

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

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 (Andreolini, 2004).

*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 *Hypertext Transfer Protocol* (HTTP) untuk mengirimkan data. 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*. *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. 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 komputer. Selain itu, *cluster* juga merupakan sekelompok mesin yang bertindak sebagai sebuah entitas tunggal untuk menyediakan sumber daya dan layanan ke jaringan (Kaur, 2014). 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 lain dalam kondisi yang terkendali (Dharma, 2008). 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.

## 1.2 Perumusan Masalah

Berdasarkan pemaparan pada latar belakang maka permasalahan yang dapat diambil adalah bagaimana merancang *web server* dengan metode *failover clustering* yang dapat menyediakan layanan *high availability* sehingga dapat mengatasi *failure* pada *server*. Sistem *server* dengan teknologi *high availability* sangat diperlukan untuk membangun *web server* yang bersifat *real-time*. Teknologi tersebut dapat dibangun dengan penggunaan metode *failover clustering* pada *clustering web server*. Pembangunan *web server clustering* dilakukan dengan menggunakan lebih dari satu *server*. *Server* pertama akan bekerja sebagai *server* utama yang akan melayani semua *request* dari *client* sedangkan *server* lainnya berstatus *stand-by* atau sebagai cadangan apabila terjadi kegagalan pada *server* utama. Setiap *server* harus terinstal paket-paket yang dibutuhkan untuk membangun *web server* salah satunya adalah paket *apache2*. Konfigurasi *ip address* juga harus dilakukan agar tiap *web server* dapat saling terhubung. Kemudian agar *high availability* tercapai maka kedua *web server* harus dipasang paket *heartbeat* dimana paket tersebut diperlukan sebagai sistem *failover* dalam *clustering server*.

### 1.3 Tujuan Penelitian

Tujuan dari penelitian adalah melihat pengaruh penggunaan metode *failover clustering* dalam mencapai layanan *high availability* pada *web server* gedung jurusan Informatika.

### 1.4 Pembatasan Masalah

Pembatasan masalah dari penelitian yang akan dilakukan adalah.

1. Menggunakan satu unit komputer *server* sebagai *master server* (server utama) dan satu unit PC sebagai *slave server* (server cadangan).
2. Menggunakan sistem operasi Linux Ubuntu pada *web server*.
3. Analisis hanya dilakukan terhadap penggunaan metode *failover clustering* dengan tipe *active/pasif* pada *web server cluster* di Gedung Jurusan Informatika Universitas Tanjungpura.

### 1.5 Sistematika Penulisan Skripsi

Sistematika penulisan tugas akhir ini disusun dalam lima bab yang terdiri dari Bab I Pendahuluan, Bab II Tinjauan Pustaka, Bab III Metodologi Penelitian, Bab IV Hasil Perancangan dan Analisis Kinerja Sistem, serta Bab V Penutup.

Bab I: Pendahuluan adalah bab yang berisi latar belakang, rumusan masalah, tujuan penelitian, pembatasan masalah dan sistematika penulisan. Pada bab ini membahas sekilas tentang penelitian yang diambil yaitu sistem *web server* dengan menggunakan teknologi *server clustering* yang mengimplementasikan metode *failover clustering*.

Bab II: Tinjauan Pustaka adalah bab yang berisi landasan teori yang berkaitan dengan apa saja yang dibutuhkan dalam membangun *webserver clustering* dan pengujian yang tepat pada penelitian ini. Selain itu juga berisi uraian tentang hasil penelitian terkait yang telah dilakukan oleh peneliti lain.

Bab III: Metodologi Penelitian adalah bab yang berisi tentang Metodologi Penelitian, alat yang dipergunakan dalam membangun *webserver*, analisis hasil pengujian pada parameter yang diterapkan metode *failover* serta diagram alir penelitian.

Bab IV: Hasil Analisis dan Perancangan Sistem adalah bab yang berisi data perancangan, hasil percobaan, pengamatan, pengujian, dan berbagai hal mengenai pengimplementasian metode *failover* pada *webserver clustering* yang

telah dirancang pada Bab III. Setiap hasil yang disajikan akan dilakukan analisis untuk mengarah kepada suatu kesimpulan.

Bab V: Penutup berisi kesimpulan dari tugas akhir dan saran untuk pengembangan penelitian yang lebih lanjut.

## BAB II

### TINJAUAN PUSTAKA

#### II.1 Penelitian Terkait

Akhyar Muchtar melakukan penelitian Implementasi *Failover Clustering* Pada Dua Platform Yang Berbeda Untuk Mengatasi Kegagalan Fungsi Server. Penelitian ini bertujuan untuk membuat rancangan *failover* clustering sebagai salah satu solusi dalam menangani masalah kegagalan fungsi server dengan membandingkan *performance* sistem *failover* pada sistem operasi Linux dan Windows. Penelitian ini menggunakan metode *experimental* dengan melakukan pengujian langsung pada masing-masing teknologi berdasarkan parameter tertentu dengan kondisi yang terkendali. Parameter pengujian tersebut adalah *availability*, *workload*, *latency*, dan *packet loss*. Hasil pengujian menunjukkan bahwa penerapan *failover clustering* pada Proxmox menggunakan UCARP dan DNS *failover* sama baiknya dari segi kinerja dalam mencapai nilai standar *availability*, namun DNS *failover* sedikit lebih baik karena dapat digunakan pada *platform* Linux ataupun Windows. Nilai *availability* yang diperoleh menggunakan UCARP dan DNS *failover* memperoleh tingkat *availability* yang sama yakni sebesar 99,99%.

Prayudi Aditya Nugraha (2016) melakukan penelitian Rancang Bangun Web Server Berbasis Linux Dengan Metode *Load Balancing* Pada Gedung Jurusan Informatika Universitas Tanjungpura. Penelitian ini bertujuan untuk membangun *web server* dengan menggunakan *load balancing* pada *server cluster*. Pada *load balancer* yang dibangun menerapkan algoritma *round robin* dan *least connection*. Pengujian dilakukan dengan melakukan analisis pada ketersediaan layanan, waktu respon, dan *throughput*. Hasil pengujian menunjukkan bahwa *web server* dengan *load balancing* dapat memberikan ketersediaan yang lebih baik dibandingkan *web server* tunggal. Pengujian *throughput web server load balancing* dengan algoritma *round robin* memiliki nilai yang paling baik yaitu 0,19 MB/detik sedangkan dengan algoritma *least*

*connection* memiliki *throughput* paling kecil yaitu 0,174 MB/detik. Pengujian waktu respon *web server load balancing* dengan algoritma *least connection* memiliki waktu respon tercepat yaitu 0,258 detik, sedangkan *web server* tunggal memiliki waktu respon terlama yaitu 0,284 detik.

Putu Topan Pribadi (2013) melakukan penelitian Implementasi *High-Availability* VPN Client Pada Jaringan Komputer Fakultas Hukum Universitas Udayana. Penelitian ini bertujuan untuk memastikan koneksi VPN tetap hidup dengan mengimplementasikan teknologi yang bersifat *high availability* agar administrator dapat lebih mudah *me-monitoring* jaringan dari luar. Pengujian dilakukan dengan mematikan *active node* sehingga dianggap *server* mengalami kegagalan dan fungsi akan dialihkan ke *passive node* sebagai *backup server*. Hasil pengujian menunjukkan bahwa dengan adanya sistem *High Availability VPN Client*, layanan tidak akan terganggu yang diakibatkan oleh kerusakan *primary server* karena *secondary server* akan mengambil alih tugas *primary server* dengan baik saat terjadi kegagalan.

Irfani Hernawan Sulistyanto (2015) melakukan penelitian Implementasi High Availability Server dengan Teknik *Failover* Virtual Computer Cluster. Penelitian ini bertujuan untuk merancang sistem *failover virtual computer cluster* sebagai salah satu solusi untuk mengatasi kegagalan fungsi *server* dengan menggunakan VMware Workstation 11 sebagai *platform* simulasinya. Pengujian dilakukan dengan mengukur beberapa parameter yaitu *availability*, *downtime*, CPU *utilization*, dan *throughput*. Hasil pengujian menunjukkan bahwa nilai *availability* paling besar yang diperoleh yaitu 99,50% dengan tingkat kestabilan *cluster* dari sisi CPU *utilization* dan *throughput*, sehingga sistem *cluster virtual* ini dapat menjadi solusi untuk meningkatkan sistem dengan tingkat *availability* yang tinggi.

Joko Purnomo (2017) melakukan penelitian Implementasi dan Analisis *High Availability Server* dengan Teknik *Failover Clustering* Menggunakan Heartbeat. Penelitian ini bertujuan untuk mengetahui *downtime* dan *respon time* yang digunakan saat proses terjadinya *failover*. Hasil pengujian menunjukkan nilai *availability* sangat dipengaruhi oleh *uptime* dan *downtime*. Rata-rata *availability* dari semua pengujian adalah 99,97% sehingga perancangan dapat dikatakan memiliki *high availability* yang tinggi. Persentase *availability* paling besar yaitu 99,99% dengan waktu *downtime* 2 detik dan *respon time* 2 detik.

Penelitian yang akan dilakukan adalah membangun *web server* dengan

metode *failover clustering* yang menyediakan layanan *high availability* menggunakan heartbeat sehingga dapat dilakukan analisis terhadap pengaruh metode *failover clustering* tersebut pada *web server* gedung jurusan Informatika. Penelitian tersebut apabila dikelompokkan ke dalam tabel adalah sebagai berikut :

**Tabel 2.1** Penelitian yang Telah Dilakukan

No	Penulis	Judul	Keterangan
1	Muchtar, Akhyar (t.thn)	Implementasi <i>Failover Clustering</i> Pada Dua Platform Yang Berbeda Untuk Mengatasi Kegagalan Fungsi Server	Penelitian ini bertujuan untuk membuat rancangan <i>failover clustering</i> sebagai salah satu solusi dalam menangani masalah kegagalan fungsi server dengan membandingkan <i>performance</i> sistem <i>failover</i> pada sistem operasi Linux dan Windows
2	Nugraha, Prayudi Aditya (2016)	Rancang Bangun <i>Web Server</i> Berbasis Linux Degan Metode <i>Load Balancing</i> Pada Gedung Jurusan Informatika Universitas Tanjungpura	Penelitian ini bertujuan untuk membangun <i>web server</i> dengan menggunakan <i>load balancing</i> pada <i>server cluster</i> . Pada <i>load balancer</i> yang dibangun menerapkan algoritma <i>round robin</i> dan <i>least connection</i>
3	Pribadi, Putu Topan (2013)	Implementasi <i>High-Availability</i> VPN Client Pada Jaringan Komputer Fakultas Hukum Universitas Udayana	Penelitian ini bertujuan untuk memastikan koneksi VPN tetap hidup dengan mengimplementasikan teknologi yang bersifat <i>high availability</i> agar administrator dapat lebih mudah <i>me-monitoring</i> jaringan dari luar.
4	Sulistyanto, Irfani Hernawan (2015)	Implementasi High Availability Server dengan Teknik <i>Failover Virtual Computer Cluster</i>	Penelitian ini bertujuan untuk merancang sistem <i>failover virtual computer cluster</i> sebagai salah satu solusi untuk mengatasi kegagalan fungsi <i>server</i> dengan menggunakan VMware Workstation 11 sebagai <i>platform</i> simulasinya.

No	Penulis	Judul	Keterangan
5	Purnomo, Joko (2017)	Implementasi dan Analisis <i>High</i>	Penelitian ini bertujuan untuk mengetahui <i>downtime</i> dan <i>respon</i>



		<i>Availability Server</i> dengan Teknik <i>Failover Clustering</i> Menggunakan Heartbeat	<i>time</i> yang digunakan saat proses terjadinya <i>failover</i> .
--	--	---	---

**Tabel 2.2** Penelitian yang Dilakukan

No	Penulis	Judul	Keterangan
1	Pribadi, Yulizar	Analisis Penggunaan Metode <i>Failover Clustering</i> Untuk Mencapai <i>High Availability</i> Pada Web Server Studi Kasus Gedung Jurusan Informatika	Penelitian ini bertujuan untuk melihat pengaruh penggunaan metode <i>failover clusering</i> dalam mencapai <i>high availability</i> pada web server gedung jurusan Informatika.

