

Penggunaan *Software Nginx* Sebagai *Load Balancing Web Server Clustering*

Sondang Sibuea^{1*)}, Yohanes Bowo Widodo²⁾, Muhammad Nur Khaliq³⁾

¹⁾²⁾Teknik Informatika, Fakultas Komputer, Universitas Mohammad Husni Thamrin

²⁾Sistem Informasi, Fakultas Komputer, Universitas Mohammad Husni Thamrin

^{*)}Correspondence author: sondsib@gmail.com, Jakarta, Indonesia

DOI: <https://doi.org/10.37012/jtik.v10i1.2184>

Abstrak

Ketika suatu situs web mengalami tingkat kunjungan yang tinggi, berdampak peningkatan permintaan layanan pada server web tunggal. Akibatnya, kinerja server web menjadi terbebani dan kemungkinan terjadi *overload*, sehingga server tunggal tidak mampu melayani semua permintaan dari pengguna. Untuk mengatasi hal tersebut, dapat dilakukan sistem *cluster* dan metode *load balancing* pada server. Dalam sistem *cluster*, beberapa server dapat dikombinasikan. Permintaan dari pengguna dapat dibagi secara merata ke seluruh *cluster* server web. Untuk melakukan *cluster* pada server, digunakan *software nginx* sebagai metode untuk membagi tugas yang diberikan oleh pengguna kepada *cluster server web*. Dengan menerapkan *cluster server web* dan metode *load balancing* sebagai pembagi, kinerja layanan kepada pengguna dapat dipercepat dibandingkan dengan hanya menggunakan satu server web. Jika terjadi kegagalan pada perangkat keras yang mengakibatkan matinya server secara keseluruhan, server lain dalam *cluster* akan mengambil alih fungsi server yang mati. Dengan demikian, komputer klien tidak menyadari adanya kegagalan pada server, karena proses yang sedang berlangsung pada server yang gagal atau mati akan dilanjutkan oleh server cadangan. Setelah dilakukan pengujian, dapat diambil kesimpulan bahwa dengan menggunakan *load balancing* bisa dimanfaatkan sebagai *backup* atau *failover*, jika salah satu server *down* dapat diatasi oleh server lainnya. Pelayanan request data dari user dapat ditangani lebih cepat. Rekomendasi yang dapat diberikan guna pengembangan lebih lanjut maupun melengkapi penelitian ini, yaitu dengan melakukan *load balancing* pada database server cluster.

Kata Kunci: *Load Balancing, Cluster, Web Server, Nginx, Failover.*

Abstract

When a website experiences a high level of visits, it results in increased service requests on a single web server. As a result, web server performance becomes burdened and overload is possible, so that a single server is unable to serve all requests from users. To overcome this, a cluster system and load balancing method can be implemented on the server. In a cluster system, several servers can be combined. Requests from users can be divided evenly across the web server cluster. To cluster the server, nginx software is used as a method for dividing tasks assigned by the user to the web server cluster. By implementing a web server cluster and load balancing methods as a divider, service performance to users can be accelerated compared to using only one web server. If a hardware failure occurs which results in the entire server shutting down, another server in the cluster will take over the function of the dead server. In this way, the client computer is not aware of a failure on the server, because the processes that are ongoing on the failed or dead server will be continued by the backup server. After testing, it can be concluded that using load balancing can be used as a backup or failover, if one server goes down it can be handled by another server. Data request services from users can be handled more quickly. Recommendations that can be given for further development and complementing this research are by carrying out load balancing on the cluster server database.

Keywords: *Load Balancing, Cluster, Web Server, Nginx, Failover.*

PENDAHULUAN

Perkembangan teknologi informasi dan komunikasi saat ini mengalami kemajuan yang pesat. Setiap individu atau organisasi telah menggunakan teknologi informasi dan jaringan Internet sebagai alat bantu dalam berkomunikasi untuk menyelesaikan pekerjaannya. Salah satu tantangan yang dihadapi oleh perusahaan atau instansi skala besar maupun menengah adalah harus memiliki infrastruktur teknologi informasi dan komunikasi yang kuat dan memadai untuk mengelola ratusan bahkan ribuan data penting setiap harinya. Salah satu infrastruktur yang digunakan dalam pengelolaan data tersebut adalah server. Server diharapkan dapat melayani permintaan user dalam jumlah yang besar. Gangguan pada sistem dapat terjadi ketika server utama mati dan tidak ada server cadangan untuk menggantikan fungsi server utama yang mati, sehingga komunikasi antar jaringan terganggu.

Web server adalah contoh server yang sering diakses oleh klien (*client*), karena di dalamnya terdapat sistem informasi yang diperlukan oleh klien. Semakin banyak jumlah klien yang mengakses server, semakin tinggi beban kerja server tersebut. Solusi untuk mengatasi masalah ini adalah dengan menggunakan sistem *server clustering* dengan teknik *load balancing*. Software Nginx adalah salah satu aplikasi *load balancing* yang dapat membagi permintaan dari klien.

Jaringan komputer adalah suatu sistem yang terdiri dari kumpulan komputer dan perangkat lainnya yang saling terhubung dalam satu kesatuan. Setiap komputer, printer, atau perangkat lain yang terhubung dalam jaringan disebut sebagai node. Ada dua jenis jaringan komputer berdasarkan fungsinya, yaitu :

1. *Peer-to-peer*

Peer to peer adalah jaringan komputer dimana setiap komputer bisa menjadi *server* sekaligus *client*.

