JUSTIN
Jurnal Sistem dan Teknologi Informasi
p-ISSN : 2460-3562 / e-ISSN : 2620-8989

DOI: 10.26418/justin.v8i2.31965
Vol. 8, No. 2, April 2020

Analysis of the Use of the Failover Clustering Method to Achieve High Availability on a Web Server (Case Study: Informatics Department Building)

Yulizar Pribadi[#1], Arif Bijaksana PN[#2], M. Azhar Irwansyah[#3]

[#]*Program Studi Informatika Fakultas Teknik Universitas Tanjungpura*
Jl. Prof Dr H. Hadari Nawawi, Kota Pontianak, 78115

[1]yulizarpribadi.ft@student.untan.ac.id

[2]arifbpn@informatika.untan.ac.id

[3]azharirwansyah@informatika.untan.ac.id

Abstrak

Seiring berkembangnya kebutuhan pengguna maupun peningkatan permintaan pada suatu website maka ketersediaan dari web server yang kuat dan handal merupakan permasalahan yang dihadapi dalam memenuhi kebutuhan akan data maupun layanan. Namun dalam praktiknya web server ketika diakses oleh client kadang terjadi kegagalan. Hal ini disebabkan oleh terjadinya down pada web server dan tidak ada backup dari server lain yang langsung menggantikannya. Solusi yang dapat diimplementasikan untuk mengatasi masalah tersebut adalah dengan menciptakan high availability server menggunakan teknologi web server clustering lebih tepatnya dengan mengimplementasikan metode failover clustering pada web server. Tujuan dari penelitian adalah melihat pengaruh penggunaan metode failover clustering dalam mencapai layanan high availability pada web server gedung Informatika. Berdasarkan penelitian yang telah dilakukan, dapat disimpulkan sebagai berikut. (1) Sistem failover clustering yang dibangun dapat bekerja dengan baik sesuai dengan konsep kerjanya. (2) Hasil pengujian availability dapat memberikan ketersediaan layanan yang lebih baik saat terjadi kegagalan. Nilai availability yang didapatkan dari perhitungan data pengujian sebesar 99,90%. (3) Jumlah workload yang dapat dilayani oleh web server secara bersamaan yang didapat selama pengujian menunjukkan terdapat sedikit perbedaan pada performa web server yang mempengaruhi jumlah workload dari web server. (4) Hasil pengujian QoS menunjukan nilai keseluruhan pada web server dengan failover dan web server tanpa failover dengan indeks yang sama yaitu memuaskan. Namun terdapat sedikit perbedaan pada nilai pengujian tiap parameter dimana web server tanpa failover menunjukkan performa yang lebih baik. (5) Konfigurasi failover clustering menunjukkan hasil yang baik dilihat dari segi availability pada saat terjadi kegagalan.

Kata kunci: Availability, Failover Clustering, Web server, QoS, Workload

Analysis of the Use of the Failover Clustering Method to Achieve High Availability on a Web Server (Case Study: Informatics Department Building)

Abstract

Along with the development of user needs and increasing demand on a website, the availability of a strong and reliable web server is a problem faced in meeting the needs for data and services. But in practice, the webserver, when accessed by a client, sometimes fails. This is caused by a down on the webserver, and there are no backups from other servers that directly replace it. The solution that can be implemented to overcome this problem is to create a high availability server using web server clustering technology more precisely by implementing the failover clustering method on the webserver. The purpose of this study is to look at the effect of using failover clustering methods in achieving high availability services on the Informatics building web server. Based on research that has been done, it can be concluded as follows. (1) A failover clustering system that is built can work well in accordance with its work concept. (2) Availability test results can provide better service availability in the event of failure. The availability value obtained from the calculation of testing

data is 99.90%. (3) The number of workloads that can be served by the webserver simultaneously obtained during the testing shows that there is a slight difference in the performance of the webserver that affects the amount of workload from the webserver. (4) QoS test results show the overall value on the webserver with failover and web server without failover with the same index, which is satisfactory. But there is a slight difference in the test value of each parameter where the webserver without failover shows better performance. (5) Configuration of failover clustering shows good results in terms of availability when a failure occurs.
Keywords: Availability, Failover Clustering, Web server, QoS, Workload

I. PENDAHULUAN

Perkembangan teknologi memberikan pengaruh besar bagi kehidupan salah satunya dalam penyampaian suatu informasi. Pentingnya suatu informasi menjadi salah satu kebutuhan pokok dalam kehidupan sehari-hari. Saat ini *internet* sudah menjadi kebutuhan pokok di berbagai bidang kehidupan. *Internet* telah banyak dimanfaatkan di berbagai bidang sebagai media penyebaran informasi dan komunikasi dengan biaya yang relatif murah, jangkauan yang sangat luas dan efisien. Kebutuhan akan koneksi *internet* semakin meningkat seiring banyaknya perangkat yang bisa digunakan untuk menjelajahi dunia maya. Sejak tahun 2000-an, *web* menjadi media berorientasi bisnis dan antarmuka yang lebih disukai untuk sistem informasi terbaru [1].

World Wide Web atau *web* merupakan salah satu layanan dari *internet*. *Web* adalah suatu cara mengakses informasi melalui media *internet*. *Web* bisa juga dikatakan sebagai suatu model berbagi informasi yang dibangun di atas media *internet* yang menggunakan protokol HTTP untuk mengirimkan data. Protokol biasanya berbentuk sebuah *software* yang mengatur komunikasi data tersebut [2]. Data tersebut tersebar di seluruh penjuru dunia disimpan dalam media penyimpanan berupa *server*. *Web server* bertanggung jawab melayani permintaan HTTP dari aplikasi *client* yang dikenal dengan *web browser* atau sebuah program yang dirancang untuk mengambil informasi dari *server* komputer pada suatu jaringan *internet* maupun *intranet* [3]. *Web server* akan mencari data dari *Uniform Resource Locator* (URL) yang diminta dan mengirimkan kembali hasilnya dalam bentuk halaman-halaman *web* yang umumnya berbentuk dokumen *Hypertext Markup Language* (HTML) dan semua isi dari suatu situs ke komputer *client*.

Linux merupakan sistem operasi yang banyak dipakai untuk kebutuhan *server*. Sistem itu sendiri ditemukan oleh mahasiswa berkebangsaan finlandia, yaitu linus torvalds yang merupakan seorang hobi komputer. Dengan sifatnya yang open source menjadikannya cukup populer di kalangan IT. Selain itu, linux juga merupakan sistem operasi yang stabil dan cepat yang menjadikannya sangat cocok untuk digunakan bahkan pada komputer *server*. Namun dengan kelebihan tersebut tidak menjadikannya kebal terhadap faktor luar yang dapat menyebabkan layanan terhenti.

Tersedianya data maupun layanan merupakan kebutuhan yang bersifat sangat penting terlebih lagi di dalam perusahaan maupun instansi. Salah satu sarana yang digunakan untuk kebutuhan tersebut adalah jaringan komputer yang mampu menggabungkan tidak hanya computer melainkan piranti seperti *server*, *modem*, *router*, *printer*, dan sebagainya [4]. Bergantungnya suatu perusahaan atau instansi terhadap infrastruktur jaringan komputer dan seiring dengan berkembangnya kebutuhan pengguna maupun peningkatan permintaan pada suatu situs *web* maka ketersediaan dari *web server* yang kuat dan handal merupakan permasalahan yang dihadapi oleh perusahaan atau instansi dalam memenuhi kebutuhan akan data maupun layanan.

Mengingat fungsi yang dimiliki *server* yaitu memberikan layanan kepada *client* maka *server* dituntut untuk bisa memberikan layanan secara *real-time* (terus-menerus) terhadap permintaan / *request* dari semua *client*. Namun, dalam praktiknya *web server* ketika diakses oleh *client* kadang terjadi kegagalan. Hal tersebut disebabkan karena di sisi *server* terjadi *failure* (kegagalan). *Failure* itu sendiri disebabkan karena *server down* dan tidak ada *backup* dari *server* lain yang langsung menggantikan ketika *master server* (*server* utama) mati. Hal tersebut menunjukkan pentingnya sistem *server* yang terus-menerus berfungsi dalam arti lain memiliki sifat *high availability* (ketersediaan yang tinggi).