## II.2 Jaringan Komputer

Jaringan komputer merupakan sebuah interkoneksi antara dua atau lebih perangkat komputer. Jaringan komputer adalah salah satu bentuk komunikasi antar komputer, sama halnya seperti yang dilakukan oleh manusia dengan manusia. Walaupun namanya jaringan komputer, namun pembuatan jaringan komputer tidak hanya melibatkan komputer saja, namun juga mampu mneggabungkan peranti lain seperti *server*, *router*, *modem*, *printer*, dan sebagainya. (Sahala, 2014).

Jaringan komputer pada umumnya termasuk dalam pokok bahasan dalam bidang telekomunikasi, ilmu komputer, teknologi informasi, dan teknik komputer. sifat dari jaringan komputer adalah memungkinkan adanya transfer data antar komputer atau perangkat yang terhubung didalamnya. Contoh jaringan yang lazim digunakan adalah LAN (*Local Area Network*), WAN (*Wide Area Network*), WLAN & WWAN (*Wireless LAN & Wireless WAN*). Sebuah jaringan komputer dihubungkan menggunakan berbagai medium, seperti kabel *twisted pair*, kabel tembaga, kabel koaksial, kabel serat optik, dan berbagai macam teknologi *wireless* (Sahala, 2014).

## II.3 Server

*Server* adalah terminal induk dimana semua kontrol terhadap jaringan terpusat. *Server* berfungsi untuk melayani dan mengatur semua komputer yang terhubung dalam jaringan, termasuk hubungan dengan perangkat (Joko, 2006).

Bentuk pelayanan yang diberikan oleh *server* meliputi:

1. *Resource sharing* yaitu berupa penggunaan perangkat tambahan bersama-sama seperti : *printer*, *scanner*, dan lain-lain.
2. *Data sharing* yaitu berupa pengolahan sebuah data atau informasi secara bersama-sama.
3. Mengatur keamanan dalam jaringan.
4. Mengatur hak akses bagi pengguna jaringan.

## II.4 Client

*Client* adalah terminal yang digunakan pengguna jaringan untuk bekerja. *Client* juga bisa digunakan pengguna untuk mengakses komputer *server* dengan batasan tertentu yang disebut hak akses. Selain mengakses komputer *server*, antar *client* juga bisa saling berkomunikasi.

Idealnya komputer yang digunakan sebagai *server* spesifikasinya haruslah lebih tinggi dari pada komputer *client*, karena mengingat tugas dan fungsinya yang sedemikian rupa. Apabila *server* harus melayani sejumlah *client* secara *non stop*, maka komputer yang digunakan sebagai *server* juga harus memiliki daya tahan yang tinggi (Joko, 2006).

## II.5 Cluster Computing

### II.5.1 Definisi Cluster Computing

Komputer kluster adalah sekumpulan komputer (umumnya server jaringan) independen yang bekerja secara bersama sebagai sumber daya komputasi tunggal yang terintegrasi dan terlihat oleh klien seolah-olah komputer tersebut adalah satu buah unit komputer (Kahanwal, 2012). Teknologi komputer kluster ini membuat klien yang menggunakan layanan tidak mengetahui ada berapa komputer yang bekerja memberikan pelayanan. Proses menghubungkan beberapa komputer agar dapat bekerja seperti itu dinamakan dengan *clustering*. Komponen kluster biasanya saling terhubung melalui sebuah interkoneksi yang sangat cepat, atau bisa juga melalui jaringan lokal (LAN) (Pribadi, 2013).

## II.5.2 Klasifikasi Cluster Computing

### 2.5.2.1 High-Availability Clusters

*High-availability Clusters*, yang juga disebut *failover cluster* pada umumnya diimplementasikan untuk tujuan meningkatkan ketersediaan layanan yang disediakan oleh kluster. Elemen kluster memiliki *node-node* redudan yang akan digunakan untuk menyediakan layanan ketika salah satu komponen mengalami kegagalan. Dibutuhkan dua buah *node* sebagai syarat minimum suatu kluster untuk dapat melakukan redudansi (Kahanwal, 2012).

Pada klasifikasi kluster ini, terdapat mode *active/passive failover* yang dapat digunakan untuk membuat *backup* jaringan. Pada mode ini terdapat dua komponen, yang satu menjadi komponen atau *node* aktif dan yang lainnya pasif. *Node* aktif bertugas untuk melakukan eksekusi terhadap aplikasi atau tugas tertentu, sedangkan *node* pasif berstatus *stand by* dengan tidak melakukan tugas apapun sampai mendeteksi bahwa terdapat masalah pada *node* utama/aktif. Pada saat *node* utama mengalami kegagalan, *node* pasif akan mengambil alih tugas tadinya dilakukan oleh *node* utama (Rao, 2010).

### 2.5.2.2 Load-Balancing Clusters

Kluster kategori ini beroperasi dengan mendistribusikan beban kerja secara merata kepada beberapa *node* yang bekerja dibelakang (*back-end node*) sehingga beban kerja disisi *server* menjadi lebih ringan. Tujuan dari *load balancing* adalah mempersingkat waktu rata-rata pengerjaan tugas pada server dan ketersediaan layanan yang tinggi (Kahanwal, 2012).

### 2.5.2.3 High-Performance Clusters

*High Performance Cluster*, cluster yang bertujuan untuk meningkatkan kemampuan komputasi dengan memanfaatkan utilitas perangkat secara maksimal (Kahanwal, 2012).

## II.6 Protokol HTTP (*Hypertext Transfer Protocol*)

Protokol adalah suatu kumpulan dari aturan-aturan yang berhubungan dengan komunikasi data antara alat-alat komunikasi supaya komunikasi data dapat

dilakukan dengan benar. Protokol biasanya berbentuk sebuah software yang mengatur komunikasi data tersebut (Iswan, 2010). Dalam *web / WWW(world wide web)* protokol HTTP merupakan protokol yang berperan penting dalam mengatur komunikasi data. HTTP juga mendefinisikan bagaimana suatu pesan bisa diformat dan dikirim dari server ke *client*.

## II.7 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. *Server* akan menanggapi koneksi tersebut dengan mengirimkan isi *file* yang diminta dan memutuskan koneksi tersebut. *Browser* kemudian mengolah informasi yang didapat dari *server*. Pada bagian *server*, *browser* yang berbeda dapat melakukan koneksi pada *server* yang sama untuk memperoleh informasi yang sama. Data ini mempunyai format yang standar, disebut dengan format SGML(*Standart General Markup Language*). Data yang berupa format ini kemudian akan ditampilkan oleh *browser* sesuai dengan kemampuan *browser* tersebut. *Web server* yang terkenal adalah *Apache*. *Web server* merupakan *software* yang menjadi tulang punggung dari *World Wide Web* (Nugraha, 2016).

## II.8 Web Browser

Menurut Sampurna (1996), *browser* adalah sebuah program yang dirancang untuk mengambil informasi-informasi dari *server* komputer pada suatu jaringan *internet* maupun *intranet*. *Web Browser* adalah suatu program yang digunakan untuk menjelajahi dunia *internet* atau untuk mencari informasi tentang suatu halaman *web* yang tersimpan di komputer. Cara kerja *web browser* adalah pada saat kita mengetikkan suatu alamat pada *browser* maka data akan dilewatkan oleh suatu protokol HTTP melewati port 80 pada *server*.

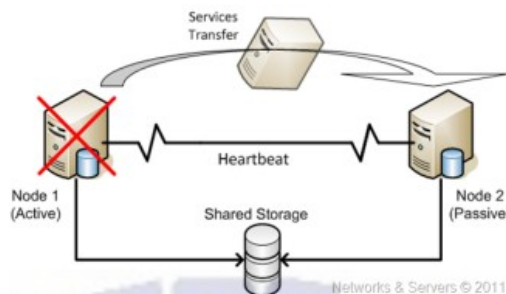
## II.9 Apache Web Server

*Apache* merupakan *web server* unggul daripada banyak *server web* berbasis unix lainnya dalam hal fungsi, efisiensi, dan kecepatan. *Apache* juga merupakan sebuah aplikasi *multi-platform*, *responsive*, dan juga *open-source*.

Hampir 50% dari penyedia layanan *web* dalam sepuluh tahun terakhir ini menggunakan *apache* (Apache, 2015).

## II.10 Heartbeat

*Heartbeat* merupakan aplikasi dasar untuk Linux-HA (Linux *High-Availability*). *HeartBeat* adalah salah satu program yang terpisah atau termasuk dalam fungsi utama dari aplikasi *cluster*. *Heartbeat* bertujuan untuk terus *polling* server dalam konfigurasi *cluster* untuk memastikan bahwa mereka sudah *up* dan merespon (Charles Bookman, 2002). *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. *Heartbeat* akan menjalankan *script* inisialisasi untuk HA dan saat *node* atau server mati dan hidup. Selain itu juga menangani *service-service* apa saja yang akan dijalankan pada saat server menjadi aktif (Purnomo, 2017). Cara kerja *heartbeat* dapat dilihat pada gambar 2.1 berikut.



**Gambar 2.1** Ilustrasi *Heartbeat*

## II.11 DRBD (*Distributed Replicated Block Device*)

DRBD merupakan perangkat lunak yang memberikan solusi replikasi *storage block device* (*hardisk*, *partisi*, *logical volume*, dll.) antar dua server yang identik pada sistem operasi Linux. DRBD melakukan replikasi melalui jaringan LAN, atau bisa disebut RAID-1 over network (Syafrizal, 2013). Konsep kerja dari DRBD sama persis dengan RAID-1 (*Redudant Array of Independent Disk 1*) yaitu *disk Mirroring*, dimana *harddisk* bekerja dengan prinsip cermin, semua data yang ada kembar identik satu dan yang lainnya.

Perbedaan antara DRBD dan RAID-1 hanya terletak pada perangkat saja, jika RAID-1 bekerja pada *harddisk*, maka DRBD bekerja pada server, sistem

yang dibuat oleh DRBD adalah mengintegrasikan kedua penyimpanan pada masing-masing *server* sehingga kedua *server* terlihat hanya memiliki satu tempat penyimpanan yang terpusat (Irfani, 2015).

Sistem yang dibuat oleh DRBD yaitu menggunakan dua *server* yang masing-masing *server* terdapat *block device* yang berfungsi untuk menyimpan data secara terpusat, jika salah satu *server* mengalami gangguan maka akan disinkronisasikan dengan *server* kedua untuk mereplikasi data yang terdapat di *server* pertama, sehingga pada sistem akan terlihat menggunakan penyimpanan terpusat pada satu *server* saja. Konsep kerja dari DRBD adalah *primary secondary*, *primary* berjalan pada *block device* / *block drive* pada *server* kedua. Aplikasi DRBD akan berjalan pada *server* utama dan mengakses *block drive* pada *server* utama. Setiap data yang disimpan pada *server* utama akan disimpan pada penyimpanan lokal dan disinkronisasikan ke *server* kedua dan bersamaan akan dikirimkan ke *secondary server* pada bagian yang sama. DRBD pada *secondary server* akan aktif dan bisa diakses jika *primary server* mati atau mengalami gangguan (Purnomo, 2017).

## II.12 Pengujian Sistem

Pengujian yang dilakukan pada penelitian ini menggunakan pendekatan eksperimental yang digunakan untuk mencari pengaruh perlakuan tertentu terhadap kondisi yang terkendali. Pengujian dilakukan untuk mengetahui pengaruh penggunaan metode *failover clustering* terhadap beberapa parameter sehingga dapat dilihat perbedaan apabila *web server* dibangun dengan penerapan metode tersebut. Adapun parameter yang akan diuji yaitu *availability*, *workload*, dan QoS (*Quality of Service*).

### II.12.1 Availability

Berdasarkan dokumen ISO 2382-14 (1997) *information technology part 14* tentang *reliability, maintainability and availability*, *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.

### II.12.2 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*.

### II.12.3 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 sebuah jaringan untuk menyediakan layanan yang lebih baik lagi bagi layanan trafik yang melewatinya. Qos suatu jaringan merujuk ke tingkat kecepatan dan keandalan penyampaian berbagai jenis beban data dalam suatu komunikasi yang sangat ditentukan oleh kualitas jaringan yang digunakan.