2. *Client-Server*

Jaringan dengan jenis *client* dan *server* adalah jaringan dimana ada sebuah komputer yang selalu menyediakan sumber daya dan digunakan oleh komputer lain, yang disebut komputer *server*. Komputer yang hanya menerima dan mengakses ketersediaan data dari komputer lain akan disebut komputer *client*.

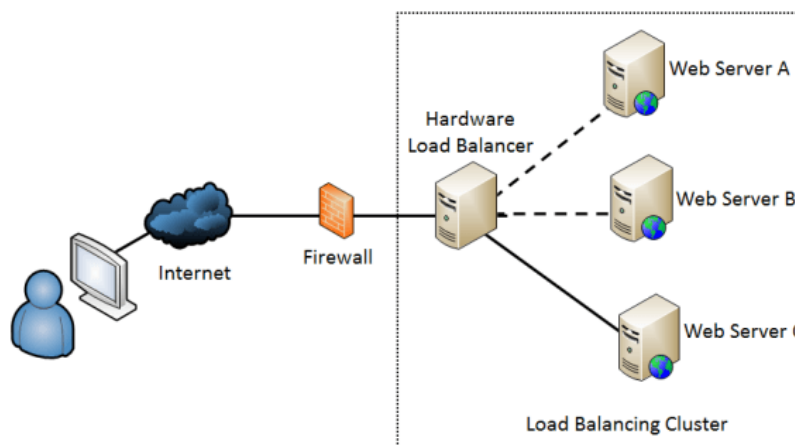
Secara umum *load balancing* adalah pembagian beban kerja secara seimbang. *Load balancing* dalam *computer internet working* adalah suatu metode untuk mendistribusikan beban kepada beberapa *host* sehingga beban kerja menjadi lebih ringan. Ini bertujuan agar waktu rata-rata mengerjakan tugas menjadi singkat. *Load balancing* adalah teknik untuk mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara seimbang, agar trafik dapat berjalan optimal, memaksimalkan *throughput*, memperkecil waktu tanggap dan menghindari *overload* pada salah satu jalur koneksi. Manfaat dari *load balancing*:

1. Keandalan (*Reliability*)

Dapat melayani *user* dengan baik, dan jaminan *reliability* memungkinkan *user* dapat melakukan pekerjaan dengan lancar.

2. Skalabilitas dan ketersediaan

Skalabilitas akan meningkat, selain itu faktor ketersediaan data juga akan meningkat, karena apabila salah satu *server* mati maka layanan terhadap pengguna tidak akan terganggu, karena ada *server* yang lain yang sudah dipersiapkan sebagai *server* dengan pengaturan dengan *Load Balancing*.



Gambar 1. Struktur *Load Balancing*

Web Server merupakan perangkat lunak *server* yang menjadi tulang belakang dari *world wide web (www)*. *Web server* menunggu permintaan dari *client* yang menggunakan *browser* seperti *mozilla*, *chrome*, dan program *browser* lainnya. Jika terdapat permintaan dari *browser*, maka *web server* akan memproses permintaan itu kemudian memberikan hasil prosesnya berupa data yang diinginkan pada *browser*.

Proses menghubungkan beberapa komputer agar dapat bekerja dinamakan *clustering*. *Cluster* merupakan mekanisme yang mendasar untuk membuat kompleksitas dan keberagaman melalui pengumpulan dan penggabungan dari elemen-elemen dasar yang sederhana. Ada beberapa jenis *cluster*, antara lain :

1. *High Availability Clusters*

High-availability cluster adalah untuk meningkatkan ketersediaan akan layanan yang disediakan oleh *cluster* tersebut. Pada umumnya tipe *clustering* ini sering disebut juga sebagai *Failover Cluster*.

2. *Load Balancing Clusters*

Load balancing Clusters bekerja dengan cara melakukan proses penyampaian atau pendistribusian pembagian beban kerja dari data yang diproses secara merata melalui *node-node* yang bekerja berada di belakang (*back- end node*) sehingga semua operasi dapat berjalan dengan baik.

3. *Computer Cluster*

Jenis *Computer Cluster* adalah untuk tujuan komputasi. Sebagai contoh, sebuah tugas komputasi mungkin membutuhkan komunikasi yang sering antara *node*. *Cluster* tersebut menggunakan sebuah jaringan terdedikasi yang sama, yang terletak di lokasi yang sangat berdekatan. Desain *cluster* seperti ini, umumnya disebut juga sebagai *Beowulf Cluster*.

Nginx adalah *server HTTP* dan *reverse proxy* berbasis *open-source* yang dapat digunakan sebagai *proxy IMAP/POP3*. *Nginx* tidak bergantung kepada *thread* untuk melayani *client*. Sebaliknya, *nginx* menggunakan arsitektur *asynchronus* yang lebih stabil. Arsitektur ini membutuhkan lebih sedikit memori serta dapat mengatasi ribuan koneksi pada saat yang bersamaan. *Nginx* juga bisa dimanfaatkan sebagai aplikasi *load balancing* untuk *server*.

