ANALISIS AVAILABILITAS LOAD BALANCING PADA WEB SERVER LOKAL

Dwi Nuriba

Fakultas Ilmu Komputer, Universitas Dian Nuswantoro

ABSTRACT

Perkembangan teknologi Web menyebabkan server-server yang menyediakan pelayanan di Internet harus mampu mengatasi permintaan dan beban kerja yang lebih besar dari sebelumnya. Untuk dapat memenuhi tuntutan perkembangan teknologi Web tersebut maka diperlukan teknologi load balancing. Laporan Kerja Praktek ini membahas implementasi load balancing dan menganalisa hasil availabilitas tersebut. Teknologi load balancing diimplementasikan menggunakan Linux Virtual Server (LVS). Parameter yang dianalisa adalah trhougt- put dan waktu respon. Dari hasil analisa terhadap availabilitas load balancing yang telah dilakukan, sistem load balancing dapat menjadi salah satu solusi yang efektif dan efisien untuk menciptakan sistem yang handal dengan tingkat ketersediaan tinggi, khususnya sebagai web server.

Kata Kunci : Load Balancing, Web Server, Analisis.

1. Pendahuluan

1.2 Latar Belakang

Teknologi *cluster* merupakan suatu teknologi yang memungkinkan sejumlah komputer untuk bekerja sama menyelesaikan permasalahan komputasi biasa. Permasalahan komputasi tersebut bisa berupa komputasi sains (dengan

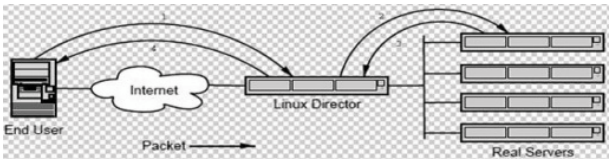
pemakaian CPU yang insentif) sampai bermacam proses dan *service* yang tidak memiliki kesamaan. Teknologi *cluster* ini

dibutuhkan untuk beberapa server agar menjadi suatu sistem tunggal sumber daya komputasi yang melakukan pekerjaan besar dan dapat meng-*handle* berbagai permintaan dari pengguna dalam sistem. Dari sisi pengguna tidak merasa bahwa pekerjaan yang dia berikan telah dibagi ke mesin fisik yang berbeda. Secara mendasar terdapat dua macam cluster[1]: *Fail-over*: bila ada layanan pada satu mesin yang gagal, mesin lainya akan mengambil alih. *Load-balancing*: kemampuan *fail over*, ditambah dengan fungsionalitas untuk memilih komputer yang paling tidak sibuk.

Load balancing merupakan hal penting untuk kinerja pemrosesan paralel. Ide dasarnya adalah memindahkan proses dari *node* yang memiliki beban kerja tinggi ke *node* yang memiliki beban kerja rendah. Tujuannya agar waktu pengerjaan tugas lebih rendah dan menaikkan *utilitas* sumber daya sistem. Bagian penting dari strategi *load balancing* adalah migration *policy*, yang menentukan kapan suatu migrasi terjadi dan proses mana yang dimigrasikan. Proses migrasi pada *load balancing* terjadi pada dua bentuk, yaitu[1]: *Remote execution* (juga disebut *non-preemptive migration*). Pada strategi ini, suatu proses baru (bisa secara otomatis) dieksekusi pada *host remote*. *Pre-emptive migration*, pada strategi ini proses akan dihentikan, dipindahkan ke *node* lain, dan diteruskan. Load balancing bisa dilakukan secara *eksplisit* oleh user ataupun secara *impilisit* oleh sistem.

2. **Linux Virtual Server**

Linux *Virtual* server adalah server yang mempunyai skalabilitas dan ketersediaan yang tinggi yang dibangun di atas sebuah *cluster* dari beberapa *real* server. Arsitektur dari sebuah server *cluster* adalah benar-benar transparan sampai ke
end-user dan masing-masing *user* berinteraksi dengan sistem seolah-olah hanya ada satu *virtual* server dengan performa yang tinggi.



Gambar 1. Arsitektur LVS

Linux *virtual* server dikembangkan dari code IP *masquerading* dan beberapa *port forwarding* dari Steven Clarke. Mendukung layanan TCP dan UDP seperti HTTP, Proxy, DNS dan sebagainya. Untuk protokol yang mengirimkan alamat IP dan nomor *port* sebagai data aplikasi, kode tambahan diperlukan untuk menanganinya.