Berdasarkan pernyataan diatas maka dapat disimpulkan bahwa Qos (*Quality of Service*) adalah kemampuan suatu jaringan untuk menyediakan layanan yang baik dengan menyediakan *bandwidth*, mengatasi *jitter* dan *delay* (Yanto, 2013). Dengan mengkonfigurasi prioritas *traffic* yang melewati jaringan Qos didesain untuk membantu *client* menjadi lebih produktif dengan memastikan bahwa user mendapatkan performansi yang handal dari aplikasi-aplikasi berbasis jaringan. Qos mengacu pada kemampuan jaringan untuk menyediakan layanan yang lebih baik pada *traffic* jaringan tertentu melalui teknologi yang berbeda-beda. Tujuan dari Qos adalah untuk memenuhi kebutuhan-kebutuhan layanan yang berbeda yang menggunakan infrastruktur yang sama (Yanto, 2013).

Performansi mengacu keningkat kecepatan dan keandalan penyampaian berbagai jenis beban akses dalam suatu komunikasi. Beberapa parameter yang dapat digunakan dalam menentukan performansi yaitu.

### 1. *Throughput*

*Throughput* yaitu kecepatan (*rate*) transfer data efektif yang diukur dalam satuan bps. *Throughput* merupakan jumlah total kedatangan paket yang sukses yang diamati pada tujuan selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut (Yanto, 2013). Berikut merupakan nilai *throughput* sesuai dengan versi TIPHON (*Telecommunication and internet protocol harmonisazation over network*).

**Tabel 2.3** *Throughput*

Kategori <i>Throughput</i>	<i>Throughput</i>	Indeks
Sangat Bagus	100 %	4
Bagus	75 %	3
Sedang	50 %	2
Jelek	< 25 %	1

### 2. *Packet Loss*

*Packet loss* merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang, dapat terjadi karena *collison* dan *congestion* pada jaringan dan hal ini berpengaruh pada semua aplikasi karena retransmisi akan mengurangi efisiensi jaringan secara keseluruhan meskipun jumlah *bandwidth* cukup tersedia untuk aplikasi tersebut. Umumnya perangkat jaringan memiliki *buffer* untuk menampung data yang diterima. Jika terjadi kongesti dalam waktu yang cukup lama, *buffer* akan penuh dan data baru tidak dapat diterima (Yanto, 2013). Semakin tinggi *packet loss* maka semakin buruk kinerja suatu server. 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 harmonisazation over network*) sebagai berikut.

**Tabel 2.4** *Packet loss*

Kategori Degradasi	<i>Packet loss</i>	Indeks
--------------------	--------------------	--------



Sangat Bagus	0 %	4
Bagus	3 %	3
Sedang	15 %	2
Jelek	25 %	1

### 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 (Yanto, 2013). Adapun komponen *delay* adalah sebagai berikut.

**Tabel 2.5** Komponen *Delay (Latency)*

<b>Jenis <i>Delay</i></b>	<b>Keterangan</b>
<i>Algoritmatic Delay</i>	<i>Delay</i> ini disebabkan oleh standar <i>codec</i> yang digunakan. Contohnya, algoritma <i>delay</i> G.711 adalah 0 ms.
<b>Jenis <i>Delay</i></b>	<b>Keterangan</b>
<i>Pakectization Delay</i>	<i>Delay</i> yang disebabkan oleh pengakumulasian bit <i>voice</i> sampel ke <i>frame</i> , seperti contohnya standar G711 untuk <i>payLoad</i> 160 <i>bytes</i> memakan waktu 20 ms.
<i>Serialization Delay</i>	<i>Delay</i> ini terjadi karena adanya waktu yang dibutuhkan untuk pentransmisiian paket IP dari sisi orginating (pengiriman).
<i>Propagation Delay</i>	<i>Delay</i> ini terjadi karena perambatan atau perjalanan paket IP ke media transmisi ke alamat tujuan. Seperti contohnya <i>delay</i> propagasi di dalam kabel akan memakan waktu 4-6 s per kilometer.
<i>Coder (processing) Delay</i>	Waktu yang diperlukan untuk <i>digital signal processing</i> (DSP) untuk mengompres sebuah blok PCM, nilainya bervariasi tergantung dari <i>codec</i> dan kecepatan <i>processor</i> .

Nilai *delay* sesuai dengan versi TIPHON (*Telecommunication and internet protocol harmonisazation over network*) sebagai berikut.

**Tabel 2.6** *Delay (Latency)*

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

#### 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*. *Jitter* lazimnya disebut variasi *delay* dan berhubungan erat dengan *latency*, yang menunjukkan banyaknya variasi *delay* pada transmisi data jaringan. *Delay* antrian pada *router* dan *switch* dapat menyebabkan *jitter* (Yanto, 2013). Menurut Joesman (2008) terdapat empat kategori penurunan performansi jaringan berdasarkan nilai *peak jitter* sesuai dengan versi TIPHON (*Telecommunication and internet protocol harmonisazation over network*) yaitu sebagai berikut.

**Tabel 2.7** *Jitter*

Kategori Degradasi	Besar <i>Jitter</i>	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

Berdasarkan nilai parameter-parameter QoS maka untuk selanjutnya dapat dilakukan perhitungan terhadap performansi. Berikut adalah tabel kualitas

QoS versi TIPHON (*Telecommunication and internet protocol harmonisazation over network*).

**Tabel 2.8** Indeks Parameter QoS

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

### II.13 Media Pengujian

Dalam penelitian ini akan digunakan beberapa *tools* untuk membantu mendapatkan nilai analisa terhadap pengujian yang dilakukan. Adapun *tools* pengujian yang dimaksud sebagai berikut:

#### II.13.1 Siege

Siege adalah utilitas pengujian HTTP *load testing* dan *benchmarking* yang dapat digunakan untuk mengukur kinerja *web server* saat berada di bawah tekanan. Siege mengevaluasi jumlah data yang ditransfer, waktu respon server, *transaction rate*, *throughput*, *concurrency*, dan waktu program berjalan kembali dengan baik. Siege menawarkan tiga mode operasi: regresi, simulasi internet, dan *brute force* (Krout, 2018).

#### II.13.2 Iperf

IPerf adalah *tool* berbentuk *command-line* yang digunakan dalam mendiagnosis masalah kecepatan jaringan dengan mengukur *throughput* jaringan maksimum yang dapat ditangani oleh server. *Tool* ini sangat berguna ketika mengalami masalah kecepatan jaringan, karena dapat digunakan untuk menentukan server mana yang tidak dapat mencapai *throughput* maksimum (Linode, 2018).



## **BAB III**

### **METODOLOGI PENELITIAN**

#### **III.1 Alat yang Digunakan**

Alat yang digunakan dalam penelitian ini yaitu berupa perangkat keras dan perangkat lunak.

##### **III.1.1 Perangkat Keras**

Adapun perangkat keras yang digunakan dalam penelitian ini antara lain sebagai berikut.

1. Komputer Master *Web server*:

- HP Proliant DL180 G6
- Prosesor Intel(R) Xeon(R)  
CPU E5606 @2.13 GHz
- RAM 4 GB DDR3
- Harddisk 500 GB

2. Komputer Slave *Web server*:

- Aspire M3985
- Prosesor Intel(R) Core(TM) i3-2120  
CPU @3.30GHz
- RAM 4GB DDR3
- Harddisk 500 GB

3. Monitor : Acer P166HQL 14"
4. Switch : Cisco SF90-24 24 Port
5. Kabel UTP : *Straight*

### **III.1.2 Perangkat Lunak**

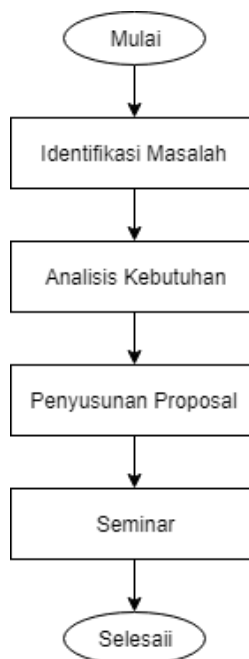
Adapun perangkat lunak yang digunakan dalam penelitian ini antara lain sebagai berikut.

1. Sistem Operasi : Ubuntu 16.04 Xenial Xerus
2. *Remote Aplikasi* : PuTTY 0.70
3. *Aplikasi Failover* : Heartbeat
4. *Web server* : Apache2
5. PHP : PHP5
6. *Aplikasi Penguji* : Siege, Iperf

### **III.2 Metodologi Penelitian**

Metodologi penelitian yang akan dilakukan dapat dijelaskan pada tahapan berikut.

## Persiapan



**Gambar 3.1** Diagram Alir Tahapan Persiapan

Terdapat empat tahapan yang dilakukan dalam persiapan penelitian tugas akhir ini. Berikut ini adalah penjelasan dari gambar alir penelitian.

1. Identifikasi Masalah

Pada tahap ini dilakukan identifikasi permasalahan yang diangkat menjadi penelitian tugas akhir. Proses identifikasi dilakukan dengan melakukan observasi pada gedung informatika.

2. Analisis Kebutuhan

Pada tahap ini dilakukan analisis terhadap kebutuhan sistem yang akan dibuat agar penelitian dapat berjalan sebagaimana mestinya.

3. Penyusunan Proposal

Pada tahap ini dilakukan penulisan proposal berdasarkan data dan informasi yang telah didapat pada tahapan penelitian sebelumnya.

4. Seminar

Pada tahap akhir dari persiapan adalah dilakukannya seminar yakni mempresentasikan hasil penelitian dan proposal kepada dosen pembimbing dan peserta seminar.

### **III.3 Identifikasi Masalah**

Pada tahap ini dilakukan identifikasi permasalahan yang diangkat menjadi penelitian tugas akhir. Proses identifikasi dilakukan dengan melakukan observasi pada gedung informatika melalui pengamatan dan pencatatan sistem atau arsitektur jaringan yang sedang digunakan sehingga dapat dilihat kemungkinan yang dapat dilakukan untuk pembuatan sistem *web server*.

### **III.4 Analisis Kebutuhan**

Pada tahap ini dilakukan analisis terhadap kebutuhan sistem yang akan dibuat agar penelitian dapat berjalan sebagaimana mestinya. Setiap kebutuhan akan ditentukan pada tahap ini sebelum masuk ke tahap selanjutnya sehingga dapat dikembangkan sebuah sistem yang sesuai dengan kebutuhan.

### **III.5 Penyusunan Proposal**

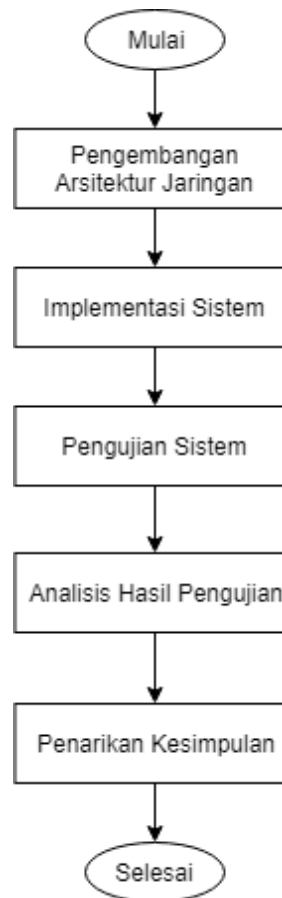
Pada tahap ini dilakukan penulisan proposal berdasarkan data dan informasi yang telah didapat pada tahapan penelitian sebelumnya. Penulisan proposal ini dilakukan untuk menggambarkan secara singkat rencana kegiatan penelitian yang akan dilakukan agar dapat memahami apa yang direncanakan.

### **III.6 Seminar**

Pada tahap akhir dari persiapan adalah dilakukannya seminar yakni mempresentasikan hasil penelitian dan proposal kepada dosen pembimbing dan peserta seminar.



### Pelaksanaan Pembuatan Sistem



**Gambar 3.2** Diagram Alir Tahapan Pelaksanaan Pembuatan Sistem

Pada tahapan pembuatan sistem merupakan gambaran dari sistem yang diajukan yaitu sistem *web server clustering*, implementasi metode *failover*, dan *web server* yang bersifat *high availability*. Kemudian dilakukan tahapan pembuatan sistem seperti pengembangan arsitektur jaringan dan pembuatan *web server*. Setelah itu dilakukan pengujian sistem, analisis hasil pengujian, dan penarikan kesimpulan.

1. Pengembangan Arsitektur Jaringan

Pada tahap ini dilakukan pengembangan arsitektur jaringan untuk meletakkan posisi server yang terhubung pada jaringan yang sudah ada.