Berikut algoritma untuk penjadwalan *Round Robin*:

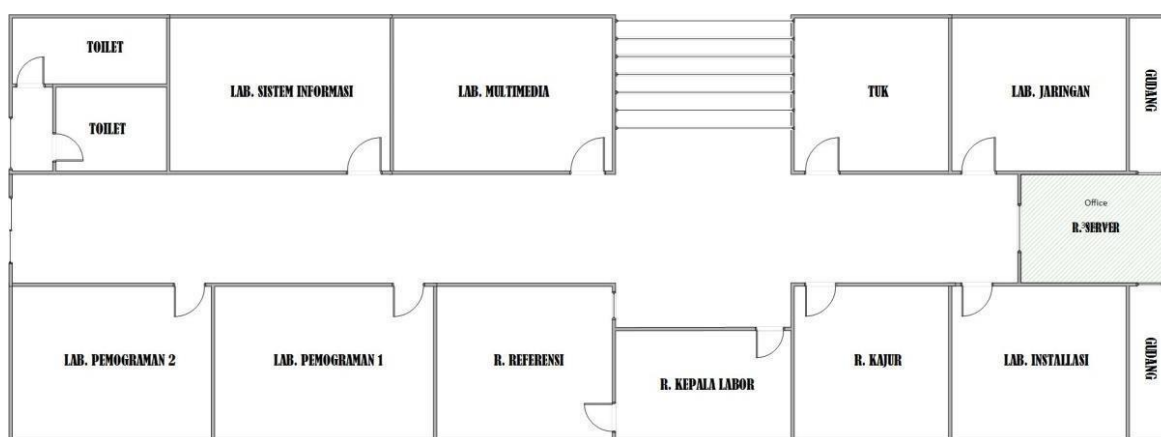
1. Setiap proses mendapat jatah waktu CPU (*time slice/ quantum*) tertentu umumnya antara 10-100 milidetik.
 - ✓ Setelah *time slice/quantum* maka proses akan di-*preempt* dan dipindahkan ke antrian *ready*.
 - ✓ Proses ini adil dan sangat sederhana.

2. Jika terdapat n proses di “antrian *ready*” dan waktu *quantum* q (milidetik), maka:
 - ✓ Setiap proses akan mendapatkan $1/n$ dari waktu CPU.
 - ✓ Proses tidak akan menunggu lebih lama dari: $(n-1)q$ *time units*.
3. Kinerja dari algoritma ini tergantung dari ukuran *time quantum*.
 - ✓ *Time quantum* dengan ukuran yang besar maka akan sama dengan *FCFS*.
 - ✓ *Time quantum* dengan ukuran yang kecil maka *time quantum* harus diubah ukurannya lebih besar dengan respek pada alih konteks sebaliknya akan memerlukan ongkos yang besar.

Sistem yang dibangun untuk *web server clustering* dengan skema *load balancing* menggunakan *nginx* adalah dengan menggunakan *nginx* sebagai *front end server load balancing* dan menggunakan dua server sebagai *web server* yang akan menyediakan informasi.

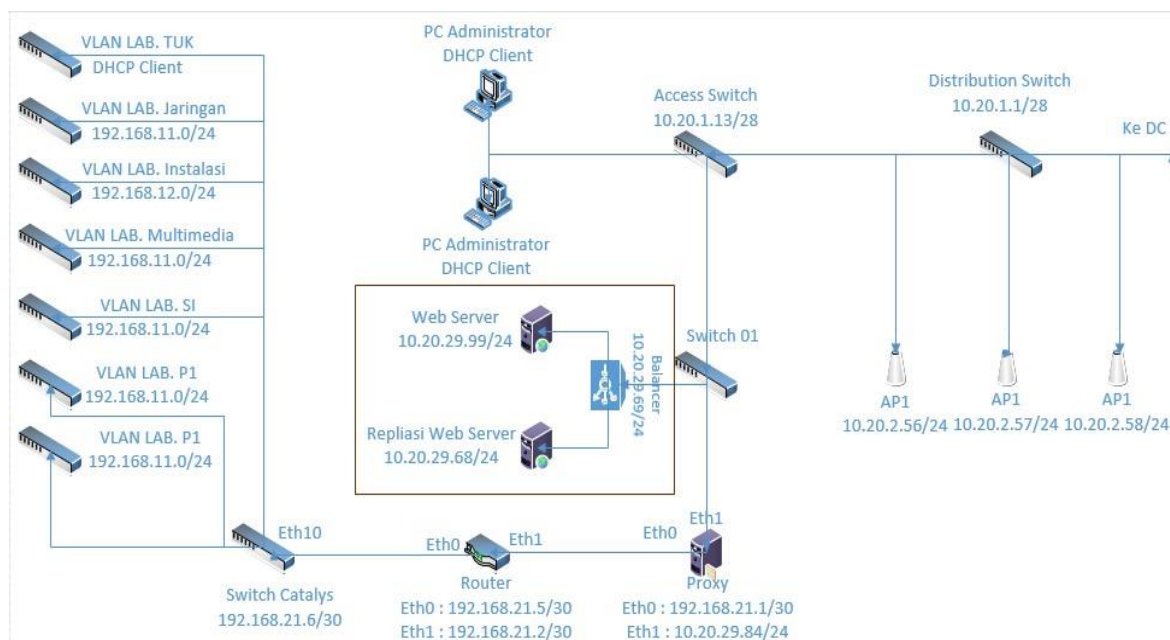
METODE

Dalam implementasi *load balancing web server cluster* diperlukan analisis terhadap beberapa aspek data. Analisis dilakukan terhadap denah ruangan gedung, penempatan *hardware*, dan topologi jaringan. Pada Gedung A Universitas MH.Thamrin Jakarta, aktifitas dilakukan di lantai 2 dan lantai 3. Lantai 2 terdiri dari 7 ruangan kelas digunakan untuk proses belajar mengajar dan untuk aktifitas dosen serta administrasi jurusan. Lantai 3 adalah ruangan Laboratorium praktikum dan ruang laboratorium jaringan komputer. Posisi ruangan server berada pada gedung E lantai 3 adalah sebagai gambar 2.



