BAB II
DASAR TEORI

2.1 KAJIAN PUSTAKA

Penelitian Dian Pratama yang berjudul "Perbandingan Kinerja Teknologi *Failover* Berbasis Klaster (*Heartbeat*) dengan Teknologi *Failover* berbasis Jaringan (*Keepalived*)" pada tahun 2021. Topologi *High Availability Web Server* dan *High Availability Load Balancing* adalah kajian yang diterapkan. Penerapan *High Availability Web Server* menggunakan *Linux Apache*, Mysql dan *PHP* yang termasuk dalam *LAMP server* untuk konten *web* menggunakan Wordpress. Perangkat lunak *Haproxy* digunakan untuk bertindak sebagai *load balancer* pada penerapan *High Availability Load Balancing*. Penelitian ini menguji nilai *downtime* dan *failback*. Topologi *High Availability web server* dan topologi *High Availability load balancing* mendapatkan hasil *downtime* terkecil pada skenario 1 sebanyak 4 detik dan 4,05 detik, skenario 2 sebanyak 0 detik dan 0 detik, sedangkan skenario 3 sebanyak 0,88 detik dan 0,78 detik. Hasil pengujian *failback* topologi *High Availability web server* dan *High Availability load balancing* mendapatkan nilai terkecil pada skenario 1 sebanyak 0,68 detik dan 0,48 detik, skenario 2 sebanyak 0 detik dan 0 detik sedangkan skenario 3 sebanyak 0,88 detik dan 0,53 detik [4].

Tinjauan pustaka berkaitan dengan penelitian yang sudah diteliti dan yang akan diteliti di masa mendatang. Terdapat beberapa jurnal yang digunakan sebagai acuan. Penelitian Y. Pribadi, A. B. Putra Negara, and M. A. Irwansyah tahun 2020 dengan judul "*Analysis of the Use of the Failover Clustering Method to Achieve High Availability on a Web Server (Case Study: Informatics Department Building)*". Pada Gedung Jurusan Informatika Universitas Tanjungpura, dilakukan penelitian untuk meningkatkan *high availability web server* dengan tujuan untuk menguji efektivitas pengunaan *failover clustering*. *Availability*, *workload* dan *Quality Of Service* (QoS) merupakan beberapa parameter yang diteliti. Penelitian ini mendapatkan nilai *availability* sebesar 99,90% dari hasil perhitungan data pengujian, terdapat perbedaan pada jumlah *workload* yaitu 806 permintaan diproses di *web server* dengan *failover*,

dan 808 permintaan diproses di *web server* tanpa *failover*. Hasil pengujian QoS menunjukkan indikator yang sama yaitu memuaskan untuk *web server* yang menggunakan *failover* maupun tanpa menggunakan *failover* [5].

Penelitian Muhammad Aldi Aditia Putra, Iskandar Fitri dan Agus Iskandar dengan judul "Implementasi *High Availability Cluster Web Server* Menggunakan Virtualisasi *Container Docker*" pada tahun 2020. Penelitian ini menerapkan aturan yaitu menggabungkan beberapa *server* yang bekerja secara bersamaan atau disebut dengan *clustering*. Dalam penerapan *clustering web server,* metode *load balancing* yang menggunakan algoritma *round robin* digunakan agar beban trafik dari setiap *web server* dapat dioptimalkan. *HAProxy* adalah alat yang dipakai dalam *load balancing*. *Haproxy* merupakan *open source* untuk *load balancing* yang dirancang untuk mengatasi permasalahan *request* berlebih pada *web server* (*overload*). Parameter yang diukur termasuk *throughput*, *response time*, *request per second* dan penggunaan CPU. Hasil pegujian sistem *load balancing Haproxy* pada *Least connection* menghasilkan nilai *request* per-detik sebesar 2607.141 req/s dan sebesar 9.25 MB/s untuk *throughput*, sedangkan pada *round robin request* per-detik sebesar 2807.171 req/s dan *throughput* bernilai 9.30 MB/s. Dari hasil tersebut menunjukan bahwa algoritma *least connection* mengungguli algoritma *round robin* [6]. Tabel 2.1 menunjukan keterkaitan dengan penelitian sebelumnya.