Salah satu solusi yang dapat diimplementasikan untuk mengatasi masalah tersebut adalah dengan menciptakan *high availability server* menggunakan teknologi *web server clustering*. Dalam dunia komputer yang dimaksud dengan *server clustering* adalah menggunakan lebih dari satu *server* yang menyediakan *redundant interconnections*, sehingga *client* hanya mengetahui ada satu sistem *server* yang tersedia dalam computer. Kompunen *cluster* biasanya saling terhubung melalui sebuah interkoneksi yang sangat cepat atau bisa juga melalui jaringan local (LAN) [5]. Selain itu, *cluster* juga merupakan sekelompok mesin yang bertindak sebagai sebuah entitas tunggal untuk menyediakan sumber daya dan layanan ke jaringan [6]. Pada *server clustering* terdapat metode *failover clustering* yang menyediakan solusi *high availability server* dimana metode tersebut akan berjalan jika terjadi kegagalan pada perangkat keras yang menyebabkan *server* mati total sehingga *server* lain akan mengambil alih fungsi dari *server* yang mati.

Konsep konfigurasi *failover clustering* adalah membuat satu *server* sebagai *master server* dan *server* yang lain menjadi *slave server* dimana saat *server* dalam keadaan normal *master server* menangani semua *request* dari *client*. *Slave server* akan mengambil alih tugas *master server* apabila *master server* tidak berfungsi atau *down*. Kegagalan pada sistem *server* tidak akan disadari oleh *client* karena tersedianya *server* lain sebagai *backup* sehingga dapat mengatasi kegagalan atau *failure* pada *web server* itu sendiri.

Eksperimental merupakan pendekatan yang digunakan untuk mencari pengaruh perlakuan tertentu terhadap yang

Korespondensi : Yulizar Pribadi

lain dalam kondisi yang terkendali. Pengaruh yang dimaksud adalah pengaruh penggunaan metode *failover clustering* terhadap parameter yang ditentukan. Parameter yang diukur pada penelitian ini adalah *availability*, *workload*, dan *quality of service* (QoS).

Berdasarkan pemaparan diatas maka dibuatlah *web server clustering* dengan mengimplementasikan metode *failover clustering* sehingga dapat terlihat performa dari *web server* yang dibangun dari parameter yang diukur agar tercipta sistem *web server* yang memiliki ketersediaan tinggi.

II. URAIAN PENELITIAN

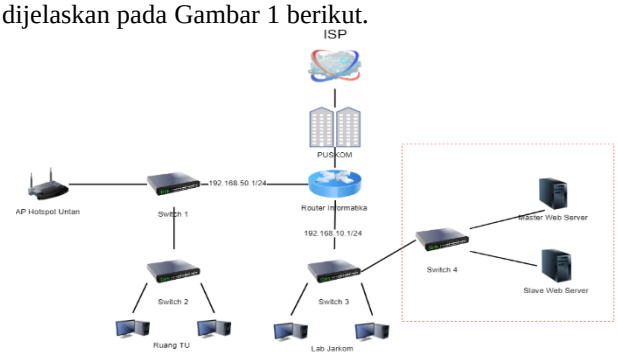
A. Penelitian Terkait

Penelitian ini bukanlah hal yang baru dilakukan adapun penelitian sejenis yang pernah dilakukan oleh Akhyar Muchtar (t.thn) melakukan penelitian dengan judul "Implementasi *Failover Clustering* Pada Dua Platform yang Berbeda Untuk Mengatasi Kegagalan Fungsi Server" untuk membuat rancangan *failover clustering* dalam menangani masalah kegagalan fungsi server. [7] Irfani Hernawan Sulistyanto (2015) melakukan penelitian yang berjudul "Implementasi *High Availability Server* dengan Teknik *Failover Virtual Computer Cluster*" untuk mengatasi kegagalan fungsi *server* dengan menggunakan VMware Workstation 11 sebagai *platform* simulasinya. [8] Prayudi Aditya Nugraha (2016) melakukan penelitian yang berjudul "Rancang Bangun *Web Server* Berbasis Linux dengan Metode *Load Balancing* Pada Gedung Jurusan Informatika Untan" untuk membangun *web server* dengan menggunakan *load balancing* pada *server cluster*. [9]

Penelitian ini dilakukan pada gedung jurusan Informatika Universitas Tanjungpura. Penelitian ini bertujuan untuk melihat pengaruh penggunaan metode *failover clusering* dalam mencapai *high availability* pada *web server* gedung jurusan Informatika Universitas Tanjungpura.

B. Pengembangan Arsitektur Jaringan

Pada tahap ini dilakukan pengembangan arsitektur jaringan untuk meletakkan posisi *server* yang terhubung pada jaringan yang sudah ada. Pengembangan arsitektur jaringan yang akan diterapkan pada jaringan Informatika dijelaskan pada Gambar 1 berikut.



Gambar. 1 Pengembangan arsitektur jaringan

- Gedung Informatika mendapatkan akses *internet* dari puskom sebesar 5 Mbps.
- Akses *internet* yang diperoleh kemudian dibagi ke seluruh ruangan gedung Informatika menggunakan mikrotik melalui *routerboard* yang diatur sesuai kebutuhan
- *Routerboard* membagi akses *internet* melalui *port-port* yang ada untuk AP Hotspot Untan, ruang TU, dan laboratorium jaringan komputer.
- Pengembangan dilakukan dengan meletakkan *Web server cluster* pada jaringan gedung Informatika dengan menghubungkannya melalui *switch* jaringan laboratorium jaringan komputer menggunakan IP *Dynamic*.

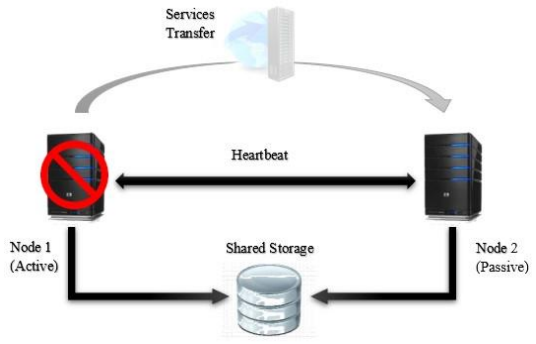
C. High-Availability Clusters

High-availability Clusters, yang juga disebut *failover cluster* pada umumnya diimplementasikan untuk tujuan meningkatkan ketersediaan layanan yang disediakan oleh kluster. Komputer kluster adalah sekumpulan computer independent yang bekerja secara bersama sebagai sumber daya komputasi tunggal yang terintegrasi dan terlihat oleh klien seolah computer tersebut adalah satu buah unit computer [10]. Dibutuhkan dua buah *node* sebagai syarat minimum suatu kluster untuk dapat melakukan redudansi [11].

Pada klasifikasi kluster ini, terdapat mode *active/passive failover*. Pada mode ini terdapat dua komponen, yang satu menjadi komponen atau *node* aktif dan yang lainnya pasif. Pada saat *node* aktif mengalami kegagalan, *node* pasif akan mengambil alih tugas *node* aktif [12].

D. Heartbeat

Heartbeat merupakan aplikasi dasar untuk Linux-HA (*Linux High-Availability*). *Heartbeat* bertujuan untuk terus mem-*polling server* dalam konfigurasi *cluster* untuk memastikan bahwa mereka sudah *up* dan merespon [13]. *Heartbeat* berfungsi untuk mempromosikan *server* aktif yang mengalami gangguan saat digunakan. *Server* aktif saat mengalami gangguan atau *down heartbeat* akan mempromosikan ke *server* kedua untuk mengambil alih dengan memindahkan IP virtual ke *server* kedua. Selain itu juga menangani *service-service* apa saja yang akan dijalankan pada saat *server* menjadi aktif [14]. Cara kerja *heartbeat* dapat dilihat pada Gambar 2 berikut.



Gambar. 2 Ilustrasi *Heartbeat*

E. Web server

Web server merupakan sebuah bentuk *server* yang khusus digunakan untuk menyimpan halaman *website* atau *homepage*. Dalam melakukan permintaan suatu halaman pada suatu situs *web*, *browser* melakukan koneksi ke suatu *server* dengan protokol HTTP. *Web server* yang terkenal adalah Apache. *Web server* merupakan *software* yang menjadi tulang punggung dari *World Wide Web* [15].

F. Availability

Berdasarkan dokumen ISO 2382-14 (1997), *availability* dapat didefinisikan sebagai "kemampuan sebuah alat untuk berada dalam kondisi siap pakai sesuai fungsi yang diinginkan pada waktu tertentu atau kapanpun dalam *interval* waktu tertentu, diasumsikan bahwa sumber eksternalnya bila diperlukan adalah tersedia". Secara garis besar *availability* merupakan nilai presentase jumlah waktu suatu jaringan mampu memberikan layanan dibandingkan dengan jumlah waktu yang diharapkan.