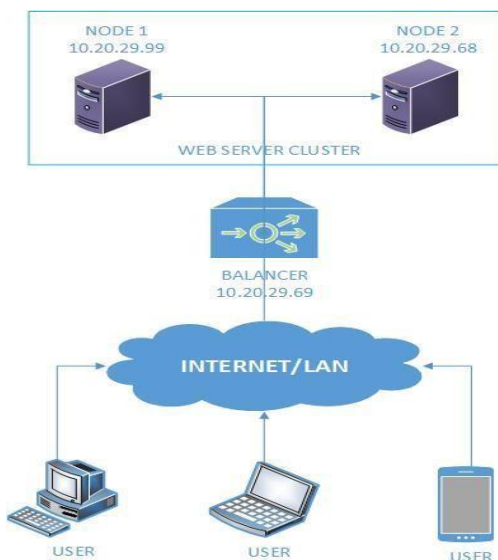
Gambar 2. Denah Gedung Lantai 3

Topologi pada jaringan yang dirancang agar pada saat melakukan pengaturan sistem *clustering* dengan skema *load balancing* menggunakan *nginx* menjadi lebih mudah.



. **Gambar 3.** Rancangan Topologi Jaringan

Rancangan konfigurasi sistem yang dibangun untuk *web server clustering* dengan skema *load balancing* menggunakan *nginx* dengan menggunakan *nginx* sebagai *front server* dari *load balancing* dan menggunakan dua *server* sebagai *web server* yang akan menyediakan informasi.

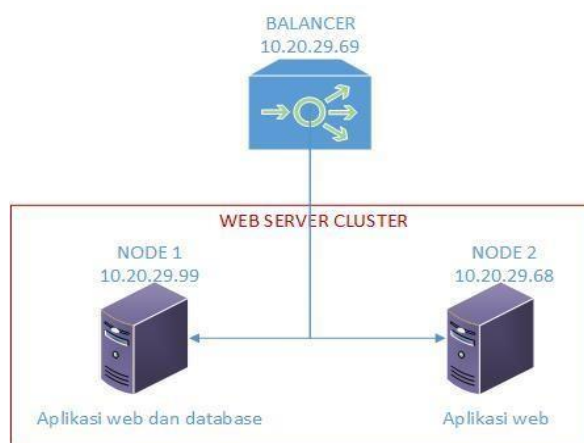


Gambar 4. Web Server Cluster dengan Skema Load Balancing

Rancangan sistem *web server clustering* dengan skema *loadbalancing* menggunakan *nginx*, dengan menggunakan satu *balancer*. *Balancer* berfungsi untuk membagi *request* dari *user* untuk dikerjakan secara bersama oleh kedua *web server*. Sehingga seolah-olah semua request dari user hanya dikerjakan oleh satu *web browser*. *IP address* dari masing- masing server sebagai berikut:

- Balancer : 10.29.20.69
- Web server 1 : 10.29.20.99
- Web server 2 : 10.29.20.68

Rancangan ini membutuhkan dua server. Server pertama untuk *web server master* dan server kedua untuk *web server slave* yang saling bekerja sama sehingga membuat kerja sistem web server tersebut menjadi sebuah sistem tunggal (cluster). Agar perancangan dapat berjalan dengan baik digunakan teknologi *server cloning* agar pada *web server* yang baru ditambahkan sama persis dengan *server master*.



Gambar 5. Sistem Cluster secara Umum

Spesifikasi perangkat lunak dan perangkat keras yang digunakan untuk implementasi *web server clustering* dengan skema *load balancing* menggunakan *nginx* adalah sebagai berikut :

Tabel 1. Spesifikasi Perangkat Lunak (Software)

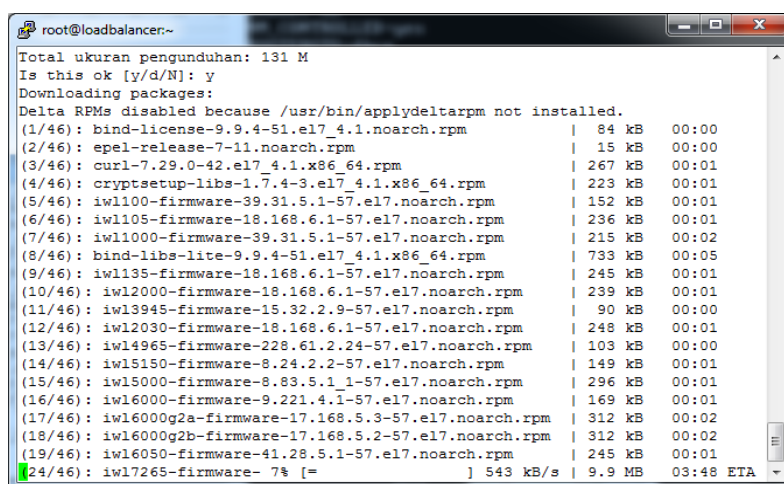
No.	Software	Spesifikasi
1	OS Server Load Balancer	CentOS 7
2	OS Server Node	CentOS 7
3	Aplikasi Web	Webmin Apache PHP Crontab
4	Balancer	Nginx

Tabel 2. Spesifikasi Perangkat Keras (Hardware)