Tabel 2.1 Rangkuman Keterkaitan dengan Penelitian Sebelumnya

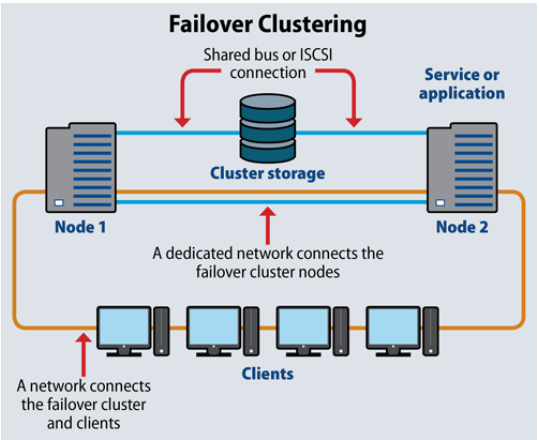
Penelitiann Oleh	Tempat Inplementasi		Parameter yang diteliti		
	Server Fisik/virtual	*Cloud*	Pengujian *Failover*	*Availability*	QoS
Dian Pratama	✓		✓	✓	
Yulizar Pribadi, Arif Bijaksana PN, dan M. Azhar Irwansyah	✓		✓	✓	✓

Penelitiann Oleh	Tempat Inplementasi		Parameter yang diteliti		
	Server Fisik/virtual	*Cloud*	Pengujian *Failover*	*Availability*	QoS
Muhammad Aldi Aditia Putra, Iskandar Fitri dan Agus Iskandar	✓				✓
Nur Ayu Widianingsih		✓	✓	✓	✓

2.2 DASAR TEORI

2.2.1 *Failover Clustering*

Failover juga dikenal sebagai *high-availability clustering* adalah sekelompok *server* yang bekerja sama dalam mempertahankan ketersediaan aplikasi dan layanan dalam tingkat yang tinggi, jika salah satu *server* mengalami kegagalan maka *server* lain akan mengambil alih beban kerjanya tanpa *downtime* [7]. *Failover* disediakan oleh *cluster* yang diimplementasikan untuk meningkatkan ketersediaan layanan. Elemen *cluster* memiliki *node* redundan yang digunakan untuk mencadangkan pada saat terjadi kegagalan pada salah satu komponen [8]. *Clustering* adalah menggabungkan beberapa *server* supaya berada dalam satu jaringan yang sama. Gambar 2.1 menunjukan visualisasi *failover clustering*.



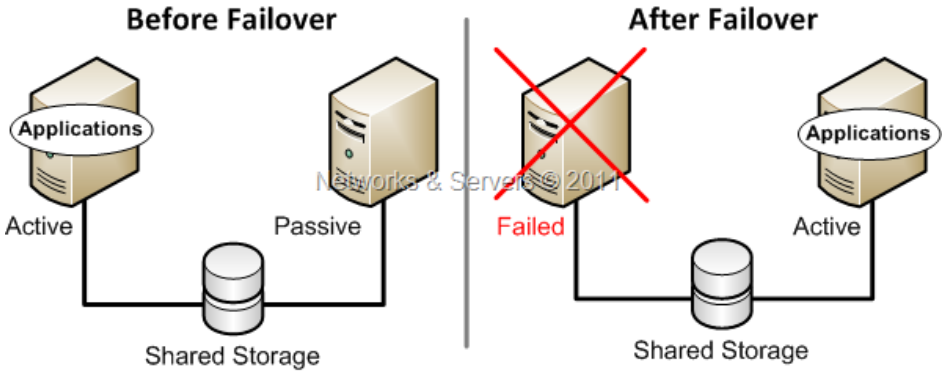
Gambar 2.1 Visualisasi *Failover*

Failover adalah kemampuan secara terus menerus dan otomatis untuk beralih ke sistem cadangan. *Failover* akan aktif jika komponen sistem utama gagal untuk meminimalkan dampak negatif bagi pengguna. Jika terjadi masalah dengan *server* utama, diperlukan *failover* untuk pemulihan. Dalam hal ini, sistem *server* cadangan harus dalam keadaan siaga dan aman dari kegagalan *server* [9].