G. Workload

Workload atau beban kerja merupakan jumlah *request* yang dapat dilayani suatu *server* dalam waktu tertentu. Berdasarkan dokumen BKN nomor 37 (2011) tentang pedoman penataan pegawai negeri sipil, beban kerja adalah sejumlah target pekerjaan atau target hasil yang harus dicapai dalam satu satuan waktu tertentu. Pengujian *workload* dilakukan untuk mengetahui kemampuan suatu *server* dalam menangani sejumlah *request* dari *client*. Hal ini bertujuan untuk mengetahui batasan suatu *server* dalam jumlah *request* yang dapat ditangani dalam memberikan pelayanan kepada *client.*

H. QoS (Quality of service)

Qos (*Quality of service*) merupakan sebuah sistem arsitektur *end to end* dan bukan merupakan *feature* yang dimiliki oleh jaringan. Qos adalah kemampuan suatu jaringan untuk menyediakan layanan yang baik dengan menyediakan *bandwidth*, mengatasi *jitter* dan *delay*. [16].

Beberapa parameter yang dapat digunakan dalam menentukan Qos yaitu.

1) *Throughput: Throughput* dalam jaringan telekomunikasi merupakan rata-rata pengiriman sukses dalam suatu pengiriman (satuan bps) [17]. *Throughput* merupakan jumlah total kedatangan paket yang sukses yang diamati pada tujuan selama *interval* waktu tertentu dibagi oleh durasi *interval* waktu tersebut [18]. Nilai *throughput* sesuai dengan versi TIPHON (*Telecommunication and internet protocol harmonization over network*) dapat dilihat pada Tabel 1.

TABEL I
THROUGHPUT

Kategori *Throuhput*	***Throughpput***	**Indeks**
Sangat Bagus	100 %	4
Bagus	75 %	3
Sedang	50 %	2
Jelek	< 25 %	1

Persamaan perhitungan untuk mengukur nilai *throughput* dalam satuan persen (%), yaitu :

$$Throughput = \frac{paket\ data\ yang\ diterima}{lama\ pengamatan} \quad (1)$$

$$Throughput(\%) = \frac{Throughput}{Bandwidth} \quad (2)$$

2) *Packet loss: Packet loss* merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang. Umumnya perangkat jaringan memiliki *buffer* untuk menampung data yang diterima. Jika terjadi kongesti dalam waktu yang cukup lama, *buffur* akan penuh dan data baru tidak dapat diterima [18]. Pengujian *packet loss* dilakukan untuk mengetahui seberapa banyak data yang hilang ketika *server* diberikan gangguan. Nilai *packet loss* sesuai dengan versi TIPHON (*Telecommunication and internet protocol harmonization over network*) dapat dilihat pada Tabel 2.

TABEL II
PACKET LOSS

Kategori *Degredasi*	***Packet loss***	**Indeks**
Sangat Bagus	0 %	4
Bagus	3 %	3
Sedang	15 %	2
Jelek	25 %	1

Persamaan perhitungan untuk mengukur nilai *packet loss* :

$$Packet\ loss = \frac{(paket\ dikirim - paket\ diterima)\ x100\%}{paket\ dikirim} \quad (3)$$

3) *Delay (Latency): Delay* adalah waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan. *Delay* dapat dipengaruhi oleh jarak, media fisik, kongesti atau juga waktu proses yang lama [18]. Nilai *delay* sesuai dengan versi TIPHON (*Telecommunication and internet protocol harmonization over network*) dapat dilihat pada Tabel 3.

TABEL III
DELAY

Kategori *Delay*	***Besar Delay***	**Indeks**
Sangat Bagus	<150 ms	4
Bagus	150 s/d 300 ms	3
Sedang	300 s/d 450 ms	2
Jelek	>450 ms	1

Persamaan perhitungan untuk mengukur nilai *delay* :

$$Delay\ \text{rata-rata} = \frac{lama\ pengamatan}{paket\ diterima} \quad (4)$$

4) *Jitter: Jitter* atau disebut juga variasi kedatangan paket terjadi akibat variasi-variasi dalam panjang antrian, dalam waktu pengolahan data, dan juga dalam waktu penghimpunan ulang paket-paket di akhir perjalanan. *Jitter* yang berhubungan erat dengan *latency* menunjukkan banyaknya variasi *delay* pada transmisi data jaringan. *Delay* antrian pada *router* dan *switch* dapat menyebabkan *jitter* [18]. Terdapat empat kategori penurunan performansi jaringan berdasarkan nilai *peak jitter* sesuai dengan versi TIPHON (*Telecommunication and internet protocol harmonization over network*) yang dapat dilihat pada Tabel 4.

TABEL IV
JITTER

Kategori *Delay*	***Besar Delay***	**Indeks**
Sangat Bagus	0 ms	4
Bagus	0 s/d 75 ms	3
Sedang	75 s/d 125 ms	2
Jelek	125 s/d 225 ms	1

Persamaan perhitungan untuk mengukur nilai *jitter* :

$$Jitter = \frac{total\ variasi\ delay}{total\ paket\ diterima} \quad (5)$$

Total variasi *delay* diperoleh dari :

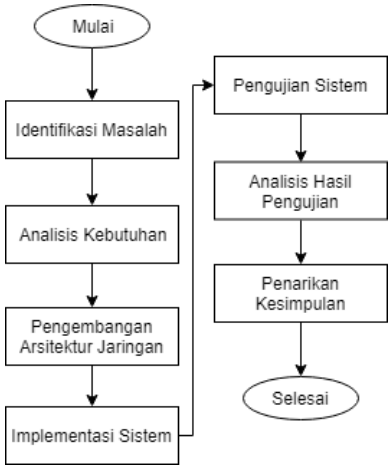
Total variasi *delay*=lama pengamatan–*delay* rata-rata (6)

Berdasarkan nilai parameter-parameter QoS maka untuk selanjutnya dapat dilakukan perhitungan terhadap performansi. Tabel 5 menunjukkan kualitas QoS versi TIPHON (*Telecommunication and internet protocol harmonization over network*).

TABEL V
QUALITY OF SERVICE

Nilai	***Persentase (%)***	**Indeks**
3,8 - 4	95 – 100	Sangat memuaskan
3 – 3,79	75 – 94,75	Memuaskan
2 – 2,99	50 – 74,75	Kurang memuaskan
1 – 1,99	25 - 49,75	Jelek

III. METODOLOGI PENELITIAN



Gambar. 3 Diagram alih penelitian

Pada Gambar 3 dapat dilihat bahwa langkah-langkah penelitian yang akan dilakukan meliputi identifikasi masalah sehingga dapat dilihat kemungkinan yang dapat dilakukan untuk pembuatan sistem *web server*. Kemudian dilanjutkan dengan analisis kebutuhan untuk menentukan kebutuhan yang diperlukan dalam penelitian serta dilakukan pengembangan arsitektur jaringan yang sesuai dengan kebutuhan.

Berdasarkan arsitektur jaringan yang telah dikembangkan selanjutnya dilakukan implementasi sistem dengan pembangunan *web server* yang dilakukan dengan konfigurasi IP *Address*, *update* dan *upgrade* karnel linux, konfigurasi Apache2, instalasi MySQL *server*, instalasi PHP, konfigurasi *Heartbeat*, dan terakhir konfigurasi DRBD untuk melakukan replikasi melalui jaringan LAN atau disebut RAID-1 *over network* [19].

Melalui sistem yang telah dibangun selanjutnya dilakukan pengujian terhadap sistem *web server*. Pengujian dilakukan pada *web server* tunggal dan *web server* kluster untuk melihat perbedaan diantara keduanya yang dilakukan menggunakan *tools* siege dan iperf. Iperf dapat digunakan untuk menetukan *server* mana yang tidak dapat mencapai *throughput* maksimum [20]. Hasil pengujian akan dianalisis yang selanjutnya akan dilakukan penarikan kesimpulan pada tahap akhir.

IV. HASIL DAN PEMBAHASAN

A. Pengujian Sistem

Pada tahap ini dilakukan pengujian terhadap *web server* apakah dapat berfungsi sesuai dengan perancangan atau tidak dengan menerapkan metode *failover*. Tahap pengujian ini dilakukan untuk memastikan bahwa sistem yang telah dikonfigurasi dapat berjalan sesuai perancangan menguji pada *secondary server* dan cluster web*server*. Adapun pengujiannya yakni sebagai berikut.