2. Implementasi Sistem

Pada tahap ini dilakukan implementasi sistem yakni sistem yang telah dibuat kemudian diimplementasikan pada jaringan yang telah ada yang dibangun sesuai dengan perancangan yang dibuat.

### 3. Pengujian Sistem

Pada tahap ini dilakukan pengujian terhadap *web server* apakah dapat berfungsi sesuai dengan perancangan atau tidak dengan menerapkan metode *failover*.

### 4. Analisis Hasil Pengujian

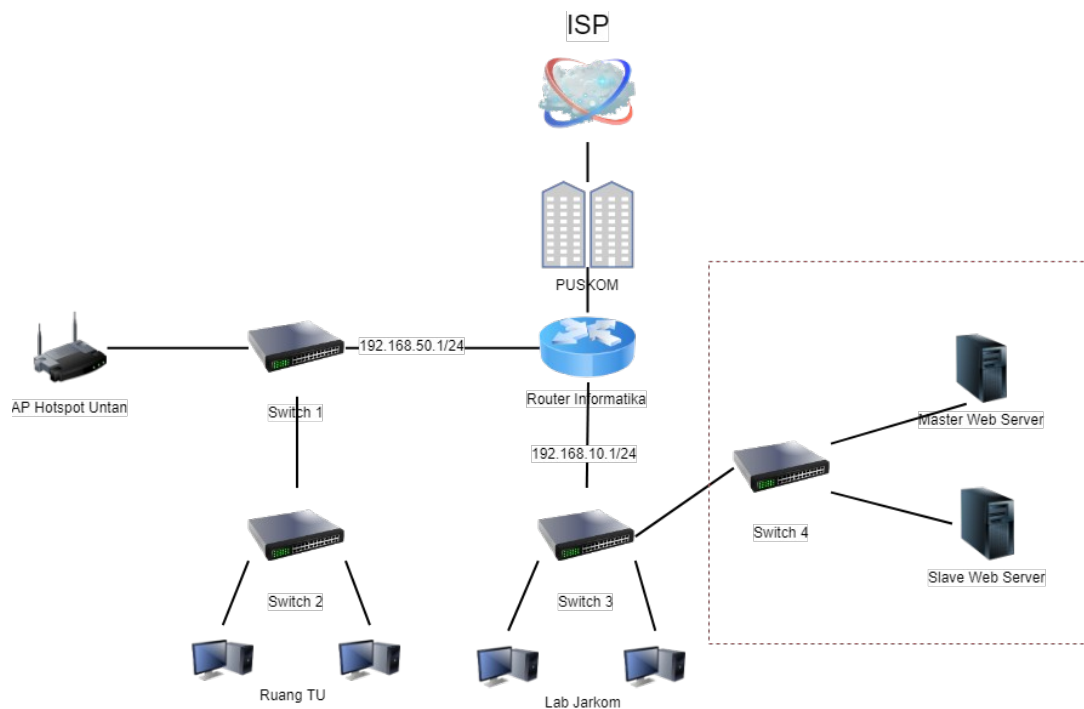
Pada tahap ini dilakukan analisis terhadap hasil pengujian yang telah dilakukan pada tahap sebelumnya.

### 5. Penarikan Kesimpulan

Berdasarkan hasil analisis hasil pengujian yang telah dilakukan maka akan ditarik kesimpulan mengenai apakah sistem yang telah dibangun dapat berjalan baik sesuai dengan yang diharapkan.

## III.7 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 3.3 berikut.



**Gambar 3.3** Pengembangan Arsitektur Jaringan

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

### III.8 Implementasi Sistem

Pada tahap ini dilakukan implementasi sistem dilakukan dengan membangun *web server*. Pembangunan *web server* dilakukan dengan melakukan konfigurasi IP Address, update dan upgrade karnel Linux, konfigurasi SSH server, konfigurasi FTP server, konfigurasi Apache2, instalasi MySQL server, instalasi PHP, konfigurasi Heartbeat, dan konfigurasi DRBD. Sistem yang telah dibangun kemudian diimplementasikan pada jaringan yang telah ada yang dibangun sesuai dengan perancangan yang dibuat.

### III.9 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 webserver*. Adapun pengujiannya yakni sebagai berikut.

#### 3.9.1 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*. 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.

1. Skenario 1 : kedua *web server* dalam posisi hidup kemudian *primary server* dimatikan.
2. Skenario 2 : *primary server* dalam kondisi mati dan *secondary server* dalam posisi hidup kemudian *primary server* dihidupkan.
3. 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.
4. Skenario 4 : kedua *web server* dalam kondisi mati kemudian *primary server* dihidupkan.
5. Skenario 5 : kedua *web server* dalam kondisi mati kemudian *secondary server* dihidupkan.
6. 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.
7. 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.

Melalui tujuh skenario tersebut, dilakukan pengujian sebanyak lima kali yang bertujuan untuk mendapatkan jumlah waktu *downtime* yang terjadi selama pemberian gangguan sehingga dapat ditentukan total *downtime* yang didapatkan dari tiap skenario. Nilai *downtime* diambil dari total waktu layanan *web server* tidak dapat menangani *request* dari *client* selama skenario pengujian dilakukan yang didapat melalui hasil *capture tool* siege. Hasil pengujian tersebut akan dirangkum pada tabel berikut.

**Tabel 3.1** Hasil Pengujian *Downtime*

Skenario	<i>Downtime</i> tiap pengujian (Second)					Rata-rata (Second)
	Ke-1	Ke-2	Ke-3	Ke-4	Ke-5	
1						
2						
...						
7						

Berdasarkan hasil *capture tool* siege selama pengujian juga akan diambil waktu *respond time* yang akan dirangkum pada tabel berikut.

**Tabel 3.2** Hasil Pengujian *Respond Time*

Skenario	<i>Respond time</i> tiap pengujian ( <i>Second</i> )					Rata-rata ( <i>Second</i> )
	Ke-1	Ke-2	Ke-3	Ke-4	Ke-5	
1						
2						
...						
7						

Kemudian dengan lama pengujian yang dilakukan selama 24 jam melalui pemberian gangguan hanya seperti yang terdapat pada skenario pengujian maka nilai dari *availability* akan didapatkan dengan persamaan berikut.

$$Availability = \frac{uptime}{uptime + downtime} \times 100\% \quad (3.1)$$

Hasil perhitungan terhadap *availability* melalui persamaan diatas akan ditampilkan pada tabel berikut.

**Tabel 3.3** Hasil Perhitungan *Availability*

Skenario	Rata-rata <i>Uptime</i> ( <i>Second</i> )	Rata-rata <i>Downtime</i> ( <i>Second</i> )	<i>Availability</i> (%)
1			
2			
...			
7			
<b>Rata-rata</b>			

*Uptime* merupakan waktu *web server* berjalan normal sehingga semua *request* dari *client* dapat ditangani oleh *web server*, sedangkan *downtime* merupakan waktu *web server* tidak dapat melayani *request* dari *client* yang disebabkan terjadinya gangguan pada *web server* sehingga layanan tidak tersedia dan *client* tidak dapat mengakses layanan dari *web server*.

### 3.9.2 Pengujian *Workload*

Pengujian *workload* dilakukan dengan menggunakan *benchmarking tool* yaitu dengan menggunakan aplikasi *siege*. Pengujian ini dilakukan melalui pemberian beban akses dengan jumlah beban yang ditentukan secara bervariasi hingga server tidak mampu lagi menanganinya. Pengujian dilakukan dengan memberikan beban akses dimulai dari kelipatan 200 *request* secara bersamaan hingga server tidak mampu menangani semua *request* tersebut. Selama itu maka dapat dilihat nilai *workload* dari server yang diuji melalui *tool siege*. Pengujian dilakukan pada *web server* kluster dan *webserver* tunggal sehingga akan terlihat perbedaan dari kedua jenis *server* tersebut. Hasil pengujian *workload* akan dirangkum dalam tabel berikut.

**Tabel 3.4** Hasil Pengujian *Workload*

<b>Jumlah Request</b>	<b>Dengan Failover</b>		<b>Tanpa Failover</b>	
	<i>Request</i> Berhasil	<i>Request</i> Gagal	<i>Request</i> Berhasil	<i>Request</i> Gagal
200				
400				
...				

Setelah didapatkan jumlah maksimum *request* yang dapat ditangani oleh *web server* secara bersamaan, hasil tersebut kemudian dijadikan acuan untuk mendapatkan hasil pasti dari *workload* dengan melakukan pengujian ulang. Hasil pengujian ulang tersebut akan ditampilkan pada tabel berikut.

**Tabel 3.5** Hasil Pengujian Ulang

<b>Jumlah Request</b>	<b>Dengan Failover</b>		<b>Tanpa Failover</b>	
	<i>Request</i> Berhasil	<i>Request</i> Gagal	<i>Request</i> Berhasil	<i>Request</i> Gagal
<b>Rata-rata</b>				

Penjelasan isi dari kolom pada tabel 3.4 dan tabel 3.5 adalah sebagai berikut.

1. Jumlah *request*: berisi jumlah *request* yang diberikan untuk ditangani *web server* secara bersamaan sebagai bahan pengujian.
2. *Request* berhasil: berisi total *request* yang dapat ditangani *web server* secara bersamaan saat diberikan *request* ketika pengujian dilakukan.
3. *Request* gagal: berisi total *request* yang tidak dapat ditangani *web server* secara bersamaan saat diberikan *request* ketika pengujian dilakukan.

### 3.9.3 Pengujian QoS (Quality of service)

Pengujian *Quality of service* dilakukan untuk melihat bagaimana kemampuan sebuah jaringan dalam menyediakan layanan yang baik bagi layanan trafik yang melewatinya. Untuk pengujiannya dilakukan dengan menggunakan aplikasi *iperf* dan *siege* yang bertugas memonitoring dan mengambil data pengujian. Dari data pengujian tersebut akan dilakukan perhitungan untuk mengetahui performa dari jaringan dalam menyediakan layanan. Pengujian terhadap performa dilakukan menggunakan parameter QoS. Parameter yang digunakan dalam pengujian yaitu *throughput*, *packet loss*, *delay (latency)*, dan *jitter*.

#### 1. Pengujian *Throughput*

Pengujian *throughput* dilakukan dengan melakukan pengukuran *throughput* pada *secondary server* dan *primary server*. *Throughput* digunakan untuk mengetahui kemampuan *web server* dalam memberikan layanan secara benar terhadap *request* yang datang secara bersamaan. Pengujian ini dilakukan menggunakan aplikasi *iperf* dengan mengirimkan paket data sebesar 1-5 MB. Hasil pengujian *throughput* akan dirangkum dalam tabel berikut.

**Tabel 3.6** Data Pengamatan Parameter *Throughput*

Paket Dikirim (MB)	Dengan Failover		Tanpa Failover	
	Lama Pengamatan (Second)	Bandwidth (Mbps)	Lama Pengamatan (Second)	Bandwidth (Mbps)
1				
...				

5				
---	--	--	--	--

Penjelasan isi dari kolom pada tabel 3.6 adalah sebagai berikut.

- Paket dikirim : berisi jumlah paket data yang dikirimkan sebagai bahan pengujian dalam satuan MB.
- Lama pengamatan : berisi waktu yang ditempuh untuk menyelesaikan pengujian.
- Bandwidth* : berisi kecepatan transfer yang didapatkan selama pengujian yang diukur menggunakan satuan *mega bit per second*.

Data hasil pengamatan selanjutnya akan dilakukan perhitungan terhadap *throughput*. Nilai *throughput* didapatkan dengan menggunakan persamaan berikut.

$$\text{Throughput} = \frac{\text{paket data yang diterima}}{\text{lama pengamatan}} \quad (3.2)$$

$$\text{Throughput(\%)} = \frac{\text{throughput}}{\text{bandwidth}} \quad (3.3)$$

Hasil perhitungan tersebut akan ditentukan kategori indeks berdasarkan versi TIPHON (*Telecommunication and internet protocol harmonisazation over network*) yang akan ditampilkan pada tabel berikut.

**Tabel 3.7** Hasil Perhitungan Parameter *Throughput*

Paket Dikirim (MB)	Dengan <i>Failover</i>		Tanpa <i>Failover</i>	
	<i>Throughput</i> (%)	Kategori	<i>Throughput</i> (%)	Kategori
1				
...				
5				
<b>Rata-rata</b>				

Penjelasan isi dari kolom pada tabel 3.7 adalah sebagai berikut.

- Paket dikirim : berisi jumlah paket data yang telah dikirimkan selama pengujian *throughput*.