Terdapat beberapa macam *Failover* sebagai berikut:

1. *Active/Passive Failover*

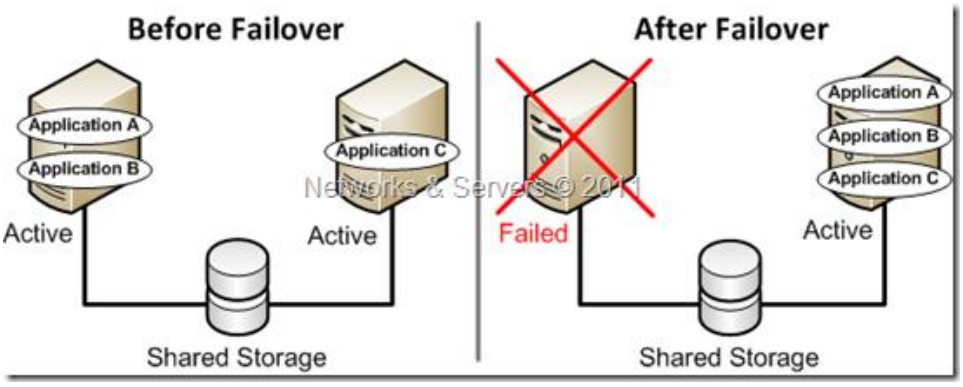
   Terdapat dua komponen *node* pada jenis *failover* ini yaitu satu komponen *node* aktif dan yang lain sebagai komponen *node* pasif. *Node* aktif bertugas menjalankan aplikasi atau tugas tertentu, sedangkan *node* pasif berada pada mode siaga dan tidak melakukan tugas apa pun sampai terdeteksi masalah pada *node* utama atau *node* yang sedang aktif. *Node* pasif akan menggantikan peran yang telah dilakukan sebelumnya oleh *node* aktif saat *node* tersebut mengalami kegagalan [7]. Contoh *active/active failover* terdapat pada Gambar 2.2



Gambar 2.2 *Active/Passive Failover*

2. *Active/Active Failover*

   Dalam jenis ini, seluruh *node* aktif menjalankan program atau proses untuk mewakili beban kerja setiap *node*. Beban kerja pada salah satu *node* yang gagal akan diambil alih oleh *node* aktif tambahan yang terus mengoperasikan seluruh program dan proses yang ada. Tujuan dari aktif/aktif *failover* adalah untuk mencapai *load balancing* [7]. Contoh *active/active failover* terdapat pada Gambar 2.3



Gambar 2.3 ***Active/Active Failover***

2.2.2 *Down Time*

Downtime adalah waktu ketika *web server down* dan tidak tersedia yang terjadi selama *web server* gangguan. Pada *downtime* tidak terdapat kategori pengukuran. Pengetesan *downtine* bertujuan melihat waktu yang dibutuhkan *web server* saat beroperasi kembali setelah pulih dari kegagalan *web server*. Nilai *downtime* pada penelitian diambil saat *web server* diberikan gangguan sesuai dengan skenario penelitian [5].

2.2.3 *Workload*

Workload / beban kerja dilakukan untuk mengevaluasi kemampuan *server* untuk menangani berbagai permintaan *client*. Tujuannya adalah untuk menghitung jumlah permintaan yang dapat ditangani *server* dalam memberikan layanan kepada *client* [5]. Beban kerja yang diberikan pada penelitian dengan memberikan beragam *request* yaitu 500, 1000, 1500, 2000, dan 2500 kepada *server*.

2.2.4 *Haproxy*

Haproxy (*High Availability Proxy*) adalah produk *open source* berbasis *linux* yang menawarkan solusi untuk menciptakan sistem dan mengelola berbagai skenario *load balancing* dan *failover* untuk aplikasi berbasis TCP dan HTTP. Perangkat lunak ini sangat cocok untuk *website* yang memiliki banyak trafik setiap hari dan membutuhkan keteguhan dan kekuatan pemrosesan layer7. *Haproxy* diinstal pada

server front-end yang biasanya memiliki IP statis yang terdaftar dengan DNS. *Haproxy* merupakan solusi gratis, sangat cepat dan dapat diandalkan dalam *high availability*, *load balancing*, dan *proxy* [7]. *Haproxy* berfungsi sebagai tempat implementasi *web server* yang bekerja dengan memindahkan *server* yang mengalami kegagalan ke *server* yang sedang *standby*.

2.2.5 *High Availability*

High Availability (HA) adalah sistem yang bekerja dan beroperasi secara terus menerus tanpa mengalami kegagalan [9]. *High availability* bertujuan untuk menghilangkan waktu henti (*downtime*) yang direncanakan dan tidak direncanakan dari sistem komputer. *Downtime* terencana disebabkan oleh pemeliharaan yang mengganggu sistem operasi. *Downtime* tidak direncanakan biasanya disebabkan oleh peristiwa fisik, seperti kegagalan *hardware*, *software* atau lingkungan [10].