1) Pengujian Availability: Pengujian *availability* adalah pengujian terhadap ketersediaan layanan dari *web server*. Pada pengujian ini akan dilihat apakah *web server* dapat memberikan layanan ketika terjadi *down*. Nilai dari *availability* akan didapatkan dengan persamaan berikut.

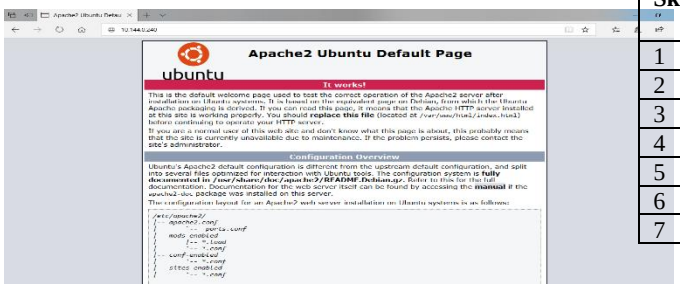
$$Availability = \frac{uptime}{uptime + downtime} x100\% \quad (7)$$

Pengujian ini bertujuan untuk melihat seberapa bagus layanan yang dapat diberikan oleh *web server* meskipun terjadi kegagalan pada *web server* melalui beberapa skenario pengujian. Adapun skenario yang diberikan adalah sebagai berikut.

- Skenario 1: kedua *web server* dalam posisi hidup kemudian *primary server* dimatikan.
- Skenario 2: *primary server* dalam kondisi mati dan *secondary server* dalam posisi hidup kemudian *primary server* dihidupkan.
- Skenario 3: kedua *web server* dalam kondisi hidup lalu *primary server* dimatikan setelah itu *secondary server* juga dimatikan. Kemudian kedua *web server* dihidupkan secara bersamaan.
- Skenario 4: kedua *web server* dalam kondisi mati kemudian *primary server* dihidupkan.
- Skenario 5: kedua *web server* dalam kondisi mati kemudian *secondary server* dihidupkan.
- Skenario 6: kedua *web server* dalam kondisi hidup lalu *primary server* dimatikan setelah itu *secondary server* dimatikan. Kemudian *primary server* dihidupkan setelah itu *secondary server* dihidupkan.
- Skenario 7: kedua *web server* dalam kondisi hidup lalu *primary server* dimatikan setelah itu *secondary server* dimatikan. Kemudian *secondary server* dihidupkan setelah itu *primary server* dihidupkan.

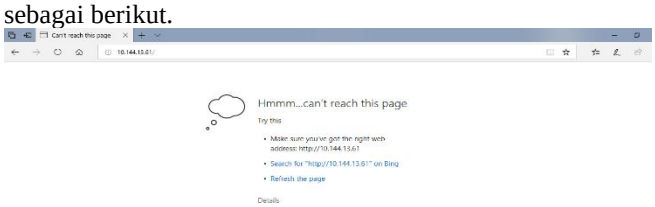
Pengujian dilakukan dengan skenario *down* pada *web server*. *Primary web server* akan dibuat seolah-olah *down* dengan cara mematikannya kemudian dalam kondisi *down* tersebut dilakukan percobaan akses website pada melalui *client*. Adapun hasil pengujian akses website pada *web*

server dari *client* ketika diberikan gangguan adalah sebagai berikut.



Gambar. 4 Hasil pengujian akses website pada *web server* dengan *failover*

Gambar 4 merupakan hasil pengujian dari *client* pada *web server* dengan *failover* untuk membuktikan bahwa layanan *web server* tersedia. Pada saat *primary server* diberikan gangguan *client* masih bisa mengakses *web server* meskipun terdapat waktu beberapa detik *web server* tidak bisa diakses. Adapun ketika diujikan pada *web server* tanpa menggunakan *failover* didapatkan hasil sebagai berikut.



Gambar. 5 Hasil pengujian akses website pada *web server* tanpa *failover*

Gambar 5 merupakan hasil pengujian dari *client* pada *web server* tanpa *failover* yang menunjukkan bahwa *website* tidak dapat diakses. Berdasarkan hasil pengujian tersebut menunjukkan bahwa layanan *web server* tanpa *failover* tidak tersedia ketika terjadi *down* saat diberikan gangguan pada *web server*.

Waktu *web server* tidak bisa diakses tersebut merupakan *down*time yang terjadi selama *web server* mengalami gangguan. Pengujian *down*time dilakukan untuk mengukur waktu yang diperlukan suatu *web server* untuk mengembalikan layanannya ketika terjadi gangguan. Pengujian dilakukan dengan memberikan gangguan pada *web server* sesuai skenario yang telah ditetapkan agar didapatkan nilai *down*time. Pengujian dilakukan sebanyak lima kali untuk mendapatkan nilai rata-rata *down*time dengan menggunakan *tool* siege. Pengujian dilakukan dengan mengirimkan sejumlah *request* selama pengujian kemudian *down*time dapat ditentukan melalui jumlah *request* yang gagal dibagi jumlah rata-rata eksekusi *request* perdetik yang didapat melalui hasil capture *tool* siege. Adapun hasil pengujian yang diperoleh melalui *tool* siege ditampilkan pada Tabel 6 berikut.

TABEL VI
HASIL PENGUJIAN *DOWN*TIME

Skenario	*Downtime* tiap pengujian (*second*)					Rata-rata
	Ke-1	Ke-2	Ke-3	Ke-4	Ke--5	
1	1,35	1,59	1,33	1,5	1,4	1,434
2	1,77	1,45	1,66	1,38	1,52	1,556
3	27,24	11,47	44,82	44,95	27	31,1
4	162,9	283,7	172,7	216,9	129,7	193,18
5	253,4	58,3	80,38	52,79	96,45	108,3
6	32,22	110,4	121,5	37,08	165,9	93,42
7	72,79	147,6	140,3	232,9	155,9	149,9

Tabel 6 merupakan hasil pengujian *down*time melalui beberapa skenario. Hasil pengujian menunjukkan perpindahan dari *primary server* ke *secondary server* yang diperoleh melalui pengujian skenario 1 lebih cepat yaitu selama 1,434 detik sedangkan perpindahan dari *secondary server* ke *primary server* yang diperoleh melalui pengujian skenario 2 selama 1,556 detik. Pengujian skenario 3 menunjukkan hasil bahwa ketika *web server* dihidupkan secara bersamaan dari keadaan *down* waktu yang dibutuhkan untuk up kembali lebih cepat dengan waktu selama 31,1 detik daripada saat *web server* dihidupkan satu persatu pada pengujian skenario 6 dan skenario 7. Waktu yang dibutuhkan saat hanya *primary server* yang dihidupkan dari keaadaan *down* melalui skenario 4 lebih lama yaitu selama 193,18 detik dibandingkan saat hanya *secondary server* yang dihidupkan dari keadaan *down* melalui skenario 5 yaitu selama 108,3 detik. Total *down*time yang diperoleh melalui skenario pengujian yaitu 82,7 detik. Berdasarkan hasil pengujian menggunakan *tool* siege tersebut juga didapatkan nilai respon time yaitu waktu sesaat setelah terjadi *down* hingga *secondary server* mendapatkan sinyal untuk mengambil alih layanan.

TABEL VII
HASIL PENGUJIAN *RESPOND TIME*

Skenario	*Downtime* tiap pengujian (*second*)					Rata-rata
	Ke-1	Ke-2	Ke-3	Ke-4	Ke--5	
1	0,07	0,07	0,16	0,07	0,09	0,09
2	0,13	0,06	0,07	0,07	0,07	0,08
3	0,03	0,08	0,05	0,22	0,06	0,09
4	0,03	0,01	0,02	0,01	0,03	0,02
5	0,10	0,06	0,05	0,12	0,06	0,08
6	0,05	0,11	0,06	0,02	0,20	0,09
7	0,07	0,15	0,10	0,29	0,23	0,17

Tabel 7 menunjukkan nilai *respond time* yang didapatkan melalui pengujian menggunakan *tool* siege. Nilai *respond time* terkecil diperoleh pada skenario 4 dengan nilai 0,02 detik dan respon time terbesar diperoleh pada skenario 7 dengan nilai 0,17 detik. Perbedaan hasil tersebut terjadi dikarenakan adanya waktu yang diperlukan *service heartbeat* untuk memastikan koneksi dari anggota klusternya. Hasil rata-rata *respond time* yang diperoleh dari keseluruhan nilai *respond time* adalah 0,09 detik.