Linux *virtual* server via NAT, merupakan suatu metode yang memanipulasi alamat IP dan nomor *port* baik sumber maupun tujuannya. Alamat IP *public* disamarkan untuk digunakan oleh alamat IP *private* agar bisa berhubungan dengan dunia luar (internet). Semua proses masuk (*end-user*) dan keluarnya (*real* server) paket harus melalui satu alamat IP *public* saja

(*director/load balancer/virtual* server). *Network Address Translation* (NAT) adalah sebuah metode pada mesin *load balancing* yang memetakan alamat IP (*Internet Protocol*) dan nomor *port* dari sebuah *group (private)* ke *group (public)* lainnya.

3. Metode Penelitian

1. Linux *director*, yang akan berperan sebagai *load balancing* yang bertugas mengalihkan permintaan untuk *service*

   http ke beberapa *real* server *cluster*.

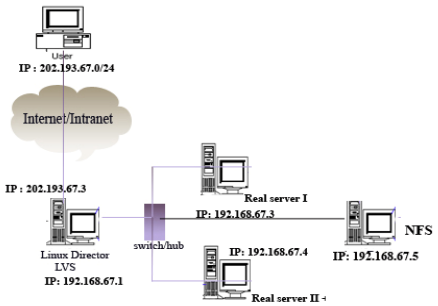
   - Linux *director* : 202.193.67.3/24 dan 192.168.67.1/28 sebagai *gateway*.

2. *Real* server, merupakan server sebenarnya yang akan melayani permintaan pada *service* http yang diberikan oleh linux director. Dengan masing-masing IP *address*:

   - *Real* server I : 192.168.67.3/28.
   - *Real* server II : 192.168.67.4/28.

3. NFS server, merupakan penyedia konten atau *file*, untuk dapat digunakan secara bersama oleh *real* server dalam *cluster*.

Arsitektur topologi jaringan seperti pada gambar di bawah ini:



Gambar 2. Desain Topologi Jaringan Linux Virtual Server

4. Hasil dan Pembahasan

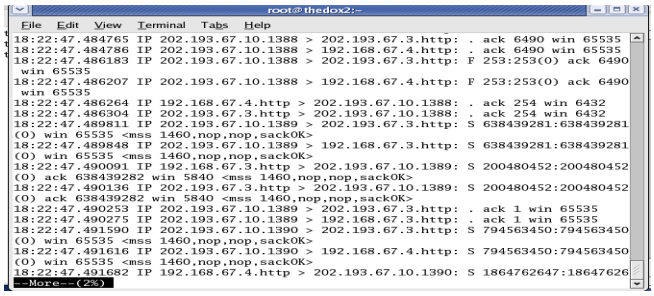
4.1 Pengujian Kerja *Load Balancing*

Pada tahap ini akan dilakukan pengujian terhadap *load balancer*. apakah *load balancer* telah dapat berfungsi dengan baik dan dapat menjalankan tugasnya. Untuk pengujiannya akan dilakukan dalam beberapa langkah:

- Pengujian dengan melihat paket yang lewat dari pengguna ke *load balancing* tersebut.

Tetapi *load balancing*,

membaginya kepada masing-masing *real* server. Berikut adalah gambar hasil *capture* dengan menggunkan tcpdump pada *port* 80:



Gambar 3. *Capture* Dengan

Tcpdump

Dari gambar hasil *capture* di atas. Dapat dilihat kalau permintaan yang mengarah ke *load balancing* dengan IP

202.193.67.3 akan dialihkan ke masing masing *real* server.

- Pengujian algoritma yang digunakan *load balancing*

  Pengujian pada tahap ini, dilakukan untuk dapat membuktikan, apakah algoritma yang diterapkan dari sisi *load balancing* telah dapat dijalankan sesuai dengan teori yang ada. Algoritma yang digunakan adalah WLC (*Weight Least Connection*), yang melihat bobot dan koneksi tersedikit dari *real* server. Berikut adalah gambar tabel pengujiannya:

  - Percobaan I

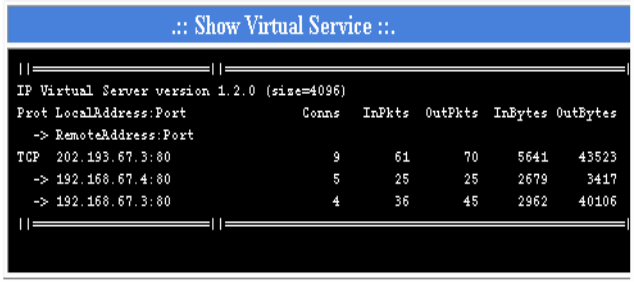
.:: Show Virtual Service ::.