Availability merupakan waktu tersedianya sebuah layanan. *Availability* terdiri dari dua komponen yaitu MTTR (*Mean Time To Restore*) dan MTBF (*Mean Time Between Failures*). MTTR atau dikenal dengan *downtime*, dalam keadaan ini akan diinterupsi untuk didiagnosa, diperbaiki, dan di*recovery*. MTTR adalah rata-rata waktu yang diperlukan komponen untuk melakukan perbaikan [11]. MTBF juga dikenal dengan *uptime* menunjukan berapa lama waktu *respons* sebelum terjadi kegagalan. Jaringan komputer dalam keadaan ini akan beroperasi secara normal hingga terjadi gangguan baru selanjutnya [12]. MTBF adalah rata-rata waktu antara kegagalan sebelum terjadinya kegagalan komponen atau dengan kata lain waktu suatu sistem berjalan dengan normal. *Availability* diperkirakan meggunakan sebutan "*nine*", lebih banyak '*nine'* maka semakin tinggi ketersediaan sistem. *Availability* dapat dihitung menggunakan rumus pada persamaan 2.1 sebagai berikut [13]

$$Availability = \frac{MTBF}{MTBF+MTTR}\ x\ 100 \tag{2.1}$$

2.2.6 *Server*

Komputer yang digunakan untuk mengelola jaringan aplikasi dan telekomunikasi, serta mendistribusikan perangkat lunak dan *database* dikenal sebagai

server. Penyimpanan data merupakan layanan khusus yang dimiliki oleh *Server*. Layanan ini khusus ditujukan untuk pelanggan yang perlu memberikan informasi kepada pengguna atau pengunjungnya [14].

Fungsi *server* biasanya dilakukan oleh *server* komputer meliputi:

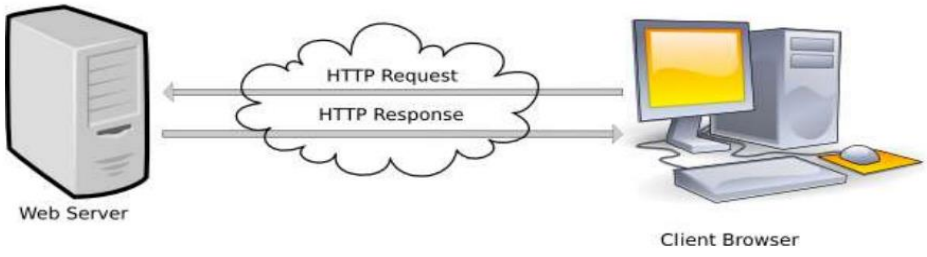
a. Menyimpan aplikasi dan database yang diperlukan untuk masing-masing komputer yang terhubung.
b. Penyediaan karakteristik pada keamanan komputer.
c. Perlindungan *Firewall* dari komputer yang terkoneksi.
d. Memberikan alamat IP untuk komputer yang terhubung [15]

2.2.7 *Web Server*

Menurut Fathansyah *Server web* berfungsi untuk menyediakan perangkat lunak dan perangkat keras (*server*) kepada pengguna melalui protokol komunikasi HTTP atau berbagai protokol lainnya, seperti FTP dan HTTPS, atau melalui *file* yang tersedia melalui URL sebagai layanan akses yang disediakan untuk pengguna [16].

Web server melayani *client* dengan menerima permintaan dan memberikan *respons* HTTP (*HyperText Transfer Protocol*) atau disebut *web browser* berupa konten data, yang biasanya terdiri dari halaman *web* yang terdiri dari dokumen HTML bersama dengan objek terkait seperti gambar dll [17].

Apache Tomcat, Microsoft windows *Server 2003 Internet Information Services* (IIS), *Lighttpd*, *Sun Java System Web Server*, *Xitami Web Server*, dan *Zeus Web Server* merupakan aplikasi yang bertindak sebagai *web server*. Gambar 2.4 menunjukan arsitektur *request* dan *response web server* yang terdapat pada *web server* [18].