Berdasarkan hasil pengujian selama 24 jam dengan memberikan gangguan sesuai skenario diatas, maka

didapatkan nilai *availability* pada tabel 4.4 melalui perhitungan persamaan (7) berikut.

TABEL VIII
HASIL PERHITUNGAN *AVAILABILITY*

Skenario	**Rata-rata *uptime* (*second*)**	**Rata-rata *Down*time (*second*)**	***Availbility* (%)**
1	86398,566	1,434	99,99
2	86398,444	1,556	99,99
3	86368,900	31,1	99,96
4	86206,820	193,18	99,77
5	86291,700	108,3	99,88
6	86306,580	93,42	99,89
7	86250,100	149,9	99,83
Rata-rata	**86317,300**	**82,7**	**99,90**

Tabel 8 menunjukkan nilai *availability* yang didapatkan dengan memberikan gangguan sesuai skenario pengujian. Pada skenario 1 dan skenario 2 menunjukkan hasil yang sama dengan nilai *availability* terbesar yaitu 99,99%. Nilai *availability* terkecil didapatkan pada pengujian skenario 4 yaitu sebesar 99,77%. Hasil rata-rata keseluruhan didapatkan nilai *availability* yang tinggi yaitu sebesar 99,90%.

2) Pengujian Workload: Pengujian *workload* dilakukan untuk melihat jumlah beban kerja yang dapat ditangani oleh suatu *server*. Untuk pengujian *workload* dilakukan dengan menggunakan *benchmarking tool* yaitu aplikasi siege. Pengujian dilakukan dengan memberikan beban akses dimulai dari 200 *request* secara bersamaan hingga *server* tidak mampu menangani semua *request* tersebut. Selama itu dapat dilihat nilai *workload* dari *server* yang diuji melalui *tool* siege. Pengujian dilakukan pada kedua *web server* sehingga akan terlihat perbedaannya.

Pengujian dilakukan dengan membandingkan jumlah *request* yang dapat ditangani *web server* dengan *failover* maupun *web server* tanpa *failover* untuk melihat perbandingan diantara kedua *web server* tersebut. Hasil pengujian yang diperoleh dari pengujian *tool* siege ditampilkan pada Tabel 9 berikut.

TABEL IX
HASIL PENGUJIAN *WORKLOAD*

Jumlah *Request*	**Dengan *Failover***		**Tanpa *Failover***	
	***Request* Berhasil**	***Request* Gagal**	***Request* Berhasil**	***Request* Gagal**
200	200	0	200	0
400	400	0	400	0
600	600	0	600	0
800	800	0	800	0
1000	823	177	931	69

Tabel 9 menunjukan hasil pengamatan terhadap *workload* pada *web server* kluster dan *web server* tunggal dengan memberikan *request* secara bersamaan sejumlah 200 *request* hingga *web server* tidak mampu menangai jumlah *request* yang masuk. Pengujian pada kedua *web server* berhenti pada pemberian *request* sebanyak 1000 *request* karena terdapat *request* yang gagal. Kemudian dilakukan pengujian ulang untuk mendapatkan hasil pasti dari *workload* hingga *web server* tidak mampu lagi menanganinya. Hasil pengujian ulang dengan tersebut dapat dilihat pada Tabel 10 berikut.

TABEL X
HASIL PENGUJIAN ULANG

Jumlah *Request*	**Dengan *Failover***		**Tanpa *Failover***	
	***Request* Berhasil**	***Request* Gagal**	***Request* Berhasil**	***Request* Gagal**
900	844	56	844	56
850	848	2	842	8
840	833	7	813	27
830	810	20	816	14
820	809	11	805	15
810	803	7	808	2
805	805	0	805	0
Jumlah *Request*	**Dengan *Failover***		**Tanpa *Failover***	
	***Request* Berhasil**	***Request* Gagal**	***Request* Berhasil**	***Request* Gagal**
806	806	0	806	0
807	803	4	807	0
808	-	-	808	0
809	-	-	798	11

Tabel 10 menunjukkan hasil pengujian ulang untuk menentukan hasil pasti *workload* dari *web server*. Pengujian pada *web server* dengan *failover* berhenti pada pemberian 807 *request* dikarenakan terdapat *request* gagal, sedangkan pada *web server* tanpa *failover* berhenti pada pemberian 809 *request* dikarenakan terdapat *request* gagal. Berdasarkan pengujian ulang didapatkan hasil pada *web server* dengan *failover* hanya mampu menangani 806 request tanpa kegagalan sedangkan pada *web server* tanpa *failover* masih bisa menangani *request* hingga 808 *request* tanpa kegagalan. Jumlah tersebut menunjukkan batasan dari masing-masing *workload* dari *web server* dimana *web server* tanpa *failover* memiliki jumlah yang lebih tinggi dibandingkan dengan *web server* dengan *failover*.

3) Pengujian Quality of service: Pengujian ini dilakukan dengan memberikan beban akses kepada *web server* dengan jumlah tertentu. Data yang didapatkan selama pengujian akan dilakukan perhitungan untuk mengetahui performa dari *web server*. Adapun pengujian dilakukan untuk mengetahui perbedaan antara *web server* dengan *failover* dengan *web server* tanpa *failover*. Perhitungan tersebut dilakukan terhadap parameter-parameter QoS yaitu *throughput* , *packet loss*, *delay*, dan *jitter*. Hasil pengujian tersebut akan menentukan bagaimana nilai dari QoS suatu *web server*. Standar yang digunakan untuk menentukan nilai QoS adalah *Telecommunication and Internet Protocol Harmonization Over Networks* (TIPHON) dengan pengujian terhadap parameter yang telah disebutkan.

Adapun pengujian yang dilakukan adalah sebagai berikut.

- Pengujian *Throughput*
  Pengujian parameter *throughput* menggunakan *tool* iperf dengan memberikan beban akses berupa sejumlah paket data yang jumlahnya semakin

meningkat. Pengujian dilakukan dengan mengirimkan paket data sebesar 1 MB, 2 MB, 3 MB, 4 MB, dan 5 MB sehingga akan diperoleh waktu pengamatan serta bandwidth dari *web server* yang diujikan melalui hasil capture yang dilakukan *tool* iperf. Selanjutnya hasil pengujian tersebut akan dihitung menggunakan persamaan (1) agar didapatkan nilai *throughput* dari *web server*. Kemudian dengan menggunakan persamaan (2) maka akan didapatkan tingkat *throughput* dalam bentuk persen (%). Pengujian dilakukan dengan mengamati kedua sisi *web server* sehingga melalui pengujian dengan *tool* iperf didapat data pengamatan seperti yang ditunjukkan pada Tabel 11 berikut.

TABEL XI
DATA PENGAMATAN PARAMETER *THROUGHPUT*

Paket Dikirim (MB)	**Dengan *Failover***		**Tanpa *Failover***	
	Lama Pengamatan (*second*)	***Bandwidth (Mbps)***	**Lama Pengamatan (*second*)**	***Bandwidth (Mbps)***
1	1,8	4,69	0,4	21
2	0,9	19,1	2,1	8,11
3	1,4	18,5	2,2	11,5
4	1,7	19,7	3,1	11
5	2,2	19,5	9,2	4,54

Data pada Tabel 11 adalah hasil pengamatan dari beban akses yang diberikan pada *web server* selama pengujian menggunakan *tool* siege yang dapat dilihat pada lampiran. Paket diterima merupakan jumlah Byte dari paket data yang diterima selama pengujian, dan lama pengamatan merupakan jumlah waktu pengiriman paket data pertama hingga paket data terakhir. Berdasarkan data yang diperoleh melalui pengujian maka dengan persamaan (1) dan persamaan (2) didapatkan nilai *throughput* sebagai berikut.