IP Virtual Server version 1.2.0 (size=4096)

Prot LocalAddress:Port -> RemoteAddress:Port	Conns	InPkts	OutPkts	InBytes	OutBytes
TCP 202.193.67.3:80	7	49	57	4370	35681
-> 192.168.67.4:80	4	20	20	2158	2716
-> 192.168.67.3:80	3	29	37	2212	32965

Gambar 4. Tabel *Virtual Service*

- Percobaan II



Gambar 5. Tabel *Virtual Service* Coba II

Pada percobaan kedua ini, *request* kembali dilakukan oleh pengguna. Hasil yang diperoleh ketika *request* dilakukan lagi yaitu penambahan jumlah koneksi dan paket pada tabel *virtual service*. Koneksi yang masuk bertambah 2 menjadi 9 koneksi dan paket bertambah 12 menjadi 61 paket yang diterima *load balancing*. Setelah itu *load balancing* menambahkan masing-masing 1 koneksi untuk setiap *real* server.

2 Perbandingan Waktu Pelayanan Webserver

Tabel 1. Perbandingan Webserver Tunggal dan Cluster

	Client	Client Windows	Client Linux	Client Linux
Webserver	9:0	7:5	4:2	4:2
Webserver	9:2	8:1	5:2	5:2
Jumlah	14 detik	27 detik	59 detik	1 menit 3

Dari tabel perbandingan di atas dapat dilihat. Lamanya waktu yang diperoleh saat melakukan proses *download mirror* sebanyak dua kali secara bersamaan oleh masing-masing *client* kepada webserver tunggal dan *cluster*. Hasil yang diperoleh ialah waktu *request* ke webserver *cluster* untuk dua *request* oleh *client* windows dan linux lebih cepat dibandingkan webserver tunggal. Dimana perbandinganya lebih cepat 14 detik dan 27 detik, 59 detik dan 1,3 menit daripada webserver tunggal. Dari hasil yang diperoleh di atas dapat dibuktikan bahwa webserver *cluster* lebih memiliki performa waktu dalam melayani permintaan *client* lebih cepat dibandingkan webserver tunggal.

3 Perbandingan Beban Webserver

3.1 Webserver Tunggal

Tabel 2. Perbandingan Beban Webserver Tunggal

Jumlah	Paket	Paket	Byte	Byte Keluar
393	84215	639015	5242547	933171K

Dari tabel di atas dapat dilihat bahwa beban yang diperoleh webserver tunggal dari pengguna adalah sebagai berikut:

Jumlah Koneksi	Paket Masuk	Paket Keluar	Byte Masuk	Byte Keluar
2074	43089	324339	2698925	473140K
1856	38608	314992	2413418	460056K

- Koneksi sebanyak 3930.
- Paket yang masuk sebanyak 84215.
- Paket yang keluar sebanyak 639015.
- Paket yang masuk dalam *byte* sebanyak 5242547.
- Paket yang keluar dalam hitungan *byte* sebanyak 955567104 atau 933171 Kb.

3.2 Webserver *Cluster*

Tabel 3. Total Paket Masuk

Jumlah	Paket Masuk	Paket	Byte	Byte Keluar
3930	81697	639331	5112343	933299K

Tabel 4. Total Paket Webserver Cluster

.

4.4 Perbandingan Kecepatan *Transfer* Data

- Perbandingan webserver *cluster* dan tunggal

Tabel 5. Perbandingan Hasil

Download

Besar File	Webserver	Webserver
133	810.08 kb/s	905.32 kb/s
133	785.84 kb/s	826.05 b/s

4.5 Perbandingan Availibilitas Webserver

- Webserver Tunggal

Pengujian pertama akan dilakukan pada webserver tunggal, dengan cara me-*non*-aktifkan *service* http pada webserver tunggal dengan berasumsi bahwa server tersebut telah *down*. Kemudian dilakukan pengaksesan ke