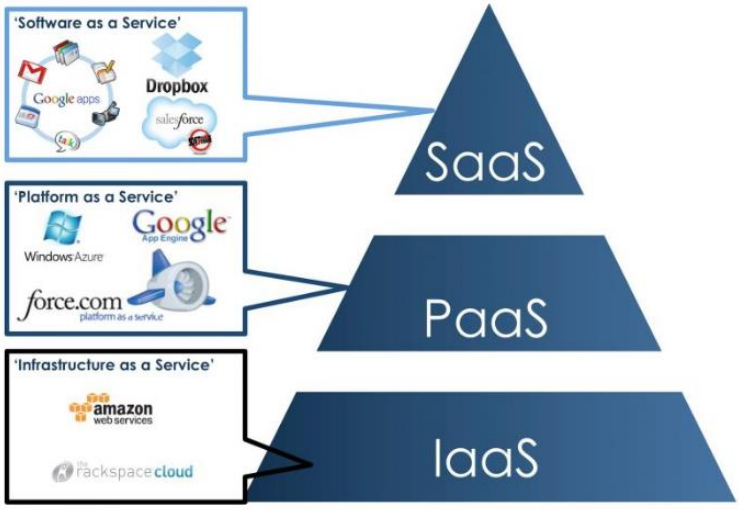
Gambar 2.4 Struktur permintaan dan *respon* pada *web server*

2.2.8 *Apache*

Apache adalah *server web* yang menangani permintaan dan tanggapan HTTP dengan memberikan informasi terperinci. *Apache* ini adalah *server* modular yang ringkas dengan standar protokol HTTP dan sangat populer [19]. *Apache* digunakan sebagai *web server* karena menyediakan *web server* yang aman, efisien, dan dapat dikembangkan serta dikonfigurasi dengan mudah *Apache* mempunyai tugas utama yaitu membuat situs *web* yang dapat diakses oleh pengguna yang terdiri dari kode-kode PHP yang dibuat oleh pembuat situs *web* [20].

2.2.9 *Cloud Computing*

Cloud computing adalah sebuah sistem komputasi yang menggabungkan *processor*/*computing* sumber daya, penyimpanan, jaringan, dan perangkat lunak sebagai layanan melalui jaringan/internet untuk komputasi jarak jauh (*remote)* yang efisien [21]. *Cloud computing* secara sederhana hanyalah penyediaan yang melibatkan berbagai layanan komputasi seperti *server*, penyimpanan data, dan analisis melalui internet (*cloud*). *Cloud computing* adalah kombinasi teknologi pengembangan dan komputasi berbasis internet yang menyediakan fasilitas berbagi *power* tanpa adanya perangkat tambahan, harga yang relatif rendah dan kapasitas data *unlimited* [22]. Gambar 2.5 berikut ini merupakan model layanan *Cloud Computing* :



Gambar 2.5 Layanan *Cloud Computing*

Model layanan *cloud computing* terdiri tiga jenis yaitu:

a) *Software as a Service* (SaaS)

Software yang diberikan untuk pengguna online oleh penyedia aplikasi yang berjalan di infrastruktur *cloud* dikenal sebagai SaaS. Aplikasi dalam perangkat *client* dapat diakses melalui antarmuka *web browser, email* berbasis *web*.

b) *Platform as a Service* (PaaS)

Cloud computing berbentuk platform yang digunakan pengguna untuk membuat aplikasi. Pengguna diberi kesempatan untuk menerapkan aplikasi khusus atau yang dikembangkan pada infrastruktur *cloud* menggunakan bahasa dan alat pemrograman yang didukung oleh penyedia.

c) *Infrastructure as a Service* (IaaS)

Iaas adalah model layanan *cloud* yang mengacu pada penyebaran, penyimpanan, jaringan, dan sumber daya yang memungkinkan pelanggan untuk menyebarkan dan memelihara perangkat lunak kustom, yang mungkin mencakup baik sistem operasi maupun aplikasi. Pada dasarnya IaaS merupakan bentuk *server* fisik dan virtual [22].

Terdapat empat model *cloud computing* yaitu:

a) *Private Cloud*

Infrastruktur *cloud* hanya digunakan oleh satu organisasi atau perusahaan yang menyewa dan dibuat untuk memenuhi kebutuhan organisasi tertentu. Privat *cloud* tidak dapat diakses secara umum karena hanya pihak internal perusahaan saja yang dapat mengakses.

b) *Public Cloud*

Public cloud merupakan layanan yang ditunjukan untuk masyarakat umum atau perusahaan besar yang disediakan *provider* melalui internet. *Public cloud* dimiliki oleh organisasi yang menjual layanan *cloud* dengan model pembayaran sesuai pemakaian. Walaupun dapat diakses dan digunakan secara umum, *public cloud* mempunyai keamanan yang tinggi

sehingga pengguna lain tidak dapat mencuri dan melihat data pengguna lain.