TABEL XII
HASIL PERHITUNGAN PARAMETER *THROUGHPUT*

Paket Dikirim (MB)	**Dengan *Failover***		**Tanpa *Failover***	
	Throughput (%)	**Kategori**	***Throughput (%)***	**Kategori**
1	97,04	Bagus	97,52	Bagus
2	95,31	Bagus	96,2	Bagus
3	94,89	Bagus	97,14	Bagus
4	97,84	Bagus	96,09	Bagus
5	95,48	Bagus	98,07	Bagus
Rata-rata	**96,11**	**Bagus**	**97**	**Bagus**

Tabel 12 merupakan hasil perhitungan terhadap parameter *throughput* pada *web server* kluster dan *web server* tunggal. Melalui nilai rata-rata *throughput* secara keseluruhan, didapatkan nilai tertinggi pada *web server* tanpa *failover* yaitu 96,11% dengan kategori bagus dan nilai terendah pada *web server* dengan *failover* yaitu 97% dengan kategori bagus. Berdasarkan pengujian *throughput*, tidak terdapat perbedaan nilai yang terlalu besar antara *web server* dengan *failover* dan *web server* tanpa *failover* sehingga kedua *web server* berada pada kategori bagus.

- Pengujian *Packet loss*
  Pengujian ini dilakukan dengan memberikan beban akses kepada *web server* dengan jumlah tertentu yang semakin meningkat dengan menggunakan *tool* siege. Beban akses yang diberikan pada pengujian ini yaitu sebesar 200 hingga 1000 *request* sehingga dapat diperoleh total paket yang diterima melalui hasil capture yang dilakukan *tool* siege. Kemudian dari hasil pengamatan, dilakukan perhitungan dengan persamaan (3) untuk mendapatkan nilai dari *packet loss*. Pengujian dilakukan dengan mengamati kedua sisi *web server* sehingga melalui data pengujian *tool* siege didapat data pengamatan seperti yang ditunjukkan pada sebagai berikut.

TABEL XIII
DATA PENGAMATAN PARAMETER *PACKET LOSS*

Jumlah *Request*	**Dengan *Failover***	
	Dengan *Failover*	**Tanpa *Failover***
200	200	200
400	400	400
600	600	600
800	800	800
1000	823	931

Tabel 13 adalah hasil pengamatan dari *web server* yang diberikan beban akses selama pengujian. Jumlah *request* merupakan total beban akses yang diberikan kepada *web server* dan paket diterima merupakan jumlah *request* yang dapat ditangani oleh *web server*. Berdasarkan data yang diperoleh melalui pengujian maka nilai *packet loss* akan diperoleh melalui perhitungan dengan persamaan (3) sebagai berikut.

TABEL XIV
HASIL PERHITUNGAN PARAMETER *PACKET LOSS*

Jumlah *Request*	**Dengan *Failover***		**Tanpa *Failover***	
	Packet loss (%)	**Kategori**	***Packet loss (%)***	**Kategori**
200	97,04	Bagus	97,52	Bagus
400	95,31	Bagus	96,2	Bagus
600	94,89	Bagus	97,14	Bagus
800	97,84	Bagus	96,09	Bagus
1000	95,48	Bagus	98,07	Bagus
Rata-rata	**96,11**	**Bagus**	**97**	**Bagus**

Tabel 14 merupakan hasil perhitungan terhadap parameter *packet loss* pada kedua jenis *web server*. Dari perhitungan tersebut maka didapatkan nilai *packet loss* terendah pada *web server* tanpa *failover* yaitu sebesar 1,38% dengan kategori bagus dan tertinggi pada *web server* dengan *failover* yaitu sebesar 3,54% dengan kategori sedang. Berdasarkan pengujian *packet loss*, terdapat perbedaan antara *web server* dengan *failover* dengan *web server* tanpa *failover*.

- Pengujian *Delay (Latency)*
  Pengujian *delay* dilakukan dengan mengamati jumlah paket yang diterima serta waktu yang

diperlukan dalam pengujian. Pengujian ini dilakukan dengan cara mengirimkan beban akses kepada *web server* dengan jumlah *request* tertentu yang dimulai dari 200 sampai 1000 *request* menggunakan *tool* siege. Kemudian dengan menggunakan *tool* siege yang akan mengcapture data pengujian berupa jumlah paket yang berhasil diterima beserta waktu pengujiannya yang dapat dilihat pada lampiran. Data tersebut akan dihitung untuk mendapatkan *delay* rata-rata menggunakan persamaan (4). Adapun pengujian dilakukan dengan mengamati kedua sisi *web server* untuk melihat perbedaannya sehingga didapat data pengamatan melalui pengujian *tool* siege seperti yang ditunjukkan sebagai berikut.

TABEL XV
DATA PENGAMATAN PARAMETER *DELAY*

Jumlah *Request*	**Dengan *Failover***		**Tanpa *Failover***	
	Lama Pengamatan (*second*)	**Paket Diterima**	**Lama Pengamatan (*second*)**	**Paket Diterima**
200	4,58	200	5,49	200
400	8,91	400	8,42	400
600	22,34	600	21,87	600
800	39,14	800	32,74	800
1000	32,48	823	35,67	931

Tabel 15 adalah hasil pengamatan yang didapatkan dengan pengujian menggunakan *tool* siege dimana web *server* diberikan beban akses secara bersamaan sejumlah tertentu selama pengujian. Jumlah *request* merupakan total beban akses yang diberikan kepada *web server* dan paket diterima merupakan jumlah *request* yang dapat ditangani oleh *web server*. Selain itu juga terdapat lama pengamatan yang merupakan waktu yang diperlukan dalam menangani *request* dari *client*. Berdasarkan data yang diperoleh melalui pengujian tersebut maka nilai *delay* rata-rata akan diperoleh melalui perhitungan dengan persamaan (4) sebagai berikut.

TABEL XVI
HASIL PERHITUNGAN PARAMETER *DELAY*

Jumlah *Request*	**Dengan *Failover***		**Tanpa *Failover***	
	Delay (ms)	**Kategori**	***Delay (ms)***	**Kategori**
200	22,9	Sangat Bagus	27,45	Sangat Bagus
400	22,28	Sangat Bagus	21,05	Sangat Bagus
600	37,23	Sangat Bagus	36,45	Sangat Bagus
800	48,93	Sangat Bagus	40,93	Sangat Bagus
1000	39,47	Sangat Bagus	38,31	Sangat Bagus
Rata-rata	**34,16**	**Sangat Bagus**	**32,84**	**Sangat Bagus**

Tabel 16 merupakan hasil perhitungan terhadap parameter *delay* pada kedua jenis *web server*. Hasil perhitungan menunjukkan nilai *delay* paling rendah terdapat pada *web server* tanpa *failover* yaitu sebesar 32,84ms dan nilai *delay* paling tinggi terdapat pada *web server* dengan *failover* yaitu sebesar 34,16ms dengan kategori sangat bagus. Berdasarkan pengujian *delay*, terdapat perbedaan nilai diantara *web server* dengan *failover* dan *web server* tanpa *failover* meskipun perbedaan yang didapat tidak terlalu besar sehingga kedua jenis *web server* sama-sama berada pada kategori sangat bagus.

- Pengujian *Jitter*
  Pengujian *jitter* dilakukan dilakukan dengan mengamati jumlah paket yang diterima beserta waktu pengamatannya selama pengujian dilakukan. Dikarenakan *jitter* memilliki hubungan erat dengan latency maka data *delay* rata-rata juga diperlukan untuk mengdapatkan nilai *jitter*. Pengujian dilakukan dengan memberikan beban akses kepada *web server* dengan jumlah tertentu yang dimulai dari 200 hingga 1000 *request* menggunakan *tool* siege. Data pengujian dicapture oleh *tool* siege yang kemudian akan dilakukan perhitungan dengan persamaan (2.5) untuk mendapatkan nilai *jitter* pada *web server*. Pengujian dilakukan dengan mengamati kedua sisi *web server* sehingga didapat data pengamatan melalui pengujian *tool* siege sebagai berikut.

TABEL XVII
DATA PENGAMATAN PARAMETER *JITTER*

Jumlah *Request*	**Dengan *Failover***		**Tanpa *Failover***	
	Lama Pengamatan (*second*)	**Paket Diterima**	**Lama Pengamatan (*second*)**	**Paket Diterima**
200	4,58	200	5,49	200
400	8,91	400	8,42	400
600	22,34	600	21,87	600
800	39,14	800	32,74	800
1000	32,48	823	35,67	931

Tabel 17 adalah data hasil pengamatan yang didapatkan pada saat pengujian menggunakan *tool* siege dengan memberikan beban akses secara bersamaan dengan jumlah yang semakin meningkat selama pengujian. Jumlah *request* merupakan total beban akses yang diberikan kepada *web server* dan paket diterima merupakan jumlah *request* yang dapat ditangani oleh *web server*. Selain itu juga terdapat lama pengamatan yang merupakan waktu yang diperlukan dalam menangani *request* dari *client*. Untuk mendapatkan nilai *jitter* diperlukan total variasi *delay* yang didapatkan dengan menggunakan persamaan (6) terlebih dahulu. Kemudian berdasarkan data yang telah diperoleh maka nilai *jitter* dapat dihitung menggunakan persamaan (5) sebagai berikut.