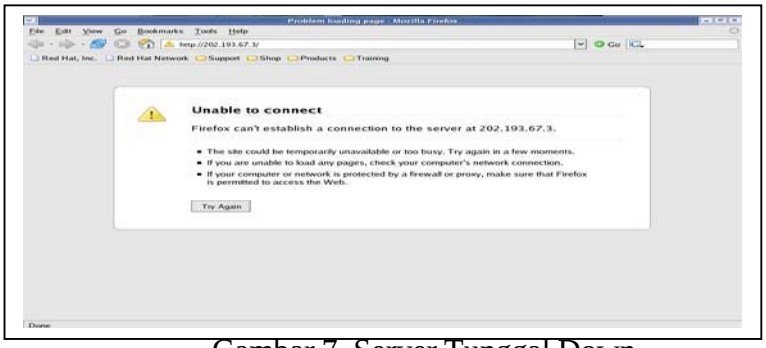
webserver tersebut. Dan hasil yang diperoleh adalah webserver tersebut tidak dapat diakses oleh pengguna, dikarenakan webserver tersebut telah *down*. Dengan demikian maka butuh

beberapa saat untuk dapat mengaktifkan atau memperbaiki webserver tersebut. Sehingga avalaibilitas dari webserver tunggal tidak selalu dapat memberikan *non-stop* servis terhadap pengguna. Hasil pengujiannya dapat dilihat seperti di bawah ini:

- Pengujian akses ke webserver dengan URL-nya http://202.193.67.3, sebelum webserver *down*.

Gambar 6. Server Sesaat Sebelum Down

- Pengujian akses ke webserver tunggal, setelah webserver *down*.



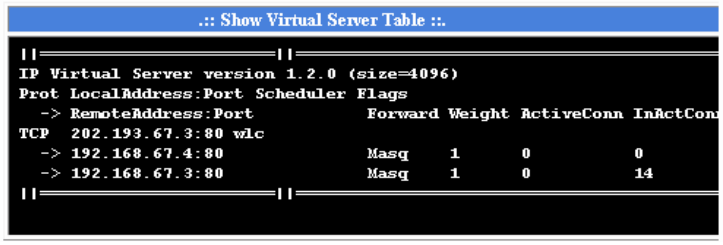
Gambar 7. Server Tunggal Down

- Webserver *Cluster*

Pengujian yang kedua akan dilakukan pada webserver *cluster*, dengan cara me-*non*-aktifkan salah satu *service* http pada *real* Server, dengan berasumsi bahwa server tersebut telah menglami *down*. Pada pengujian ini juga, akan dilakukan uji coba pada mesin *load balancer* dalam meng-implementasikan avalaibilitas yang dapat dicapai dengan pengurangan dan penambahan *real* server secara otomatis atau implisit oleh sistem saat mengalami kegagalan. Kemudian akan dilakukan

permintaan ke webserver, jika permintaan masih dapat dilayani, maka dapat ditarik kesimpulan bahwa avalaibilitas dari webserver *cluster* akan tetap terjaga dalam melayani permintaan pengguna. Hasil pengujian adalah sebagai berikut:

- Tabel IPVSADM sebelum salah satu webserver *down*.

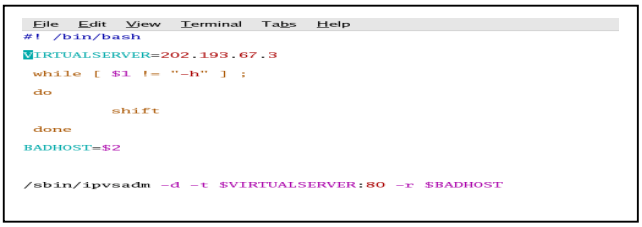


Gambar 8. Tabel IPVSADM

Dari table di atas dapat dilihat bahwa sebelumnya webserver *cluster* masih terdiri dari dua server dengan IP *address*:
192.168.67.4 dan 192.168.67.3.

- Webserver 192.168.67.3 *Down*.

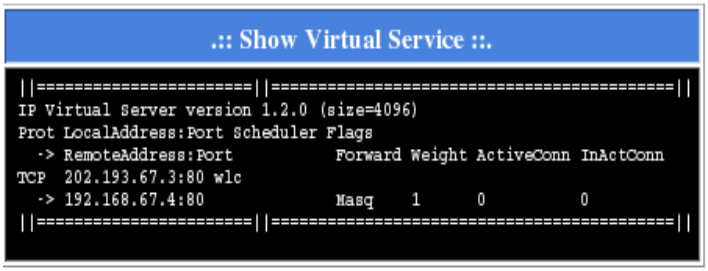
  Pada tahap ini *service* http pada web server di *non*-aktifkan dengan berasumsi bahwa webserver ini mengalami kegagalan atau *down*. Dengan demikian sistem akan menghapus secara otomatis *real* server 192.168.67.3 dari *real* server. Dan agar proses penghapusan dapat dilakukan secara otomatis maka pada *load balancing* sendiri dipasang sebuah sistem *monitoring daemon* dari masing-masing *real* server. Sistem *monitoring* ini me-*monitoring daemon* httpd pada masing-masing *real* server. Jika salah satu *daemon* httpd pada *real* server tidak berjalan, maka sistem akan menjalankan *script alert* yang isinya adalah penghapusan *real* server pada tabel. Berikut adalah penggalan *script* bash *shell* saat *alert* oleh sistem *monitoring*:



Gambar 9. *Script Alert*

Dengan demikian, jika salah satu *real server* mengalami *down* maka secara otomatis akan dihapus dari tabel *virtual service*. Sehingga *load balancing* tidak lagi mengalihkan permintaan ke server tersebut.

- Tabel IPVSADM setelah salah satu webserver down.



Gambar 10. Tabel Ipvsadm

Pada tabel di atas dapat dilihat bahwa *real* server dengan IP 192.168.67.3 telah dihapus secara implisit oleh sistem dari tabel dan anggota *cluster* karena mengalami *down* atau kegagalan. Namun saat salah satu webserver dalam *cluster down*. Sistem akan tetap dapat melakukan pelayanan dengan tetap menampilkan halaman web jika dikontak dari web *browser*. Dan dapat dilihat pada pengujian di bawah ini.

- Pengujian akses halaman web ke webserver.

Gambar 11. Web *Page Cluster*

- Webserver 192.168.67.3 diaktifkan

Pada tahap ini *service* http pada webserver diaktifkan kembali dengan berasumsi bahwa webserver ini kembali telah aktif dan siap untuk digunakan lagi. Saat server kembali diaktifkan dan dimasukkan ke dalam *cluster*. Maka secara otomatis *load balancing* akan

menambahkan *real server* tersebut ke dalam tabel *virtual service*. Sehingga *load balancing* dapat kembali memberikan beban ke *real* server tersebut.

```
File  Edit  View  Terminal  Tabs  Help
#! /bin/bash

VIRTUALSERVER=202.193.67.3

 while [ $1 != "-h" ] ;

 do

        shift
```

Dengan demikian jika server yang *down* kembali aktif, maka dapat langsung dimasukkan ke dalam *cluster* secara otomatis, untuk kembali dapat melaksanakan tugasnya.

- Tabel IPVSADM setelah webserver 192.168.67.3 kembali aktif

Gambar 12. Tabel Ipvsadm *Up*

Pada tabel di atas dapat dilihat bahwa *real server* 192.168.67.3 telah dimasukkan kembali ke dalam tabel dan menjadi anggota *cluster* setelah *service* webservernya diaktifkan kembali.

5. Kesimpulan

Implementasi webserver *cluster* dengan skema *load balancing* dapat meningkatkan performa sistem yang lebih baik dibandingkan dengan menggunakan webserver tunggal. Implementasi webserver *cluster* dengan skema *load balancing* dapat memberikan availabilitas sistem yang tetap terjaga dan skalabilitas yang cukup untuk dapat tetap melayani setiap
request dari pengguna. Linux *virtual* server via NAT dapat menjadi solusi jika ingin menerapkan webserver *cluster* dengan skema *load balancing*, jika memiliki keterbatasan IP *public*.

Daftar Pustaka

[1] Eko Budi Kristanto, "Mengenal Teknologi Load Balancing", 2 Mei 2014. [Online]. Avaliable:http://www.fxekobudi.net/networking/mengenal-teknologi-load-balancing/. [Accessed 4 Mei 2012].

[2] Utdirartatmo, F. (2004). *Clustering PC di Linux dengan OpenMosix dan ClusterKnoppix*, Yogyakarta: Penerbit ANDI.

[3] Sirajuddin, Affandi, Ahmad, and Setijadi, Eko. "Rancang Bangun Server Learning Management System Menggunakan Load Balancer dan Reverse Proxy", *Jurnal Teknik POMITS,* vol. 1, no. 1, pp.23-28, Maret 2012.

[4] Badrul, Sugiarto, Wahyudi, and Suprayogi, Teknik Komputer Jaringan seri B. Jakarta Timur, Inti Prima Promosindo, 2012.

[5] Tim Penyusun Buku Pedoman TA Fasilkom UDINUS, Buku Pedoman TA TI-S1 Fasilkom Udinus, Semarang: Udinus Press, 2013.