic", Volume 1, Number 1.

[11] N. Khrisnamoorthy, R. Asokan, PhD and S. Sangeetha (2013)," Performance Evaluation of Weighted Round Robin Grid Schedulling", Volume 68, Number 13.

[12] Seok-Pil Lee and Eui-Seok Nahm (2012), "A New Approach to Modelling of Linux Virtual *Server* based on Performance Metrics Using an Optimal *Load balancing* Algorithm", Volume 6, Number 2.

[13] Sumit Srivastava, Pankaj Dadheech and Mahender Kumar Beniwal (2011)," Load *balancing* Using High Performance Computing *Cluster* Programming', Volume 8, Issue 1.

[14] Theddy R Maitimu (2008), "Perancangan dan Implementasi *Web Server Cluster*ing dengan Skema *Load balance* Menggunakan Linux Virtual *Server* Via NAT, Volume 5, Number 1.

[15] Yessy Asri (2010), "Rancang Bangun Aplikasi Setting *Load balancing Web server* Pada Freebsd", Volume 3, Number 1.