HARDWARE	SPESIFIKASI
BALANCER	• Manufacture : OEM • Processor : Intel (R) Core (TM) 2Quad CPU Q9 • Memory : 1GB • Disk : 1TB
SERVER REPLIKASI	• Manufacture : Asus • Processor : Intel (R) Core (TM)i3-4150 CPU @3.50GHz • Memory : 4GB • Disk : 1TB
SERVER TI	• Manufacture : PowerEdge T620 • Processor : Intel Xeon processorE5-2600 product family • Memory : 8GB • Power Supply : 1100W • Hardisk : 1TB

HASIL DAN PEMBAHASAN

Untuk melakukan *load balancing web server clustering*, terlebih dahulu dilakukan Instalasi Sistem Operasi CentOS 7. Server yang digunakan sebagai *load balancer* dan replikasi *server* menggunakan Personal Computer (PC) dengan *ip address* masing-masing 10.20.29.69 dan 10.20.29.68. Tahapan yang dilakukan untuk melakukan konfigurasi *server load balancing* dan replikasi *server* adalah sebagai berikut :



Gambar 6. Proses Update & Upgrade Server Balancer

Untuk dapat menggunakan *web server clustering* dengan skema *load balancing*, perlu dilakukan Instalasi paket dan *software* pendukung pada *server*. Kemudian dilakukan peng-*update*-an dan peng-*upgrade*-an sistem *server* dengan menggunakan perintah `#yum update && yum upgrade`.

Untuk menginstal *nginx* pada sistem operasi *CentOS* yang digunakan sebagai *balancer*, terlebih dahulu dibuat *file repo* dengan menggunakan perintah `#vi/etc/yum.repos.d/nginx.repo`, selanjutnya ditambahkan baris perintah seperti pada Gambar 7 berikut. Kemudian *file* yang dibuat disimpan.

```
GNU nano 2.3.1      File: /etc/yum.repos.d/nginx.repo

[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=0
enable=1
```

Gambar 7. Membuat *File Repo* untuk *Nginx*

Selanjutnya menginstall ***nginx*** dengan menggunakan perintah `#yuminstall nginx`.

```
* base: mirror.dionipe.net
* extras: mirror.dionipe.net
* updates: mirror.dionipe.net
Sedang menyelesaikan Ketergantungan
--> Menjalankan pemeriksaan transaksi
--> Paket nginx.x86_64 1:1.12.2-1.el7_4ngx akan terpasang
--> Resolusi ketergantungan telah diselesaikan

Ketergantungan-ketergantungan telah Diselesaikan

=====
Package      Arst      Versi      Repositori  Ukuran
=====
Memasang:
nginx        x86_64    1:1.12.2-1.el7_4ngx  nginx      716 k
Ringkasan Transaksi
=====
Instal 1 Paket

Total ukuran pengunduhan: 716 k
Ukuran terpasang: 2.5 M
Is this ok [y/d/N]: y
Downloading packages:
nginx-1.12.2-1.el7_4ngx.x 25% [==== 1 3.3 kB/s | 184 kB  02:39 ETA
```

Gambar 8. Menginstal *Nginx*

Setelah dilakukan *restart service nginx* dengan menggunakan perintah `#systemctl restart nginx`, kemudian dilakukan diuji apakah *server* dapat membalas permintaan *HTTP* dengan membuka alamat *IP* dari *server load balancer* di *web browser*. Apabila pada *web browser* muncul halaman *default* dari *nginx*, maka instalasi *nginx* berhasil.

Welcome to nginx!

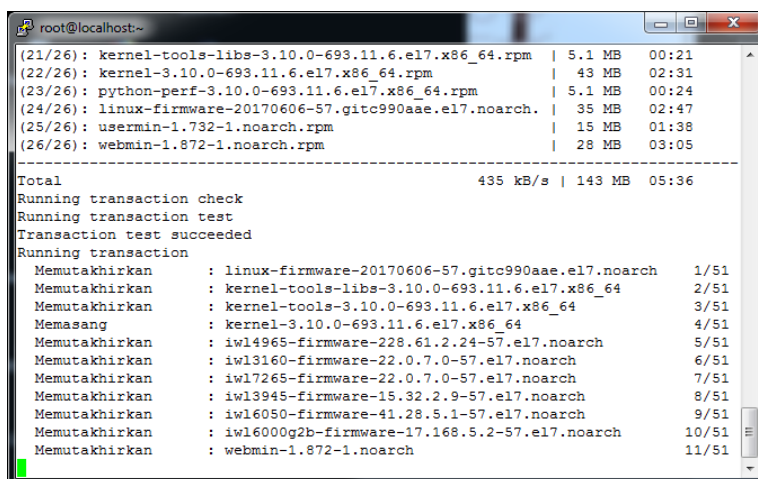
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Gambar 9. Halaman *Default Nginx*