- b. *Throughput* : berisi persentase *throughput* yang didapatkan pada perhitungan dengan persamaan (3.3) dari data hasil pengujian.
- c. Kategori : berisi kategori indeks yang didapatkan dengan menggunakan versi TIPHON.

## 2. Pengujian Packet Loss

Pengujian *packet loss* dilakukan dengan pengamatan pada jumlah paket yang hilang disaat terjadi komunikasi. Pengujian *packet loss* dilakukan untuk melihat seberapa besar jumlah paket yang hilang saat *web server* menangani permintaan dari *client*.. Pengujian dilakukan dengan membandingkan *web server* dengan *failover* dengan tanpa *failover*. *Web server* akan diberikan beban akses dengan jumlah yang ditentukan secara bervariasi yaitu sebesar 200 hingga 1000 *request*. Selama itu *tools siege* akan *capture* jumlah paket yang masuk sehingga didapatkan data hasil pengujian yang akan dirangkum pada tabel berikut.

**Tabel 3.8** Data Pengamatan Parameter *Packet Loss*

Jumlah <i>Request</i>	Paket Diterima	
	Dengan <i>Failover</i>	Tanpa <i>Failover</i>
200		
...		
1000		

Penjelasan isi dari kolom pada tabel 3.8 adalah sebagai berikut.

- a. Jumlah *request* : berisi jumlah *request* yang diberikan kepada *web server* sebagai bahan pengujian untuk ditangani secara bersamaan oleh *web server*.
- b. Paket diterima : berisi total paket yang diterima melalui pemberian *request* pada pengujian.

Data hasil pengamatan selanjutnya akan dilakukan perhitungan terhadap *packet loss*. Nilai *packet loss* didapatkan dengan menggunakan persamaan berikut.

$$Packet\ loss = \frac{(paket\ dikirim - paket\ diterima) \times 100\%}{paket\ dikirim} \quad (3.4)$$

Hasil perhitungan tersebut akan ditentukan kategori indeks berdasarkan versi TIPHON (*Telecommunication and internet protocol harmonisazation over network*) yang akan ditampilkan pada tabel berikut.

**Tabel 3.9** Hasil Perhitungan Parameter *Packet Loss*

Jumlah Request	Dengan Failover		Tanpa Failover	
	Packet loss (%)	Kategori	Packet loss (%)	Kategori
200				
...				
1000				
<b>Rata-rata</b>				

Penjelasan isi dari kolom pada tabel 3.9 adalah sebagai berikut.

- Jumlah *request* : berisi jumlah *request* yang diberikan kepada *web server* sebagai bahan pengujian untuk ditangani secara bersamaan oleh *web server*.
- Packet loss* : berisi persentase *packet loss* yang didapatkan pada perhitungan dengan persamaan (3.4) dari data hasil pengujian.
- Kategori : berisi kategori indeks yang didapatkan dengan menggunakan versi TIPHON.

### 3. Pengujian *Delay* (Latency)

Pengujian *delay* dilakukan dengan melakukan pengamatan pada jumlah paket yang diterima selama pengujian. *Web server* akan diberikan beban akses dengan jumlah yang ditentukan secara bervariasi yaitu dengan mengirimkan beban akses kepada *web server* dengan jumlah *request* tertentu yang dimulai dari 200 sampai 1000 *request*. Adapun pengujian dilakukan dengan menggunakan *tools* *siege* yang akan *capture* data pengamatan selama pengujian. Hasil pengujian *delay* akan dirangkum pada tabel berikut.

**Tabel 3.10** Data Pengamatan Parameter *Delay*

<b>Jumlah Request</b>	<b>Dengan Failover</b>		<b>Tanpa Failover</b>	
	Lama Pengamatan (second)	Paket Diterima	Lama Pengamatan (second)	Paket Diterima
200				
...				
1000				

Penjelasan isi dari kolom pada tabel 3.10 adalah sebagai berikut.

- Jumlah *request* : berisi jumlah *request* yang diberikan kepada *web server* sebagai bahan pengujian untuk ditangani secara bersamaan oleh *web server*.
- Lama pengamatan : berisi waktu yang ditempuh untuk menyelesaikan pengujian.
- Paket diterima : berisi total paket yang diterima melalui pemberian *request* pada pengujian.

Data hasil pengamatan selanjutnya akan dilakukan perhitungan terhadap nilai *delay*. Nilai *delay* didapatkan dengan menggunakan persamaan berikut.

$$\text{Delay rata - rata} = \frac{\text{Lama pengamatan}}{\text{paket diterima}} \quad (3.5)$$

Hasil perhitungan tersebut akan ditentukan kategori indeks berdasarkan versi TIPPHON (*Telecommunication and internet protocol harmonisazation over network*) yang akan ditampilkan pada tabel berikut.

**Tabel 3.11** Hasil Perhitungan Parameter *Delay*

<b>Jumlah Request</b>	<b>Dengan Failover</b>		<b>Tanpa Failover</b>	
	<i>Delay (ms)</i>	Kategori	<i>Delay (ms)</i>	Kategori
200				
<b>Jumlah Request</b>	<b>Dengan Failover</b>		<b>Tanpa Failover</b>	
	<i>Delay (ms)</i>	Kategori	<i>Delay (ms)</i>	Kategori
...				

1000				
<b>Rata-rata</b>				

Penjelasan isi dari kolom pada tabel 3.11 adalah sebagai berikut.

- a. Jumlah *request* : berisi jumlah *request* yang diberikan kepada *web server* sebagai bahan pengujian untuk ditangani secara bersamaan oleh *web server*.
- b. *Delay* : berisi nilai *delay* yang didapatkan pada perhitungan dengan persamaan (3.5) dari data hasil pengujian.
- c. Kategori : berisi kategori indeks yang didapatkan dengan menggunakan versi TIPHON.

#### 4. Pengujian *Jitter*

Pengujian *jitter* dilakukan dengan melakukan pengamatan pada jumlah paket yang diterima selama pengujian. Selain itu nilai *delay* juga diperhatikan selama pengujian yang akan digunakan untuk mendapatkan nilai *jitter*. Selama pengujian *web server* akan diberikan beban akses dengan jumlah yang ditentukan secara bervariasi yaitu dimulai dari 200 hingga 1000 *request*. Kemudian dengan *tools siege* yang mengcapture data pengamatan selama pengujian. Hasil pengujian akan dirangkum pada tabel berikut.

**Tabel 3.12** Data Pengamatan Parameter *Jitter*

Jumlah Request	Dengan <i>Failover</i>		Tanpa <i>Failover</i>	
	Lama Pengamatan (second)	Paket Diterima	Lama Pengamatan (second)	Paket Diterima
200				
...				
1000				

Penjelasan isi dari kolom pada tabel 3.12 adalah sebagai berikut.

- a. Jumlah *request* : berisi jumlah *request* yang diberikan kepada *web server* sebagai bahan pengujian untuk ditangani secara bersamaan oleh *web server*.

- b. Lama pengamatan : berisi waktu yang ditempuh untuk menyelesaikan pengujian.
- c. Paket diterima : berisi total paket yang diterima melalui pemberian *request* pada pengujian.

Data hasil pengamatan selanjutnya akan dilakukan perhitungan terhadap nilai *jitter*. Nilai *jitter* didapatkan dengan menggunakan persamaan berikut.

$$Jitter = \frac{\text{Total variasi delay}}{\text{Total paket diterima}} \quad (3.6)$$

Total variasi *delay* diperoleh dari :

$$\text{Total variasi delay} = \text{lama pengamatan} - \text{delay rata-rata} \quad (3.7)$$

Hasil perhitungan tersebut akan ditentukan kategori indeks berdasarkan versi TIPHON (*Telecommunication and internet protocol harmonisazation over network*) yang akan ditampilkan pada tabel berikut.

**Tabel 3.13** Hasil Perhitungan Parameter *Jitter*

Jumlah Request	Dengan Failover		Tanpa Failover	
	<i>Jitter (ms)</i>	Kategori	<i>Jitter (ms)</i>	Kategori
200				
...				
1000				
<b>Rata-rata</b>				

Penjelasan isi dari kolom pada tabel 3.13 adalah sebagai berikut.

- a. Jumlah *request* : berisi jumlah *request* yang diberikan kepada *web server* sebagai bahan pengujian untuk ditangani secara bersamaan oleh *web server*.
- b. *Jitter* : berisi nilai *jitter* yang didapatkan pada perhitungan dengan persamaan (3.8) dari data hasil pengujian.
- c. Kategori : berisi kategori indeks yang didapatkan dengan menggunakan versi TIPHON.

### **III.10 Analisis Hasil Pengujian**

Pada tahap ini dilakukan analisis terhadap hasil pengujian yang telah dilakukan pada tahap sebelumnya. Dari hasil pengujian akan dilakukan perbandingan antara *web server* dengan *failover* dengan *webserver* tanpa *failover* sehingga akan terlihat perbedaan dari kedua server tersebut dan dapat diperoleh hasil dari penggunaan metode *failover* pada *clustering webserver*.

### **III.11 Penarikan Kesimpulan**

Berdasarkan hasil analisis hasil pengujian yang telah dilakukan maka akan ditarik kesimpulan mengenai apakah sistem yang telah dibangun dapat berjalan baik sesuai dengan yang diharapkan. Penarikan kesimpulan dilakukan dengan cara melihat analisis hasil pengujian. Penarikan kesimpulan dilakukan berdasarkan kepada tujuan dilakukan penelitian sehingga hasil dari penarikan kesimpulan merupakan jawaban dari pertanyaan yang disampaikan pada tujuan dilakukan penelitian.

### **III.12 Penulisan Laporan**

Tahapan penulisan laporan yakni berfokus pada penelitian laporan tugas akhir. Penulisan laporan disesuaikan dengan format penulisan yang telah ditetapkan. Setelah penulisan selesai akan dilakukan seminar hasil kemudian sidang terbuka.

## BAB IV

### IMPLEMENTASI DAN PENGUJIAN

#### 4.1 Implementasi Sistem

Implementasi dilakukan berdasarkan perancangan yang telah dibuat. Implementasi sistem dilakukan dengan membangun kedua *web server* berdasarkan kebutuhan yang sama. *Web server* yang dibangun kemudian diletakkan kedalam arsitektur jaringan Informatika sesuai dengan pengembangan arsitektur jaringan pada gambar 3.3. Pengembangan *web server* dalam penelitian ini dibuat dengan mengimplementasikan teknik *clustering* dimana *server* yang dibuat lebih dari satu dan dikelompokkan sebagai sebuah entitas dengan menggunakan metode *failover*. *Web server* dibangun menggunakan beberapa konfigurasi antara lain konfigurasi IP Address, *update* dan *upgrade* Kernal Linux, konfigurasi SSH *server*, konfigurasi FTP *server*, konfigurasi Apache2, instalasi MySQL *server*, instalasi PHP, dan konfigurasi Hearbeat. *Web server* dibangun menggunakan sistem operasi berbasis *open source* yaitu Linux Ubuntu Server 16.04 Xenial Xerus. Berikut ini adalah sejumlah proses yang dikerjakan untuk mengimplementasikan sistem.

##### 4.1.1 Konfigurasi IP Address

Komputer *server* yang digunakan sebagai *web server* menggunakan *ethernet card* di tiap *server* yaitu *ethernet card* pertama dengan nama *interface* *eth0* yang kedua menggunakan IP *Dynamic* yang langsung terhubung ke jaringan Lab. informatika. Informasi IP Address dari kedua *ethernet card* pada tiap *server* tersebut dapat dilihat pada tabel 4.1 berikut.

**Tabel 4.1** Informasi IP Address *web server*

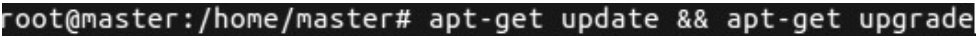
<i>Server</i>	<i>Address</i>	<i>Netmask</i>	<i>Broadcast</i>
<i>Master</i>	10.144.13.61	255.255.240.0	10.144.15.255
<i>Slave</i>	10.144.13.62	255.255.240.0	10.144.15.255

Tabel 4.1 merupakan informasi *ip address* yang didapatkan dari konfigurasi secara dhcp jaringan gedung Informatika. *Web server* yang dibangun yakni berbasis *intranet* sehingga *ip address* yang digunakan berasal dari konfigurasi secara dhcp jaringan gedung Informatika.