c) *Community Cloud*

Infrastruktur *cloud* merupakan bentuk *hybrid cloud* dan *private cloud* dimana beberapa oganisasi menggunakannya secara bersama dengan melayani dan mendukung komunitas tertentu. Infrastruktur ini biasanya dikelola oleh organisasi terkait atau pihak ketiga.

d) *Hybrid Cloud*

Hybrid cloud adalah teknologi *cloud* yang menggabungkan jenis *private* dan *public cloud* untuk dikelola tetapi dihubungkan oleh teknologi atau standar kepemilikan yang memungkinkan portabilitas data dan aplikasi [22].

2.2.10 *Response Time*

Response time adalah waktu yang dibutuhkan oleh salah satu *server* dalam menerima *respons* setelah *server* mengalami kegagalan sampai *server* cadangan mendapat *respon. Response time* mengacu pada seberapa cepat suatu sistem/aplikasi dapat merespon suatu aksi. *Response time* diambil dari waktu diantara sebelum terjadi proses *failover* sampai dengan proses *failover* itu terjadi [24]. *Response time* berfungsi untuk melihat waktu aktif pada *web server*.

2.2.11 QoS

Quality of Service bertujuan untuk mengurangi *delay* dan *jitter* mengacu pada teknologi apa pun yang mengelola lalu lintas data untuk mengurangi packet loss (kehilangan paket), latency, dan jitter pada jaringan. [25].

Teknik yang digunakan sebagai pengukuran kualitas jaringan dan upaya yang digunakan untuk menentukan definisi karakteristik dan layanan disebut *Quality of Service* (QoS). QoS digunakan untuk mengukur berbagai kinerja pekerjaan yang telah diklasifikasikan dan dihubungkan dengan layanan tertentu [26]. Terdapat beberapa parameter QoS yaitu *throughput, packet loss, delay,* dan *jitter*.

2.2.12 *Throughput*

Throughput adalah jumlah data yang berhasil terkirim dan mencapai tujuannya dalam jangka waktu tertentu dibagi dengan jangka waktu tersebut yang diukur dalam bps (bit per *second*) [27]. Tabel 2.2 menunjukan kategori versi *Telecommunication and internet protocol harmonization over network* (TIPHON) untuk melihat nilai *throughput.*

Tabel 2.2 Kategori *Throughput*

Kategori *Throughput*	*Throughput* (bps)	Indeks
Sangat bagus	1200 kbps-2.1 Mbps	4
Bagus	700-1200 kbps	3
Sedang	338-700 kbps	2
Buruk	0-338 kbps	1

(a)

Untuk menghitung nilai *throughput*, rumus perhitungan yang digunakan terdapat pada persamaan 2.2 [5].

$$\textit{Throughput} : \frac{\text{Jumlah paket data yang dikirim}}{\text{Waktu pengiriman paket}} \quad (2.2)$$

2.2.13 *Delay (Latecy)*

Delay (*latency)* adalah lamanya waktu yang diperlukan untuk menempuh jarak dari sumber ke tujuan dalam proses pengiriman data. Jarak, media fisik (kabel), kongesti (kenaikan beban) atau waktu pemrosesan yang lama merupakan pengaruh yang menyebabkan terjadinya *delay* [5]. Kategori *delay* terdapat pada Tabel 2.3 berikut:

Tabel 2.3 Kategori *Delay*

Kategori *Delay*	Besar *Delay* (ms)	Indeks
Sangat bagus	< 150ms	4
Bagus	150 hingga 300ms	3

Kategori *Delay*	Besar *Delay* (ms)	Indeks
Sedang	300 hingga 450ms	2
Buruk	> 450ms	1

(b)

Berikut rumus pada persamaan 2.3 yang digunakan sebagai perhitungan *delay* [5].

$$Delay : \frac{\text{Total delay}}{\text{Total paket yang diterima}} \quad (2.3)$$

2.2.14 *Packet Loss*

Packet Loss yaitu kondisi yang menampilkan hilangnya paket dimana jumlah total paket yang dikirimkan tidak seutuhnya sampai ke tujuan. *Collision* dan *congestion* pada jaringan merupakan penyebab hilangnya paket yang dikirimkan [27]. Tabel 2.4 menunjukkan kategori yang terdapat pada *packet loss*