Tabel 18 merupakan hasil perhitungan terhadap parameter *jitter* pada kedua *web server*. Hasil perhitungan menunjukkan nilai *jitter* paling rendah terdapat pada *web server* tanpa *failover* yaitu sebesar 32,77ms dengan kategori bagus dan nilai *jitter* paling tinggi terdapat pada *web server* dengan *failover* yaitu sebesar 34,09 dengan kategori bagus. Berdasarkan pengujian *jitter*, terdapat

perbedaan nilai diantara *web server* dengan *failover* dan *web server* tanpa *failover* meskipun perbedaan yang didapat tidak terlalu besar sehingga kedua jenis *web server* sama-sama berada pada kategori bagus.

TABEL XVIII
HASIL PERHITUNGAN PARAMETER *JITTER*

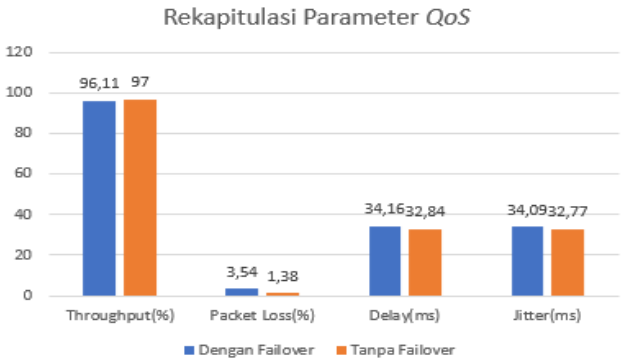
Jumlah *Request*	Dengan *Failover*		Tanpa *Failover*	
	Delay (ms)	Kategori	*Delay (ms)*	Kategori
200	22,79	Sangat Bagus	27,31	Sangat Bagus
400	22,22	Sangat Bagus	21	Sangat Bagus
600	37,27	Sangat Bagus	36,39	Sangat Bagus
800	48,86	Sangat Bagus	40,87	Sangat Bagus
1000	39,42	Sangat Bagus	38,27	Sangat Bagus
Rata-rata	**34,09**	**Sangat Bagus**	**32,77**	**Sangat Bagus**

Hasil dari perhitungan masing-masing parameter selanjutnya dikelompokkan dan ditentukan nilai indeks berdasarkan standar TIPHON (*Telecommunication and internet protocol harmonization over network*). Berikut merupakan hasil rekapitulasi pengujian parameter QoS secara keseluruhan.

TABEL XIX
HASIL PERHITUNGAN PARAMETER *DELAY*

Parameter QoS	Dengan *Failover*			Tanpa *Failover*		
	Rata-rata	Kategori	Nilai Indeks	Rata-rata	Kategori	Nilai Indeks
Throughput(%)	96,11	Bagus	3	97	Bagus	3
Packet loss(%)	3,54	Sedang	2	1,38	Bagus	3
Delay(ms)	34,16	Sangat Bagus	4	32,84	Sangat Bagus	4
Jitter (ms)	34,09	Bagus	3	32,77	Bagus	3
Rata-rata	**34,16**	**Memuaskan**	**3**		**Memuaskan**	**3,25**

Tabel 19 menunjukkan tingkatan QoS yang didapat selama pengujian terhadap parameter-parameter QoS. Hasil yang didapatkan terhadap pengujian QoS menunjukkan perbedaan antara *web server* dengan *failover* dan *web server* tanpa *failover* secara keseluruhan. Tingkatan QoS yang didapatkan pada pengujian menunjukkan kedua jenis *web server* berada pada tingkat memuaskan namun dengan nilai indeks yang berbeda. Nilai indeks paling tinggi terdapat pada *web server* tanpa *failover* yaitu sebesar 3,25 dan nilai indeks paling rendah terdapat pada *web server* dengan *failover* yaitu sebesar 3. Perbedaan tersebut jika ditampilkan ke dalam grafik adalah sebagai berikut.



Gambar. 6 Hasil rekapitulasi parameter QoS

Gambar 6 menunjukkan perbedaan berdasarkan pengujian keseluruhan parameter QoS yang menunjukkan adanya sedikit pengaruh penggunaan *failover* terhadap QoS dari *web server*. Perbedaan tersebut dapat dilihat melalui pengujian tiap parameter QoS. Hasil pengujian parameter-parameter QoS menunjukkan bahwa *web server* tanpa *failover* sedikit lebih baik daripada *web server* dengan *failover*. Perbedaan tersebut terjadi dikarenakan penggunaan *resource* yang sedikit lebih banyak pada *web server* dengan *failover* untuk menjalankan sistem *failover* tersebut sehingga mempengaruhi performa dari *web server*. Berbeda dengan *web server* tanpa *failover* dimana *web server* tersebut tidak perlu menjalankan sistem *failover* sehingga *resource* yang dapat digunakan sedikit lebih banyak.

B. Analisis Hasil Pengujian

Rincian hasil analisis pengujian *web server* yang telah dilakukan adalah sebagai berikut.

- Berdasarkan hasil pengujian *availability* didapatkan hasil bahwa meskipun terjadi *down* pada *primary server*, *client* masih bisa mengakses layanan dari *web server* yang membuktikan bahwa layanan *high availability web server* tersebut berjalan dengan baik meskipun terdapat *down*time selama beberapa detik dikarenakan adanya *backup server* yang mengambil alih layanan disaat terjadi *down* pada *primary server*. Berbeda saat *web server* tidak menggunakan *failover*, layanan dari *web server* langsung terputus ketika terjadi *down* sehingga *client* tidak bisa mengakses *web server* tersebut. *Down*time yang terjadi pada pengujian *web server* dengan *failover* tersebut merupakan waktu perpindahan *web server* saat terjadi *down*. Hasil pengujian *down*time tersebut berdasarkan skenario pengujian yang telah ditetapkan menunjukkan bahwa perpindahan dari *primary server* ke *secondary server* lebih cepat 0,122 detik dibanding dengan perpindahan sebaliknya. *Respond time* tercepat diperoleh pada skenario 4 yaitu saat hanya *primary server* yang dihidupkan. Selain itu, waktu yang dibutuhkan *web server* untuk *up* kembali dari keadaan *down* juga berbeda saat kedua *web server* dihidupkan secara bersamaan, *web server* dihidupkan satu persatu, maupun hanya satu *web server* yang dihidupkan. Waktu pengembalian sistem tercepat saat kedua *web server* dihidupkan secara bersamaan yaitu sebesar 31,1

detik dengan pengujian skenario 3 yang dikarenakan sistem *failover* menunggu koneksi *web server* lain yang termasuk anggota klusternya saat pertama kali dijalankan selama beberapa saat sebelum sistem *failover* memutuskan untuk memulai tugasnya. Berdasarkan hasil pengujian melalui pemberian gangguan sesuai skenario yang ditetapkan dengan waktu pengujian yang diambil selama 24 jam menunjukkan rata-rata persentase tingkat *availability* sebesar 99,90% yang menunjukkan tingginya tingkat *availability* dari *web server* tersebut.

- Berdasarkan hasil pengujian *workload* menunjukkan jumlah rata-rata *request* yang dapat ditangani oleh *web server*. Jumlah *request* terkecil dimiliki *web server* dengan *failover* yaitu sebesar 806 *request* dan jumlah *request* terbesar dimiliki *web server* tanpa *failover* yaitu sebesar 808 *request*. Pengujian ini menunjukkan adanya perbedaan antara kedua jenis *web server* meskipun tidak terlalu besar pada jumlah *workload* yang dapat ditangani masing-masing *web server*. Perbedaan tersebut terjadi karena jumlah penggunaan *resource* pada *web server*. *Web server* dengan *failover* menggunakan jumlah *resource* yang sedikit lebih besar untuk menjalankan sistem *failover* sehingga jumlah tersebut ikut mempengaruhi kinerja dari *web server*.
- Berdasarkan hasil pengujian QoS menunjukkan perbedaan antara *web server* dengan *failover* dan *web server* tanpa *failover* secara keseluruhan. Tingkatan QoS yang didapatkan pada pengujian menunjukkan kedua jenis *web server* berada pada tingkat memuaskan namun dengan nilai indeks yang berbeda. Nilai indeks paling tinggi terdapat pada *web server* tanpa *failover* yaitu sebesar 3,25 dan nilai indeks paling rendah terdapat pada *web server* dengan *failover* yaitu sebesar 3. Perbedaan tersebut dapat dilihat melalui pengujian tiap parameter QoS. Hasil pengujian parameter-parameter QoS menunjukkan bahwa *web server* tanpa *failover* sedikit lebih baik daripada *web server* dengan *failover*. Perbedaan tersebut terjadi dikarenakan penggunaan *resource* yang sedikit lebih banyak pada *web server* dengan *failover* untuk menjalankan sistem *failover* tersebut sehingga mempengaruhi performa dari *web server*. Berbeda dengan *web server* tanpa *failover* dimana *web server* tersebut tidak perlu menjalankan sistem *failover* sehingga *resource* yang dapat digunakan sedikit lebih banyak.