Server replikasi merupakan replikasi dari *server master* (*server web*), dimana yang akan direplikasi merupakan aplikasi dari *web server master*. Sebelum melakukan replikasi perlu diinstallkan paket dan *software* pendukung pada *server*. Kemudian melakukan peng-*update*-an dan peng-*upgrade*-an dengan menggunakan perintah `# yum update && yum upgrade`.



```
root@localhost:~
(21/26): kernel-tools-libs-3.10.0-693.11.6.el7.x86_64.rpm | 5.1 MB 00:21
(22/26): kernel-3.10.0-693.11.6.el7.x86_64.rpm | 43 MB 02:31
(23/26): python-perf-3.10.0-693.11.6.el7.x86_64.rpm | 5.1 MB 00:24
(24/26): linux-firmware-20170606-57.gitc990aae.el7.noarch. | 35 MB 02:47
(25/26): usermin-1.732-1.noarch.rpm | 15 MB 01:38
(26/26): webmin-1.872-1.noarch.rpm | 28 MB 03:05
-----
Total | 435 kB/s | 143 MB 05:36
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Memutakhirkan : linux-firmware-20170606-57.gitc990aae.el7.noarch 1/51
  Memutakhirkan : kernel-tools-libs-3.10.0-693.11.6.el7.x86_64 2/51
  Memutakhirkan : kernel-tools-3.10.0-693.11.6.el7.x86_64 3/51
  Memasang : kernel-3.10.0-693.11.6.el7.x86_64 4/51
  Memutakhirkan : iwl4965-firmware-228.61.2.24-57.el7.noarch 5/51
  Memutakhirkan : iwl3160-firmware-22.0.7.0-57.el7.noarch 6/51
  Memutakhirkan : iwl7265-firmware-22.0.7.0-57.el7.noarch 7/51
  Memutakhirkan : iwl3945-firmware-15.32.2.9-57.el7.noarch 8/51
  Memutakhirkan : iwl6050-firmware-41.28.5.1-57.el7.noarch 9/51
  Memutakhirkan : iwl6000g2b-firmware-17.168.5.2-57.el7.noarch 10/51
  Memutakhirkan : webmin-1.872-1.noarch 11/51
```

Gambar 10. Proses *Update* dan *Upgrade* Server Replikasi

Selanjutnya mengatur *ip address* secara statik. Sama seperti mengatur *ipaddress* pada *server load balancing* diatas, dengan menggunakan perintah `#vi/etc/sysconfig/network-scripts/ifcfg-enp3s0`.

```
root@localhost:~  
TYPE="Ethernet"  
PROXY_METHOD="none"  
BROWSER_ONLY="no"  
BOOTPROTO="static"  
DEFROUTE="yes"  
IPV4_FAILURE_FATAL="no"  
IPV6INIT="yes"  
IPV6_AUTOCONF="yes"  
IPV6_DEFROUTE="yes"  
IPV6_FAILURE_FATAL="no"  
IPV6_ADDR_GEN_MODE="stable-privacy"  
NAME="enp3s0"  
UUID="b75b9441-b921-494e-bf5d-b54428215637"  
DEVICE="enp3s0"  
ONBOOT="yes"  
IPADDR=10.20.29.68  
GATEWAY=10.20.29.1  
NETMASK=255.255.255.0  
DNS=192.168.10.2  
ZONE=public  
~  
~  
~  
"/etc/sysconfig/network-scripts/ifcfg-enp3s0" 20L, 403C
```

Gambar 11. Konfigurasi IP Address pada Server Replikasi

Selanjutnya mengkonfigurasi *nameserver* pada file */etc/resolv.conf* dengan menggunakan perintah *#vi /etc/resolv.conf* seperti pada Gambar 12 berikut.

```
GNU nano 2.3.1      File: /etc/resolv.conf  
# Generated by NetworkManager  
nameserver 127.0.0.1  
nameserver 192.168.10.2
```