Tabel 2.4 Kategori *Packet loss*

Kategori *Packet loss*	*Packet loss* (%)	Indeks
Buruk	>25 %	1
Sedang	15-24 %	2
Bagus	3-14 %	3
Sangat bagus	0-2%	4

Pada persamaan 2.4 merupakan rumus yang digunakan sebagai perhitungan *packet loss* [5]

$$Packet\ Loss : \frac{\text{paket data yang dikirim} - \text{paket data yang diterima}) \times 100\%}{\text{Total paket yang dikirim}} \quad (2.4)$$

2.2.15 *Jitter*

Jitter atau dikenal sebagai variasi waktu kedatangan paket yang disebabkan oleh panjang antrian, kemacetan jaringan, waktu pemrosesan data, dan waktu penumpukan ulang paket ketika paket telah sampai dititik akhir perjalanan.

Sekumpulan *delay* yang berkumpul menjadi satu selama proses pengiriman paket atau data dinamakan *Jitter* [5]. Variasi beban trafik dan banyaknya paket yang tertumpuk (*congestion*) yang ada di jaringan sangat mempengaruhi nilai *jitter* [29]. Tabel 2.5 menunjukkan indeks kategori yang terdapat pada *jitter*.

Tabel 2.5 Kategori *Jitter*

Kategori *Jitter*	*Jitter*(ms)	Indeks
Buruk	125 s/d 225 ms	1
Sedang	75 s/d 125 ms	2
Bagus	0 s/d 75 ms	3
Sangat bagus	0 ms	4

Rumus yang digunakan untuk menghitung *jitter* dapat dilihat pada persamaan 2.5 berikut [5].

$$Jitter : \frac{\text{Total variasi delay}}{\text{Total Paket diterima}-1} \quad (2.5)$$

2.2.16 CPU *Usage*

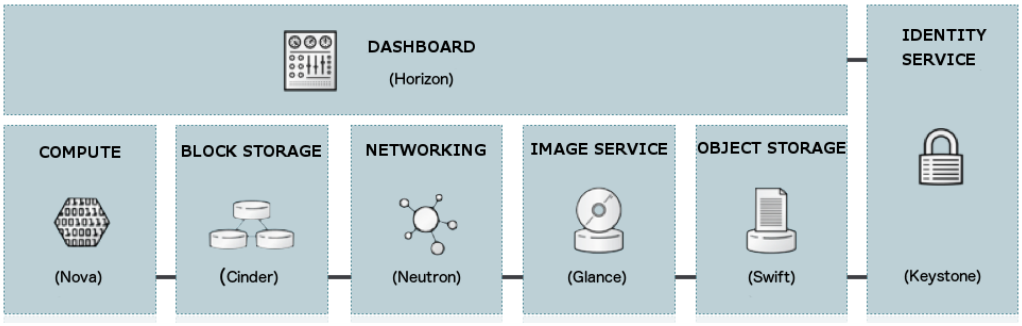
CPU Usage adalah sumber daya yang diperlukan dalam melaksanakan proses komputerisasi *server* pada saat program sedang berjalan. CPU bertanggung jawab untuk mengontrol dan mengolah seluruh instruksi yang diberikan kepada komputer. Peningkatan CPU sejalan dengan banyaknya jumlah permintaan yang dikirimkan. Semakin banyak permintaan yang dikirimkan maka secara otomatis kinerja CPU akan meningkat [30]. Kecepatan prosesor (CPU) diukur dalam satuan megahertz (MHz) - atau jutaan instruksi per detik - dan gigahertz (GHz), atau milyaran instruksi per detik. Semakin besar kecepatan prosesor, maka semakin cepat prosesor tersebut mengeksekusi perintah / instruksi. Hasil penggunaan CPU ditulis dalam bentuk persen (%).

2.2.17 *Virtualbox*

Oracle VM VirtualBox yang sekarang sedang dikembangkan oleh *Oracle* merupakan jenis perangkat lunak atau biasa dikenal dengan *VirtualBox*. Perusahaan Jerman, Innotek GmbH adalah pencipta awal yang mengembangkan *VirtualBox*. Pada Tahun 2008 bulan Februari Innotek GmbH digantikan oleh *Sun Microsystems* dan *Sun Microsystem* kemudian juga digantikan oleh *Oracle*. Sistem operasi secara sederhana dapat divirtualisasikan dengan *VirtualBox.* Pemanfaatan *VirtualBox* ditunjukkan untuk *desktop*, *server* dan penggunaan *embedded* (sistem tertanam seperti *router*, *firewall*). *Virtualbox* adalah jenis *hypervisor type* 2 berdasarkan tipe VMM yang ada saat ini [31].