Berdasarkan pengujian yang telah dilakukan menunjukkan bahwa sistem *failover* telah bekerja dengan baik sesuai dengan konsepnya. Dapat dilihat saat *web server* dengan *failover* diberikan gangguan yang menyebabkan *web server down* masih bisa diakses oleh client sehingga menunjukkan tercapainya *high availability web server*. Terdapat waktu beberapa saat *web server* tersebut tidak dapat memberikan layanan dikarenakan adanya waktu yang diperlukan *web server* untuk memindahkan layanannya yang disebut waktu *downtime*. Hasil pengujian menunjukkan bahwa semakin cepat *downtime* yang terjadi maka semakin besar persentase tingkat *availability*. Berdasarkan pengujian *workload* dan QoS menunjukkan bahwa penggunaan sistem *failover* memberikan pengaruh pada performa perangkat meskipun pengaruh tersebut sangat kecil. Perbedaan pengaruh tersebut dikarenakan penggunaan *service heartbeat* akan memakan beberapa *resource* pada perangkat sehingga menyebabkan berkurangnya performa dari *web server*. Hasil keseluruhan pengujian QoS menunjukkan bahwa *web server* yang dibangun layak untuk digunakan yaitu dengan tingkat memuaskan.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Setelah dilakukan analisis terhadap *web server* dengan penerapan metode *failover clustering* melalui pengujian *availability*, *workload*, dan QoS (*Quality of service*) dapat diambil kesimpulan sebagai berikut.

1. Sistem *failover clustering* yang dibangun dapat bekerja dengan baik sesuai dengan konsep kerjanya, sehingga *client* dapat mengakses layanan yang disediakan oleh *web server* meskipun terjadi kegagalan pada *web server*.
2. Hasil pengujian *availability* dapat memberikan ketersediaan layanan yang lebih baik saat terjadi kegagalan. Nilai *availability* yang didapatkan dari perhitungan data pengujian sebesar 99,90%.
3. Jumlah *workload* yang dapat dilayani oleh *web server* dengan *failover* dan *web server* tanpa *failover* secara bersamaan yang didapat selama pengujian menunjukkan bahwa dengan menggunakan *failover* menyebabkan berkurangnya *resource* sehingga terdapat sedikit perbedaan pada performa *web server* yang mempengaruhi jumlah *workload* dari *web server*. Jumlah *workload* menunjukkan perbedaan dengan jumlah 806 *request* pada *web server* dengan *failover* dan 808 *request* pada *web server* tanpa *failover* dimana perbedaanya tidak terlalu besar.
4. Hasil pengujian QoS menunjukan nilai keseluruhan pada *web server* dengan *failover* dan *web server* tanpa *failover* dengan indeks yang sama yaitu memuaskan. Namun terdapat sedikit perbedaan pada nilai pengujian tiap parameter dimana *web server* tanpa *failover* menunjukkan performa yang lebih baik. Perbedaan tersebut juga disebabkan adanya perbedaan *resource* yang digunakan pada perangkat *web server* dimana *web server* dengan *failover* lebih banyak menggunakan *resource* untuk menjalankan sistem *failover*.
5. Konfigurasi *failover clustering* menunjukkan hasil yang baik dilihat dari segi *availability* pada saat terjadi kegagalan.

B. Saran

Saran yang dapat dijadikan sebagai bahan pertimbangan dan masukan untuk pengembangan selanjutnya adalah sebagai berikut.

1. Beberapa faktor seperti jumlah *request*, hardware, dan bandwidth dapat mempengaruhi hasil pengujian sehingga perlu diperhatikan saat melakukan pengujian.

2. Dilakukan pengembangan dengan menggunakan spesifikasi hardware yang lebih *high* end untuk menambah kemampuan kerja sistem.
3. Dilakukan pengembangan dengan menambahkan beberapa kluster load balancing untuk menambah ketersediaan terhadap layanan.

DAFTAR PUSTAKA

[1] Apache. (n.d.). The Apache Software Foundation. November, 2015. https://www.apache.org/

[2] L. M. Iswan, t.thn. Implementasi Virtual Private Network (VPN) Remote Access Dengan Linux OpenSwan. Jakarta.

[3] Sampurna. 1996. Belajar Sendiri Membuat Homepage Dengan HTML. Jakarta: PT. Elex Media Komputindo.

[4] A. Sahala, 2014. Implementasi Jaringan dengan Linux Ubuntu. Yogyakarta: Andi Yogyakarta.

[5] P. T. Pribadi, 2013. Implementasi High-Availability VPN Client Pada Jaringan Komputer Fakultas Hukum Universitas Udayana. Jurnal Ilmu Komputer.

[6] E. Krout, 2018, June 1. Load Testing Web Servers with Siege. February 13, 2019. https://www.linode.com/docs/tools-reference/tools/load-testing-with-siege/

[7] A. Muchtar, t.thn. Implementasi Failover Clustering Pada Dua Platform Yang Berbeda Untuk Mengatasi Kegagalan Fungsi Server. Makassar: Universitas Negeri Makassar.

[8] Irfani. 2015. Implementasi High Availability Server dengan Teknik Failover Virtual Computer Cluster. Surakarta: Universitas Muhammadiyah Surakarta.

[9] P. A. Nugraha, 2016. Rancang Bangun Web server Berbasis Linux Dengan Metode Load Balancing. Pontianak: Program Studi Informatika Universitas Tanjungpura.

[10] Kahanwal, & B, S. 2012. The Distributed Computing Paradigma: P2P, Grid, Cluster, Cloud, and Jungle. Jurnal Internasional Penelitian Sains dan Teknologi.

[11] K. Kaur & A. K. Rai, 2014. A Comparative Analysis: Grid, Cluster and Cloud Computing. International Journal of Advanced Research in Computer and Communication Engineering.

[12] G. P. Rao, , E. R. Brueggemann & R. A.RodriguezMethod For Maintaining Transaction Integrity Across Multiple Remote Access Servers. US 11/626.334, 2010.

[13] C.Bookman, 2003. Linux Clustering: Building and Maintaining Linux Clusters. USA: New Riders Publishing.

[14] J.Purnomo & S. W.Edi Implementasi dan Analisis High Availability Server dengan Teknik Failover Clustering Menggunakan Heartbeat. Salatiga: Universitas Kristen Satya Wacana, 2017.

[15] J. I. Mumpuni & A. Wardono, Meningkatkan Kemampuan Jaringan Komputer dengan PC Cloning System Menjadikan PC x86 berkemampuan Pc iP4. Yogyakarta: Andi Offset.

[16] Yanto, Analisis QoS (Quality of service) Pada Jaringan Internet (Studi Kasus Fakultas Teknik Untan). Pontianak, 2006, 2013.

[17] D. Asterina, Implementasi Dan Analisis Metode Failover Pada Sistem Redudant Dedicated Server Dan Cloud Server Untuk Layanan VOIP. Bandung: Fakultas Teknik Elektro, 2015.

[18] Tiphon, Telecommunication And Internet Protocol Harmonization Over Network (TIPHON) General Aspects Of Quality of service (QoS). TR 101 329 V2.1.1. 1999.

[19] P. Syafrizal, Failover Cluster Server dan Tunneling EOIP untuk Sistem Disaster Recovery, Seminar Nasional Teknologi Informasi dan Multimedia. Yogyakarta: STMIK AMIKOM Yogyakarta, 2013.

[20] Linode. 2018, June 1. Network Throughput Testing with iPerf. January 25, 2019. https://www.linode.com/docs/networking/diagnostics/install-iperf-to-diagnose-network-speed-in-linux/