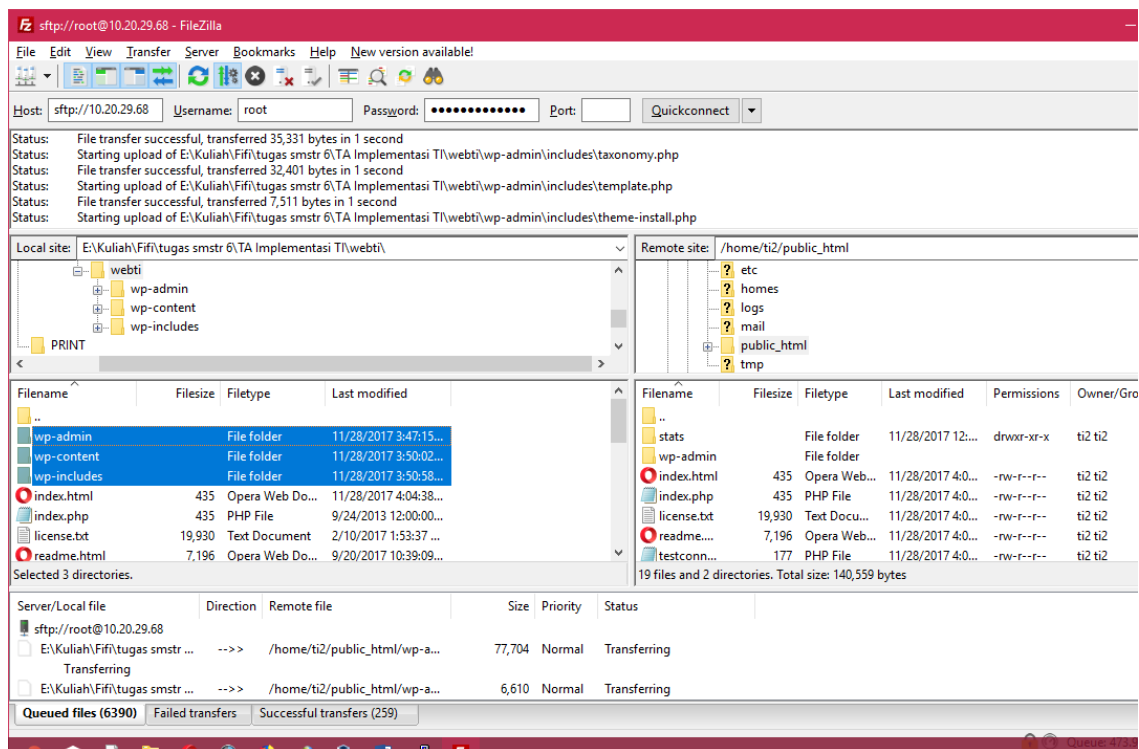
Gambar 12. Konfigurasi File DNS

Setelah mengkonfigurasi *ip address* dan *name server*, maka pada server dilakukan *restart service network* dengan menggunakan perintah *#systemctl restart network*. Kemudian melakukan verifikasi *ip address* menggunakan perintah *#ip addr*.

```
[root@10 ~]# ip addr  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP  
   qlen 1000  
    link/ether 2c:4d:54:51:0e:b7 brd ff:ff:ff:ff:ff:ff  
    inet 10.20.29.68/24 brd 10.20.29.255 scope global enp3s0  
        valid_lft forever preferred_lft forever  
    inet6 fe80::c41e:786e:7087:dc3c/64 scope link  
        valid_lft forever preferred_lft forever  
[root@10 ~]#
```

Gambar 13. Melihat IP Address Server Replikasi

Sebelum menjalankan aplikasi *web*, dilakukan *copy file web SI* dari *server master* menggunakan *filezilla*. Selanjutnya dibuka aplikasi *filezilla*, lalu dimasukkan *ip address*, *username*, *password* serta *port* untuk masuk kedalam *server*. Kemudian dipilih *file web* ti, dan di *copy file* pada direktori */home/tipnp/public_html*.



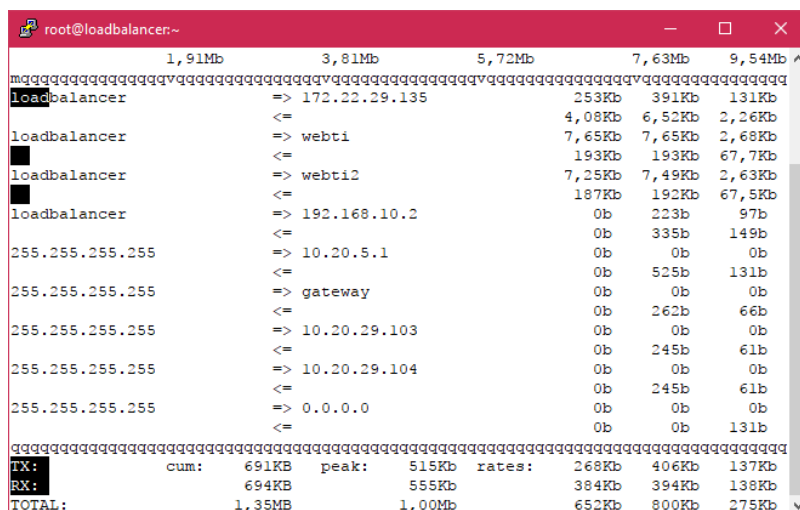
Gambar 14. Men-copy File Web SI pada Server Replikasi

Pengujian merupakan sesuatu yang dilakukan untuk memperlihatkan hasil dari Implementasian yang telah dibuat. Ada 3 pengujian yang dilakukan, yaitu :

1. Pengujian *Load Balancing*

Pengujian *load balancing* dilakukan untuk melihat apakah pembagian kerjadari *web server* terhadap *request* dari *user* sudah berjalandengan baik. Pengujian ini dilakukan dengan menggunakan *iftop* yang merupakan salah satu aplikasi monitoring jaringan untuk melihat *traffic* jaringan secara *realtime*.

Gambar 15 menunjukkan bahwa *balancer* sudah bekerja dengan baik. Hal itu ditunjukkan dengan pembagian kerja kepada *webti* dan *webti2* secara merata. Terlihat tiga kolom paling kanan pada *webti*, disana ada 193Kb, 193Kb, 67,7Kb.



Gambar 15. Monitoring cluster server dengan Iftop

2. Pengujian Waktu Respon

Setelah dapat dipastikan bahwa *load balancing* dapat berjalan dengan baik, selanjutnya dilakukan pengujian waktu respon. Pengujian ini berfungsi untuk melihat berapa lama waktu yang dibutuhkan sistem dalam melayani setiap paket yang datang dengan satuan milidetik. Tahap kedua adalah dengan melakukan pengujian pada *web server cluster*.

1) Web Server Tunggal

Pengujian ini dilakukan dengan simulasi menggunakan program yang mampu mengakses server dengan beberapa user secara bersamaan. Pengujian ini dilakukan dengan melihat waktu respon pada salah satu web server dengan menggunakan bantuan aplikasi *httpperf*.

2) Web Server Cluster

Pada tahap ini pengujiannya hampir sama dengan pengujian pada *web server* tunggal. Namun dalam melakukan pengujian pengaksesan *web server* akan dilakukan melalui *load balancer* 10.20.29.69, sehingga *load balancer* akan langsung membagi beban *request* ke *web server* yang ada dalam *cluster*.

3. Pengujian Ketersediaan (*Availability*) Web server

Pengujian pada tahap ini dilihat dari ketersediaan antara *web server* tunggal dan *web server cluster*. Pengujian ini berfungsi untuk melihat apakah dalam *web server cluster* tetap dapat melayani *request* dan menyediakan ketersediaan data bagi user.