2.2.18 *Openstack*

Salah satu jenis layanan pemrosesan *cloud* berbasis *Infrastructure as a Service* (IaaS) adalah *Openstack*. *Openstack* merupakan suatu proyek *open source* yang dapat mengelola perangkat komputasi menggunakan teknik virtualisasi dan baremetal [32]. *Openstack* mengontrol sumber daya jaringan dan proses kompuer di pusat data menggunakan *dashboard* yang menyediakan kontrol manajemen dengan memberi pengguna akses melalui antarmuka *web* (*web Interfaces*) [33]. Arsitektur *openstack* terdapat pada Gambar 2.6 sebagai berikut:



Gambar 2.6 Arsitektur *Openstack*

1. Horizon (*Dashboard*) berfungsi untuk menyediakan antarmuka berbasis *web* untuk administrator.
2. Keystone (*Identity*) berfungsi untuk menyediakan layanan autentikasi dan otorisasi di seluruh infrastruktur *cloud*.

3. Neutron (*Networking*) berfungsi untuk menyediakan berbagai layanan jaringan seperti alamat IP, DNS, DHCP, dan kelompok keamanan seperti *firewall*.
4. Cinder (*Block Storage*) berfungsi untuk menyediaan penyimpanan untuk komputasi *instance*.
5. Nova (*Compute*) berfungsi untuk mendukung pengelolaan *instance* mesin virtual melalui lapisan abstaksi yang berinteraksi dengan hypervisor.
6. Glance (*Image*) berfungsi untuk menyediakan layanan pengelolaan *disk-image*.
7. Swift (*Object Storage*) berfungsi untuk menyimpan dan mengambil data secara acak di *cloud* [34].

2.2.19 *Ubuntu*

Ubuntu adalah versi *system* operasi *Linux* berbasis Debian yang tersedia sebagai *software* bebas dan didukung oleh komunitas dan pakar profesional. *Ubuntu*, yang berarti "Kemanusiaan kepada sesama", berasal dari filosofi Afrika Selatan.”. Meskipun *Ubuntu* dibuat khusus untuk pengguna pribadi dan ada juga versi *server Ubuntu* yang banyak digunakan secara luas [35].

2.2.20 HTTPERF

David dari HP Labs membuat *HTTPerf* yang digunakan untuk mengetahui seberapa baik kinerja *web server.* Alat ini mendukung protokol HTTP baik HTTP/1.0 dan HTTP/1.l. Aplikasi ini bekerja dengan menghasilkan beban *web server*, kemudian merangkum dan mengeluarkan statistik yang dicetak setelah uji coba. Pengujian sistem *web* terdiri dari beberapa *client, web server*, dan koneksi yang menghubungkan *client* menuju *server*. Selama pengujian *web server, client* membuat banyak permintaan ke *web server*, sehingga *throughput* yang merupakan jumlah tanggapan *web server per second* dapat dihitung. Mengirimkan permintaan ke *server* pada tingkat tertentu dan kemudian menghitung berapa lama permintaan itu datang lagi merupakan salah satu cara untuk melihat baik atau tidaknya kinerja *web server* [36].

2.2.21 *Apache Benchmark*

Apache Bench adalah alat yang dikembangkan oleh organisasi *Apache* yang digunakan untuk mengukur kinerja *web server* pada *Hypertext Transfer Protocol* (HTTP). Tool ini berfungsi untuk menghitung seberapa banyak permintaan per detik yang dapat ditangani oleh *web server* yang sedang digunakan. *Apache Bench* memiliki fitur seperti *opensource* yang tersedia secara gratis, menggunakan baris perintah sederhana, *platform independent* atau tidak bergantung pada platform tertentu artinya dapat digunakan secara merata di Gnu / Linux atau di *server* Windows, untuk menguji beban dan kinerja dari *web server*, dan tidak dapat diperluas (tidak dapat ditambahkan fitur tambahan). *Apache Bench* adalah alat yang berguna untuk menguji kinerja *web server* dengan berbagai batasan pengujian, termasuk tingkat transfer (*transfer rate*) dan jumlah permintaan per detik (*request per second*) [37].