#### 4.1.2 Update dan Upgrade Kernal Linux

*Update* merupakan pembaharuan yang dirancang untuk memperbaiki masalah dengan memperbaharui sebuah program komputer atau penambahan data pendukung termasuk juga memperbaiki kelemahan-kelemahan dan meningkatkan kegunaan atau kinerjanya. Sedangkan *upgrade* digunakan untuk menginstal versi terbaru dari semua paket saat ini yang akan diinstal pada sistem. Hal ini dilakukan untuk mencegah terjadinya *error* pada saat instalasi dan konfigurasi paket sistem.

Untuk melakukan *update* dan *upgrade* kernal Linux dapat dilakukan dengan mengetikkan perintah seperti pada gambar 4.2 berikut.



```
root@master:/home/master# apt-get update && apt-get upgrade
```

Gam

**bar 4.2** Perintah *Update* dan *Upgrade* Kernal Linux

Hal ini akan melakukan *update* dan *upgrade* secara bersamaan. Jika terjadi kegagalan saat melakukan *update* dan *upgrade*, lakukan *restart server* dan ulangi kembali *update* dan *upgrade*.

#### 4.1.3 Konfigurasi SSH Server

SSH *Server* merupakan *protocol* jaringan yang dapat mengakomodasi transfer data antara dua buah komputer melalui jalur komunikasi yang aman. SSH memungkinkan untuk melakukan kendali *server* secara jarak jauh atau *remote*. Versi SSH yang digunakan pada waktu penulisan skripsi adalah OpenSSH\_7.2p2.

Langkah awal untuk konfigurasi SSH adalah dengan cara menginstal paket ssh dengan menggunakan perintah “*apt-get install openssh-server*”. Setelah paket SSH terinstal selanjutnya dapat menambahkan *user* dengan menggunakan perintah “*adduser [nama\_user\_baru]*”. Lebih jelas untuk menambahkan *user* dapat dilihat pada gambar 4.3 berikut.



```

root@master:/home/master# adduser mahasiswa
Adding user `mahasiswa' ...
Adding new group `mahasiswa' (1002) ...
Adding new user `mahasiswa' (1002) with group `mahasiswa' ...
Creating home directory `/home/mahasiswa' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for mahasiswa
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
root@master:/home/master#

```

**Gambar 4.3** Menambah user baru

Agar user yang baru dibuat langsung mengakses direktori home di “/var/www/html/directory\_pengguna” dapat dilakukan dengan membuat direktori baru melalui perintah “`mkdir /var/www/html/nama_direktori`” seperti gambar 4.6 berikut.

```

root@master:/home/master# mkdir /var/www/html/mahasiswa

```

**Gambar 4.4** Perintah membuat direktori

Kemudian pindahkan home direktori user ke direktori yang telah dibuat dengan perintah “`usermod --home /var/www/html/nama_direktori nama_user`” seperti gambar 4.5 berikut.

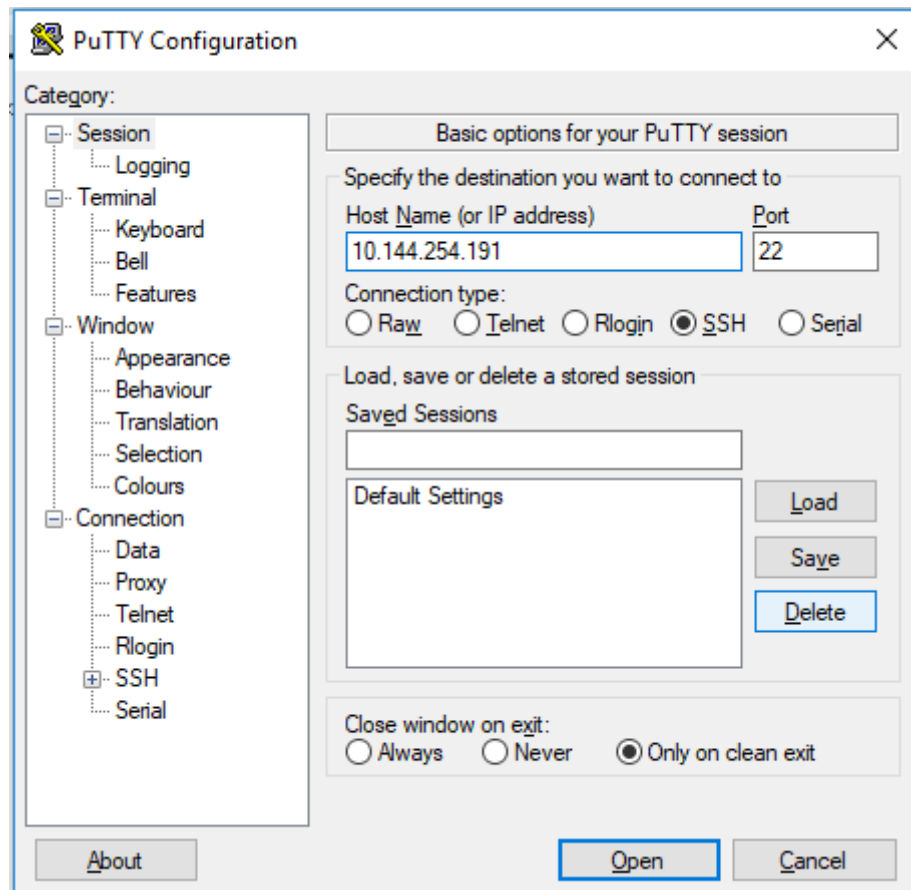
```

root@master:/home/master# usermod --home /var/www/html/mahasiswa mahasiswa

```

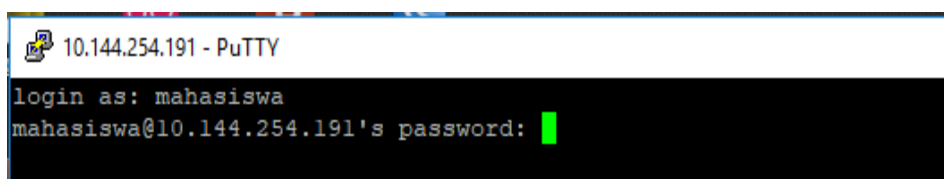
**Gambar 4.5** Perintah *setting* akses direktori user

Kemudian untuk menguji bahwa SSH server telah berjalan dengan baik dapat dilakukan dengan mengakses server melalui PC client menggunakan aplikasi PuTTY. Buka aplikasi PuTTY kemudian masukkan IP Address dan port server kemudian pilih *open* untuk mengakses server secara *remote*.



**Gambar 4.6** Tampilan aplikasi PuTTY

Setelah tombol *open* diklik maka akan terbuka sebuah jendela baru, kemudian *login* dengan menggunakan *username* dan *password server* seperti gambar 4.7 berikut.



**Gambar 4.7** Login dengan aplikasi PuTTY

Jika berhasil masuk berarti konfigurasi SSH *server* telah berjalan dengan baik.

```

10.144.254.191 - PuTTY
login as: mahasiswa
mahasiswa@10.144.254.191's password:
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-33-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

60 packages can be updated.
0 updates are security updates.

New release '18.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

mahasiswa@master:~$

```

**Gambar 4.8** PuTTY berhasil *login* ke server

#### 4.1.4 Konfigurasi FTP Server

Konfigurasi FTP server dilakukan agar administrator dapat mengunggah file *website* pada *web server*. Langkah pertama yang dilakukan untuk melakukan konfigurasi FTP server yaitu dengan menginstal paket *vsftpd* pada masing-masing server. Untuk menginstalnya dilakukan dengan menggunakan perintah berikut.

```
root@master:/home/master# apt-get install vsftpd
```

**Gambar 4.9** Perintah instal paket *vsftpd*

Setelah paket terinstal pada *web server*, langkah selanjutnya yang dilakukan adalah melakukan konfigurasi pada file “*/etc/vsftpd.conf*”. Konfigurasi yang dilakukan dapat dilihat pada gambar 4.10 berikut.

```

chroot_local_user=YES
chroot_list_enable=YES
# (default follows)
chroot_list_file=/etc/vsftpd.chroot_list

```

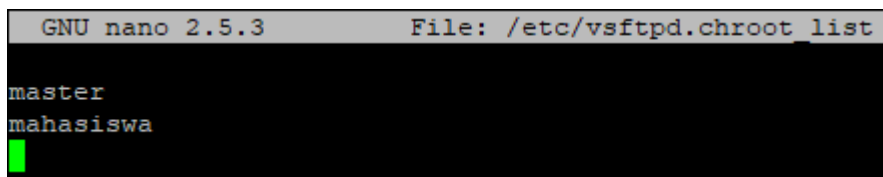
**Gambar 4.10** Konfigurasi pada file *vsftpd.conf*

Kemudian agar *user* dapat melakukan perubahan pada file pada server dapat dilakukan dengan menghilangkan tanda “#” pada gambar 4.13 berikut.

```
write_enable=YES
```

**Gambar 4.11** Baris perintah akses file server

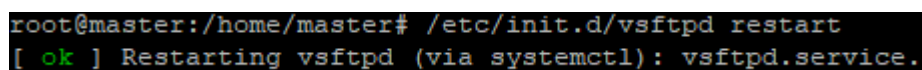
Langkah selanjutnya adalah memasukkan *username* pada file “*/etc/vsftpd.chroot\_list*”. Konfigurasi tersebut bertujuan agar dapat memberikan akses kepada *user* untuk dapat menggunakan FTP server. Konfigurasi yang dilakukan dapat dilihat pada gambar 4.12 berikut.



```
GNU nano 2.5.3 File: /etc/vsftpd.chroot_list
master
mahasiswa
█
```

**Gambar 4.12** Konfigurasi pada file *vsftpd.chroot\_list*

Kemudian lakukan *restart* pada FTP server sehingga konfigurasi yang telah dilakukan dapat berjalan dengan baik. Perintah untuk melakukan *restart* pada vsftpd dapat dilihat pada gambar 4.13 berikut.



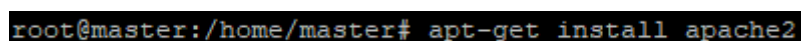
```
root@master:/home/master# /etc/init.d/vsftpd restart
[ ok ] Restarting vsftpd (via systemctl): vsftpd.service.
```

**Gambar 4.13** Perintah *restart* vsftpd

Agar *user* FTP mendapatkan hak akses pada file di server dapat dilakukan dengan perintah “*chown -R user\_ftp /var/www/html/direktori\_user*”. Kemudian unggah file *website* pada server dapat dilakukan dengan menggunakan aplikasi WinSCP.

#### 4.1.5 Konfigurasi Apache2

Penelitian ini menggunakan Apache2 sebagai *web server*. Apache2 adalah aplikasi *web server* yang digunakan untuk mengolah berkas – berkas *website* agar dapat ditampilkan pada *browser* milik *client*. Pertama kali yang dilakukan dalam konfigurasi Apache2 yaitu melakukan instalasi paket aplikasi pada masing-masing *web server* dengan menggunakan perintah pada gambar 4.14 berikut.



```
root@master:/home/master# apt-get install apache2
```

**Gambar 4.14** Perintah instal paket apache2

Kemudian lakukan konfigurasi pada file “*/etc/apache2/sites-enabled/000-default.conf*”. Konfigurasi tersebut dilakukan

untuk mendefinisikan letak folder *website* yang akan ditampilkan oleh *web server*. Konfigurasi tersebut dapat dilihat pada gambar 4.15 berikut.

```
ServerAdmin webmaster@localhost
DocumentRoot /var/www/html
```

**Gambar 4.15** Hasil konfigurasi pada file *000-default.conf*

Langkah selanjutnya adalah melakukan *restart* pada *service* *apache2* agar konfigurasi tersebut dapat berjalan. Perintah untuk melakukan *restart service* *apache2* dapat dilihat pada gambar 4.16 berikut.

```
root@master:~# /etc/init.d/apache2 restart
```

**Gambar 4.16** Perintah *restart service* *apache2*

#### 4.1.6 Instalasi MySQL Server

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL yang *multi-user*. MySQL merupakan perangkat lunak yang bersifat *open source* yang berarti bahwa memungkinkan siapa saja untuk menggunakan tanpa harus membelinya. MySQL digunakan sebagai media penyimpanan suatu informasi yang dapat menangani *database* dalam jumlah yang sangat besar dan dapat diakses oleh banyak *user*. Perintah untuk melakukan instalasi MySQL server dapat dilihat pada gambar 4.17 berikut.

```
root@master:~# apt-get install mysql-server
```

**Gambar 4.17** Perintah instal MySQL server

Setelah instalasi berhasil dilakukan, langkah selanjutnya yakni dengan memasukkan perintah “*mysql\_secure\_installation*”.