1) *Web Server Tunggal*

Pengujian dilakukan dengan cara me-*non*-aktifkan *service httpd* pada *server* dengan berasumsi bahwa *web server* tunggal telah mati (*down*), kemudian dilakukan pengaksesan ke *web server* tersebut.

2) *Web Server Cluster*

Pengujian ini dilakukan dengan cara yang sama pada pengujian *web server* tunggal yaitu dengan cara me-*non*-aktifkan *service httpd* pada salah satu *web server* yang ada pada *cluster* dengan berasumsi bahwa *web server* telah mati (*down*). Pengujian ini dilakukan pada *load balancer*, apakah *web server cluster* dapat tetap melayani apabila salah satu *web server* mengalami kerusakan atau mati (*down*).

Dari hasil pengujian yang telah dilakukan, hasil analisis dari sistem *load balancing* untuk *web server cluster* adalah :

1. Dengan menggunakan aplikasi *nginx* dengan algoritma *round-robin* sebagai *load balancer*, *request* dari *user* mampu dibagi ke *web server* secara merata.
2. Pengujian dengan menggunakan aplikasi *httperf*, sistem *web server cluster* mampu memberikan layanan data secara lebih cepat dari pada menggunakan sistem *web server* tunggal.
3. Availabilitas *web server* dapat terpenuhi dengan baik, hal ini dibuktikan dengan mematikan salah satu *web server* dan *request* dari *user* masih bisa dilayani dengan baik. Pengujian ini juga membuktikan bahwa kinerja dari *loadbalancing* sudah berjalan dengan optimal.
4. Jika jumlah *concurrent connection*-nya semakin besar maka *respon time* semakin lambat begitu pula sebaliknya, jika *concurrent*-nya semakin kecil maka *respon time* semakin cepat.
5. Implementasi aplikasi *nginx* tidak membutuhkan spesifikasi *hardware* yang tinggi.

KESIMPULAN DAN REKOMENDASI

Setelah dilakukan pengujian, dapat diambil kesimpulan bahwa dengan menggunakan *load balancing* bisa dimanfaatkan sebagai *backup* atau *failover*, jika salah satu *server down* dapat diatasi oleh *server* lainnya. Pelayanan *request* data dari *user* dapat ditangani lebih cepat. Rekomendasi yang dapat diberikan guna pengembangan lebih lanjut maupun melengkapi penelitian ini, yaitu dengan melakukan *load balancing* pada *database server cluster*.

REFERENSI

- Andarrachmi, Annisa. (2012). *Implementasi Load Balancing dan Virtual Machine dengan Algoritma Round Robin untuk Optimalisasi server (Studi Kasus Aplikasi Penerimaan Pegawai BPPT)*.
- CentOS, (2017). URL : <http://www.debian.org/about> Diakses pada : 31 Agustus 2017.
- How to Configure Load Balancing with Nginx, (2017). URL : <https://www.upcloud.com/support/how-to-set-up-load-balancing/>. Diakses pada : 2 September 2017.
- Linux, (2017). URL : <https://www.linux.com/learn/new-user-guides/376-linux-is-everywhere-an-overview-of-the-linux-operating-system>. Diakses pada : 31 Agustus 2017.
- Madcoms. (2010). *Sistem Jaringan Komputer untuk Pemula*. Yogyakarta: ANDI.
- Madcoms. (2013). *Cepat dan Mudah Membangun Sistem Jaringan Komputer*. Madiun: ANDI.
- Muliyantoro, Halim Setya. (2013). *Penerapan Metode Load-Balancing Clusters Pada Database Server Guna Peningkatan Kinerja Pengaksesan Data*.
- Nasution, A. H. (2011). *Komparasi Algoritma Penjadwalan Pada Layanan Terdistribusi Load Balancing LVS VIA NAT*.
- Nginx, (2017). URL : <https://www.nginx.com/resources/wiki/>. Diakses pada : 14 Juli 2017.
- Putra, G. B R. (2021). *Implementasi Mysql Cluster Pada Basis Data Terdistribusi*.
- Rabur, J. A., Purwadi, Joko., Raharjo, Willy S. (2012). *Implementasi Load Menggunakan Balancing Web Server Metode LVS-NAT*.

-
- Rosalia1, Maya., Munadi, Rendy., Mayasari, Ratna. (2016). *Implementasi HighAvailability Server Menggunakan Metode Load Balancing danFailover Pada Virtual Web Server Cluster*.
- Siddiq, Fajar., Wahono, C. A. U., Alfiansyah, S. H. (2020). *Penerapan MySQL Cluster untuk Membangun Replikasi Dua Arah Basis Data Terdistribusi dengan Penerapan Metode Fail Over dan Load Balancing*.
- Sofana, Iwan., (2012), *CISCO CCNA dan Jaringan Komputer*, Informatika, Bandung.
- Suparwita, P. E., (2012). "Implementasi Sistem Backup Otomatis Virtual Private Server Dengan Crontab," *Jurnal Elektronika Ilmu Komputer*, vol. Vol I, pp.29-34, 2012.
- Wahana Komputer. (2011). *Administrasi Jaringan dengan Linux Ubuntu11*. Yogyakarta: ANDI.
- Webmin, (2017). URL : <http://www.webmin.com/> Diakses pada : 19 November 2017.
- Yusuf, Effendi., Riza, Tengku A., Ariefianto,Tody. (2013). *Implementasi Teknologi Load Balancer Dengan Web Server Nginx Untuk MengatasiBeban Server*.