#### 4.1.7 Instalasi PHP

PHP merupakan sebuah bahasa pemrograman yang dirancang untuk menghasilkan halaman-halaman *website* yang dinamis. Hampir seluruh *website* yang ada saat ini telah menggunakan bahasa pemrograman ini. Instalasi PHP dilakukan agar *web server* dapat menjalankan *script* PHP yang terdapat didalam *website* yang telah diunggah. Perintah untuk melakukan instalasi PHP dapat dilihat pada gambar 4.18 berikut.

```
root@master:~# apt-get install php libapache2-mod-php php-mcrypt php-mysql
```

**Gambar 4.18** Perintah instal PHP

#### 4.1.8 Konfigurasi Heartbeat

Heartbeat merupakan sebuah perangkat lunak terpisah yang digunakan untuk membangun sistem *clustering* pada suatu *server*. Tugas dari perangkat lunak ini yaitu terus mem-*polling server* yang berada di dalam *cluster* untuk memastikan sistem berjalan dan merespon.

Langkah pertama untuk melakukan konfigurasi Heartbeat yaitu dengan menginstal paket aplikasi Heartbeat pada sistem. Adapun perintah yang digunakan pada instalasi paketnya dapat dilihat pada gambar 4.19 berikut.

```
root@master:~# apt-get install heartbeat
```

**Gambar 4.19** Perintah instal Heartbeat

Setelah paket Heartbeat terinstal maka langkah selanjutnya adalah melakukan konfigurasi pada file “*/etc/ha.d/ha.cf*”. Hasil dari konfigurasi tersebut dapat dilihat pada gambar 4.20 berikut.

```
keepalive 2
warntime 5
deadtime 15
initdead 90
udpport 694
auto_failback on
bcast eth0
node master slave
```

**Gambar 4.20** Hasil konfigurasi file ha.cf

Kemudian buat file *authkeys* pada “*/etc/ha.d/authkeys*”. Isi dari file tersebut dapat dilihat pada gambar 4.21 berikut.

```
auth 2
2 crc
```

**Gambar 4.21** Isi file *authkeys*

Selanjutnya ganti *permission* pada file *authkeys* dengan perintah pada gambar 4.22 berikut.

```
root@master:~# chmod 600 /etc/ha.d/authkeys
```

**Gambar 4.22** Perintah mengganti *permission* authkeys

Langkah selanjutnya yaitu membuat file baru dengan nama *haresources* pada “*/etc/ha.d/haresources*”. Kemudian isi file tersebut dengan *script* pada gambar 4.23 berikut.

```
master IPaddr::192.168.1.222/24/eth0 apache2
```

**Gambar 4.23** Isi file *haresources*

Setelah semua file berhasil dikonfigurasi maka langkah selanjutnya yaitu dengan menjalankan *service* *heartbeat* dengan menggunakan perintah pada gambar 4.24 berikut.

```
root@master:~# /etc/init.d/heartbeat start
```

**Gambar 4.24** Perintah menjalankan *service* *heartbeat*

Untuk konfigurasi pada *slave server* dapat dilakukan dengan menggunakan konfigurasi yang sama pada *master server*.

#### 4.1.9 Instalasi dan konfigurasi DRBD

Sebelum melakukan proses instalasi DRBD, perlu ditambahkan *harddisk* pada masing-masing *primary server* dan *secondary server* sebesar 2Gb, *harddisk* ini akan digunakan sebagai *harddisk* cluster DRBD. Setelah ditambahkan cek dengan perintah *# fdisk -l*. Hasilnya seperti gambar 4.25 berikut.

```

I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xaf66a054

Device      Boot      Start        End    Sectors    Size Id Type
/dev/sda1   *                2048 968402943 968400896 461,8G 83 Linux
/dev/sda2                968404990 976771071    8366082      4G  5 Extended
/dev/sda5                968404992 976771071    8366080      4G 82 Linux swap / Solaris

Disk /dev/sdb: 149,1 GiB, 160041885696 bytes, 312581808 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x3b84eefd

Device      Boot      Start        End    Sectors    Size Id Type
/dev/sdb1   *                2048    206847     204800    100M  7 HPFS/NTFS/exFAT
/dev/sdb2                206848 122882047 122675200    58,5G  7 HPFS/NTFS/exFAT
/dev/sdb3                122884096 126790345    3906250     1,9G 83 Linux

```

**Gambar 4.25** Penambahan Hardisk

1. Instalasi paket DRBD dapat dilakukan melalui repository Ubuntu pada kedua *server* dengan perintah: `apt-get install drbd8-utils drbdlinks`
2. Edit file `drbd.conf` dengan perintah `#etc /etc/drbd.conf`. Isi file tersebut seperti gambar 4.26 berikut.

```

global {
    dialog-refresh 1;
    usage-count yes;
    minor-count 5;
}
common {
    syncer {
        rate 10M;
    }
}
resource r0 {
    protocol C;
    disk {
        on-io-error detach;
    }
    syncer {
        rate 10M;
        al-extents 257;
    }
    on master {
        device /dev/drbd0;
        address 10.144.13.61:7788;
        meta-disk internal;
        disk /dev/sdb3;
    }
    on slave {
        device /dev/drbd0;
        address 10.144.13.62:7788;
        meta-disk internal;
        disk /dev/sdb3;
    }
}

```

**Gambar 4.26** Konfigurasi `drbd.conf`



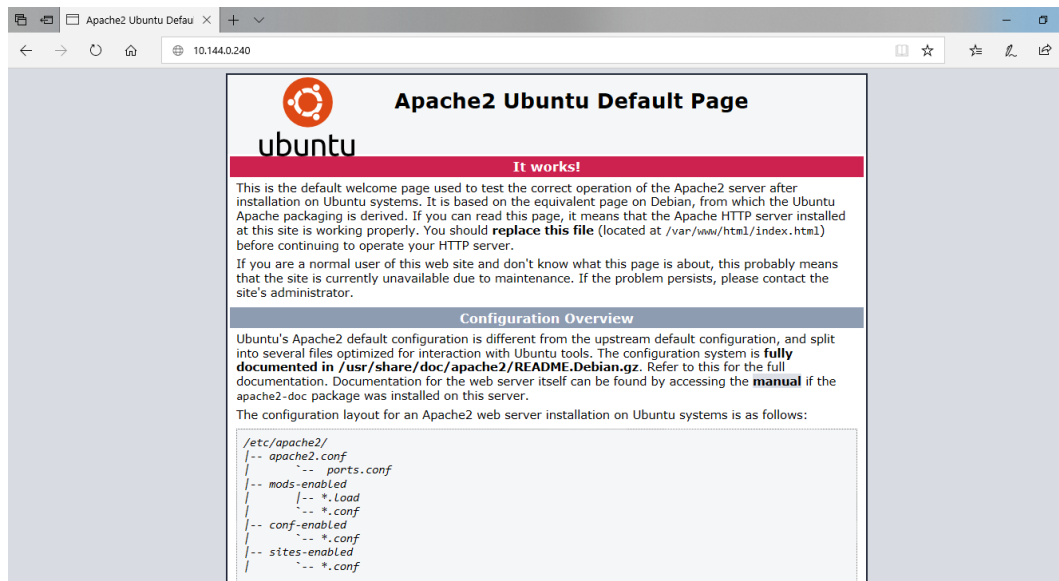
3. Buat meta data disk kedua *server* virtual dan jalankan service DRBD dengan perintah: `drbdadm create-md r0` Service `drbd` start
4. Ubah *server* utama menjadi *primary server* dengan perintah: `drbdsetup /dev/drbd0 primary --overwritedata-of-peer`

## 4.2 Pengujian Availability

Pengujian *availability* dilakukan dengan menguji tingkat ketersediaan layanan terhadap *web server* yang telah diterapkan sistem *failover*. Adapun pengujian dilakukan dengan skenario *web server down* yaitu dengan mematikan *web server*. Dalam kondisi tersebut akan dilakukan percobaan akses layanan *web server* apakah tersedia atau tidak. Selain itu juga dilakukan perhitungan terhadap lamanya waktu tidak dapat memberikan layanan ketika *web server down* yang disebut dengan istilah *downtime* sehingga dengan data tersebut dapat dilakukan analisis untuk menentukan *availability* dari suatu *web server*.

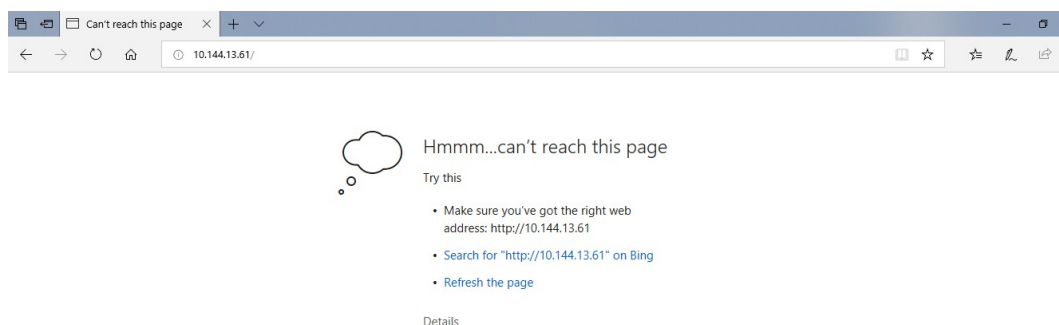
### 4.2.1 Hasil Pengujian Availability

Pengujian *availability* dilakukan dengan mengamati kinerja *web server* ketika diberikan gangguan. Pengujian dilakukan dengan skenario *down* pada *web server*. *Primary web server* akan dibuat seolah-olah *down* dengan cara memamatkannya 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 4.27** Hasil pengujian akses *website* pada *web server* dengan *failover*

Gambar 4.27 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 4.28** Hasil pengujian akses *website* pada *web server* tanpa *failover*

Gambar 4.28 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 *downtime* yang terjadi selama *web server* mengalami gangguan. Pengujian *downtime* 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 *downtime*. Pengujian dilakukan sebanyak lima kali untuk mendapatkan nilai rata-rata *downtime* dengan menggunakan *tool* *siege*. Pengujian dilakukan dengan mengirimkan sejumlah *request* selama pengujian kemudian *downtime* 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* yang terdapat pada lampiran A ditampilkan pada tabel 4.2 berikut.

**Tabel 4.2** Hasil Pengujian *Downtime*

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

Tabel 4.2 merupakan hasil pengujian *downtime* 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 keadaan *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 *downtime* 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 4.3** Hasil Pengujian *Respond Time*

Skenario	<i>Respond time</i> tiap pengujian (Second)					Rata-rata (Second)
	Ke-1	Ke-2	Ke-3	Ke-4	Ke-5	
1	0,07	0,07	0,16	0,07	0,09	<b>0,09</b>
2	0,13	0,06	0,07	0,07	0,07	<b>0,08</b>
3	0,03	0,08	0,05	0,22	0,06	<b>0,09</b>
4	0,03	0,01	0,02	0,01	0,03	<b>0,02</b>
5	0,10	0,06	0,05	0,12	0,06	<b>0,08</b>
6	0,05	0,11	0,06	0,02	0,20	<b>0,09</b>
7	0,07	0,15	0,10	0,29	0,23	<b>0,17</b>

Tabel 4.3 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 (3.1) berikut.

**Tabel 4.4** Hasil Perhitungan *Availability*

Skenario	Rata-rata <i>Uptime</i> ( <i>Second</i> )	Rata-rata <i>Downtime</i> ( <i>Second</i> )	<i>Availability</i> (%)
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
<b>Rata-rata</b>	<b>86317,300</b>	<b>82,7</b>	<b>99,90</b>

Tabel 4.4 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%.

#### 4.2.2 Analisis Pengujian *Availability*

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 *downtime* selama beberapa detik sedangkan *web server* tanpa *failover* tidak dapat diakses oleh *client* saat terjadi *down*. *Web server* dengan *failover* masih bisa diakses 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.

*Downtime* yang terjadi pada pengujian *web server* dengan *failover* tersebut merupakan waktu perpindahan *web server* saat terjadi *down*. Hasil pengujian *downtime* 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. Hal tersebut 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%. Hasil pengujian menunjukkan bahwa tingginya persentase tingkat *availability* dipengaruhi oleh lamanya *downtime*. Semakin cepat sistem *failover* dalam mengembalikan layanan maka persentase tingkat *availability* semakin besar dan begitu juga sebaliknya.

### 4.3 Pengujian Workload

Pengujian ini dilakukan untuk mengetahui batasan yang dapat ditangani *web server* dalam menangani *request* dari *client*. Adapun pengujian dilakukan dengan mengirimkan sejumlah *request* kepada *web server* secara bersamaan menggunakan *tool* *siege* yang kemudian dapat dilihat berapakah jumlah *request* yang dapat ditangani oleh *web server* secara bersamaan.

#### 4.3.1 Hasil Pengujian *Workload*

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* terdapat pada lampiran B ditampilkan pada tabel 4.5 berikut.

**Tabel 4.5** Hasil Pengujian *Workload*

Jumlah <i>Request</i>	Dengan <i>Failover</i>		Tanpa <i>Failover</i>	
	<i>Request</i> Berhasil	<i>Request</i> Gagal	<i>Request</i> Berhasil	<i>Request</i> 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 4.5 menunjukkan 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 menangani 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 tersebut menggunakan *tool* *siege* terdapat pada lampiran C dapat dilihat pada tabel 4.6 berikut.

**Tabel 4.6** Hasil pengujian ulang

Jumlah <i>Request</i>	Dengan <i>Failover</i>		Tanpa <i>Failover</i>	
	<i>Request</i> Berhasil	<i>Request</i> Gagal	<i>Request</i> Berhasil	<i>Request</i> Gagal

900	844	56	844	56
Jumlah Request	Dengan <i>Failover</i>		Tanpa <i>Failover</i>	
	<i>Request</i> Berhasil	<i>Request</i> Gagal	<i>Request</i> Berhasil	<i>Request</i> Gagal
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
806	806	0	806	0
807	803	4	807	0
808	-	-	808	0
809	-	-	798	11

Tabel 4.6 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*.

#### 4.3.2 Analisis Pengujian *Workload*

Tabel 4.6 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*.

#### 4.4 Pengujian QoS (*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*.

##### 4.4.1 Hasil Pengujian Parameter QoS

Pengujian ini 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 sebelumnya. Adapun pengujian yang dilakukan adalah sebagai berikut.

##### 4.4.1.1 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 (3.2) agar didapatkan nilai *throughput* dari *web server*. Kemudian dengan menggunakan persamaan (3.3) maka akan didapatkan tingkat *throughput* dalam bentuk persen (%). Pengujian dilakukan dengan mengamati kedua sisi *web*

server sehingga melalui data pengujian dengan *tool* iperf yang terlampir pada lampiran C didapat data pengamatan seperti yang ditunjukkan pada tabel 4.7 berikut.

**Tabel 4.7** 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 4.7 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 (3.2) dan persamaan (3.3) didapatkan nilai *throughput* sebagai berikut.

$$\text{Throughput} = \frac{\text{paket data yang diterima}}{\text{lama pengamatan}}$$

$$= \frac{3145728 \text{ Byte}}{1,4 \text{ second}}$$

$$= 338250,32 \text{ Bps}$$

$$= 330,32 \text{ KBps}$$

$$\text{Throughput (\%)} = \frac{\text{throughput}}{\text{bandwidth}} \times 100\%$$

$$= \frac{2194,286 \text{ KBps}}{2312,5 \text{ KBps}} \times 100\%$$

$$= 94,89 \% \text{ (pembulatan)}$$

**Tabel 4.8** Hasil Perhitungan Parameter *Throughput*

Paket Dikirim (MB)	Dengan <i>Failover</i>		Tanpa <i>Failover</i>	
	<i>Throughput</i> (%)	Kategori	<i>Throughput</i> (%)	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
<b>Rata-rata</b>	<b>96,11</b>	<b>Bagus</b>	<b>97</b>	<b>Bagus</b>

Tabel 4.8 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.

#### 4.4.1.2 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 (3.4) untuk mendapatkan nilai dari *packet loss*. Pengujian dilakukan dengan mengamati kedua sisi *web server* sehingga melalui

data pengujian *tool* siege yang terlampir pada lampiran B didapat data pengamatan seperti yang ditunjukkan pada tabel 4.9 berikut.

**Tabel 4.9** Data Pengamatan Parameter *Packet Loss*

Jumlah <i>Request</i>	Paket Diterima	
	Dengan <i>Failover</i>	Tanpa <i>Failover</i>
200	200	200
400	400	400
600	600	600
800	800	800
1000	823	931

Tabel 4.9 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 (3.4) sebagai berikut.

$$\text{Packet loss} = \frac{(\text{paket dikirim} - \text{paket diterima}) \times 100\%}{\text{paket dikirim}}$$

$$\text{Packet loss} = \frac{(600 - 600) \times 100\%}{600}$$

$$= \frac{0\%}{600}$$

$$= 0\%$$

**Tabel 4.10** Hasil Perhitungan Parameter *Packet Loss*

Jumlah	Dengan <i>Failover</i>	Tanpa <i>Failover</i>
--------	------------------------	-----------------------

Request				
	Packet loss (%)	Kategori	Packet loss (%)	Kategori
200	0	Sangat Bagus	0	Sangat Bagus
400	0	Sangat Bagus	0	Sangat Bagus
600	0	Sangat Bagus	0	Sangat Bagus
Jumlah Request	Dengan Failover		Tanpa Failover	
	Packet loss (%)	Kategori	Packet loss (%)	Kategori
800	0	Sangat Bagus	0	Sangat Bagus
1000	17,7	Jelek	6,9	Sedang
<b>Rata-rata</b>	<b>3,54</b>	<b>Sedang</b>	<b>1,38</b>	<b>Bagus</b>

Tabel 4.10 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*.

#### 4.4.1.3 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 *capture* 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 (3.5). Adapun pengujian dilakukan dengan mengamati kedua sisi *web server* untuk melihat perbedaannya sehingga didapat data pengamatan melalui pengujian *tool* *siege* yang terdapat pada lampiran B seperti yang ditunjukkan pada tabel 4.11 berikut.

**Tabel 4.11** Data Pengamatan Parameter *Delay*

<b>Jumlah Request</b>	<b>Dengan Failover</b>		<b>Tanpa Failover</b>	
	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 4.11 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 (3.5) sebagai berikut.

$$\begin{aligned}
 \text{Delay rata-rata} &= \frac{\text{lama pengamatan}}{\text{paket diterima}} \\
 &= \frac{22,34 \text{ s}}{600} \\
 &= 0,037 \text{ s} \\
 &= 37,23 \text{ ms (pembulatan)}
 \end{aligned}$$

**Tabel 4.12** Hasil Perhitungan Parameter *Delay*

<b>Jumlah Request</b>	<b>Dengan <i>Failover</i></b>		<b>Tanpa <i>Failover</i></b>	
	<i>Delay (ms)</i>	Kategori	<i>Delay (ms)</i>	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
<b>Rata-rata</b>	<b>34,16</b>	<b>Sangat Bagus</b>	<b>32,84</b>	<b>Sangat Bagus</b>

Tabel 4.12 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 dengan kategori sangat bagus 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.

#### 4.4.1.4 Pengujian *Jitter*

Pengujian *jitter* dilakukan dengan mengamati jumlah paket yang diterima beserta waktu pengamatannya selama pengujian dilakukan. Dikarenakan *jitter* memiliki hubungan erat dengan *latency* maka data *delay* rata-rata juga diperlukan untuk mendapatkan 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* (dapat dilihat pada lampiran) yang kemudian akan dilakukan

perhitungan dengan persamaan (3.6) untuk mendapatkan nilai *jitter* pada *web server*. Pengujian dilakukan dengan mengamati kedua sisi *web server* sehingga didapat data pengamatan melalui pengujian *tool siege* yang terdapat pada lampiran B seperti yang ditunjukkan pada tabel 4.13 berikut.

**Tabel 4.13** 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 4.13 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 (3.7) terlebih dahulu. Kemudian berdasarkan data yang telah diperoleh maka nilai *jitter* dapat dihitung menggunakan persamaan (3.6) sebagai berikut.

$$\begin{aligned}
 \text{Total variasi } delay &= \text{lama pengamatan} - \text{delay rata-rata} \\
 &= 22,34s - 0,037s \\
 &= 22,3s
 \end{aligned}$$



$$\begin{aligned}
 Jitter &= \frac{\text{total variasi delay}}{\text{total paket diterima}} \\
 &= \frac{22,3s}{600} \\
 &= 37,17ms
 \end{aligned}$$

**Tabel 4.14** Hasil Perhitungan Parameter *Jitter*

Jumlah Request	Dengan <i>Failover</i>		Tanpa <i>Failover</i>	
	<i>Jitter (ms)</i>	Kategori	<i>Jitter (ms)</i>	Kategori
200	22,79	Bagus	27,31	Bagus
400	22,22	Bagus	21	Bagus
600	37,17	Bagus	36,39	Bagus
800	48,86	Bagus	40,87	Bagus
1000	39,42	Bagus	38,27	Bagus
<b>Rata-rata</b>	<b>34,09</b>	<b>Bagus</b>	<b>32,77</b>	<b>Bagus</b>

Tabel 4.14 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.

#### 4.4.2 Analisis Pengujian Parameter QoS

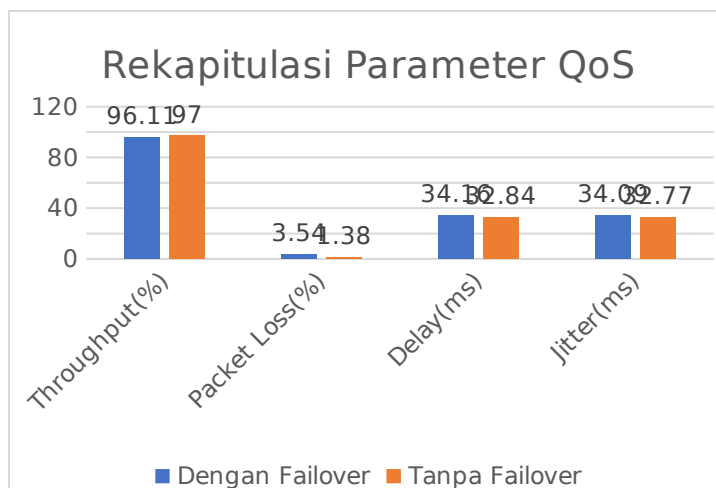
Hasil pengujian parameter QoS pada kedua jenis *web server* diperoleh dari perhitungan masing-masing parameter QoS yaitu *throughput*, *packet loss*, *delay*, dan *jitter*. Hasil dari perhitungan masing-masing parameter selanjutnya dikelompokkan dan ditentukan nilai indeks berdasarkan standar TIPHON (*Telecommunication and internet protocol harmonisazation over network*).

Berikut merupakan hasil rekapitulasi pengujian parameter QoS secara keseluruhan.

**Tabel 4.15** Rekapitulasi Perhitungan Parameter QoS

Parameter QoS	Dengan Failover			Tanpa Failover		
	Rata-rata	Kategori	Nilai Indeks	Rata-rata	Kategori	Nilai Indeks
Throughput(%)	96,11	Bagus	3	97	Bagus	3
Parameter QoS	Dengan Failover			Tanpa Failover		
	Rata-rata	Kategori	Nilai Indeks	Rata-rata	Kategori	Nilai Indeks
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
<b>Rata-rata</b>		<b>Memuaskan</b>	<b>3</b>		<b>Memuaskan</b>	<b>3,25</b>

Tabel 4.15 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 4.29** Hasil rekapitulasi parameter QoS

Gambar 4.29 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.

#### 4.5 Analisis Hasil Pengujian Keseluruhan

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*. *Web server* meskipun bisa diakses saat terjadi *down* masih terdapat waktu beberapa saat *web server* tersebut tidak dapat memberikan layanan. Hal tersebut 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 keseluruhan nilai *availability* maka didapatkan rata-rata persentase tingkat *availability* sebesar 99,90%. Nilai persentase tersebut menunjukkan tingginya tingkat *availability* dari *web server* tersebut. 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.

## BAB V

### PENUTUP

#### 5.1 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.

## 5.2 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.