BAB 2
DASAR TEORI

2.1 KAJIAN PUSTAKA

Penelitian Febby Triadi Adidrajat dan Astriana Mulyani [9] pada tahun 2019, menerapkan teknik *load balancing web server* yang berbasis *cloud* dengan menggunakan algoritma *round robin*. Tujuan dari penelitian ini adalah untuk merancang suatu *web server load balancer* pada Sampoerna University, sehingga dapat meminimalisir terjadinya *down time* pada *web server* ketika *website* diakses oleh banyak pengguna. Metode yang dilakukan pada penelitian ini dengan mengumpulkan data, seperti observasi, wawancara, studi pustaka dan melakukan analisis penelitian. Pengujian pada penelitian ini dilakukan dengan *benchmarking* memberikan beban *request* sebanyak 1000 *request* dan dengan *concurrency* sebanyak 100 *request* pada *web server* Sampoerna University. Hasil dari penelitian ini didapatkan bahwa *server* Sampoerna University mampu menangani 0,75 pengguna dalam satu detik dengan kata lain, satu pengguna dapat dilayani selama 1,3 detik. Dari hasil pengujian tersebut dengan menerapkan sistem *load balancer*, terbukti mampu menampung *request user* lebih banyak dibandingkan menggunakan *server* tunggal.

Penelitian Molion Surya Pradana dan Aditya Prapancar [10] pada tahun 2019, menganalisis performa *load balancing* menggunakan algoritma *weighted round robin* di infrastruktur BPBD. Tujuan dari penelitian ini untuk mengatasi permasalahan pada *web server* yang mengalami *overload*, karena jumlah pengunjung BPBD Jatim yang terus meningkat setiap tahunnya. Pada penelitian ini didapatkan hasil penggunaan CPU pada *load balancing* LVS *direct routing* dengan algoritma *weighted round robin* 2 banding 3 lebih stabil. Selain itu besar nilai *throughput* yang didapatkan saat menggunakan algortima *weighted round robin* 2 banding 3 lebih tinggi dan nilai *request error* yang dihasilkan lebih kecil dibandingkan pada skenario pengujian lainnya. Hal ini menjadikan penggunakan *load balancing* dengan algoritma *weighted round robin* 2 banding 3 pilihan terbaik ketika ingin mengatasi *request error*.

Penelitian Surahmat dan Alfred Tenggono [11] pada tahun 2019, menganalisis perbandingan kinerja layanan *infrastructure as a service cloud computing* pada proxmox dan xenserver. Tujuan dari penelitian ini untuk mengetahui kelebihan dan kekurangan dari kedua jenis *software* dengan teknologi *hypervisor* tipe 1 yaitu proxmox dan xenserver. Metode yang digunakan pada penelitian ini menggunakan *action research* atau penelitian terapan. Hasil dari penelitian ini didapatkan kedua *hypervisor* tipe 1 ini mampu menjalankan sistem virtualisasi dengan baik. Penggunaan CPU pada proxmox lebih besar dibandingkan penggunaan CPU pada xenserver, akan tetapi penggunaan memori pada proxmox lebih kecil dibandingkan xenserver. Untuk pengujian *network traffic* antara kedua *hypervisor* tipe 1 ini memiliki nilai pengujian yang hampir sama dengan penilaian QoS sangat baik dengan nilai untuk xenserver pada *throughput* 2.38 Mb/s, delay 0.428 ms dan *packet loss* sebesar 0.06%. Sedangkan untuk proxmox pada *throughput* sebesar 2.45 Mb/s, *delay* 0.459 ms dan *packet loss* 0.04%. Untuk hasil *response* penggunaan xenserver dan proxmox mendapatkan hasil respon yang sama - sama baik dengan total nilai perolehan untuk proxmox dari responden dengan nilai total 3848 dan xenserver sebesar 3739.

Penelitian Alam Rahmatulloh dan Firmansyah MSN [12] pada tahun 2017, implementasi *load balancing web server* menggunakan haproxy dan sinkronisasi *file* pada sistem informasi akademik Universitas Siliwangi. Tujuan dari penelitian ini untuk mengatasi *overload* pada *server* tunggal dikarenakan banyaknya jumlah *request user* yang datang secara bersamaan seperti pada pengisian kartu rencana studi mahasiswa. Pada penelitian ini menggunakan perangkat fisik untuk membangun *load balancing web server*. Pengujian *load balancing* dilakukan sebanyak 4 kali dengan melihat hasil pembagian oleh *load balancer* ke masing – masing *web server*. Hasil penelitian yang didapatkan adalah *load balancing* dengan algortima *round robin* dapat bekerja dengan baik ketika *request* yang datang dari client berhasil didistribusikan oleh *load balancer* secara merata ke masing – masing *web server*. Sehingga dengan hasil pengujian ini, kemampuan *web server* dapat melayani 10000 *request* tanpa mengalami *error request*, selain itu penerapan sinkronisasi *file* bekerja dengan baik. Dimana *file* yang di-*uplaoad* pada *web server* 1 berhasil disinkronkan dengan *web server* 2 dan 3 begitupun sebaliknya.

Penelitian Saidi Ramadan Siregar [8] pada tahun 2020, efisiensi fisik komputer *server* dengan menerapkan proxmox *virtual environment*. Tujuan dari penelitian ini untuk meminimalisir penggunaan *server* fisik untuk menerapkan sebuah sistem yang dibangun. Penelitian ini dilakukan dengan metode yang digunakan adalah *Network Development Life Cycle* (NDLC). Hasil dari penelitian teknologi virtualisasi dengan menerapkan proxmox sebagai *hypervisor type* 1 dapat mengefisiensikan unit fisik komputer *server*. Manajemen penyimpanan data menggunakan proxmox lebih lengkap, karena bisa dilakukan *backup virtual machine* serta monitoring pada masing – masing *virtual machine* lebih mudah.

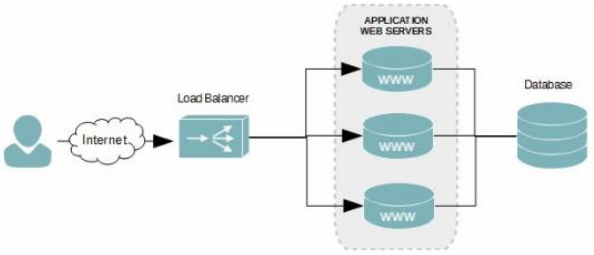
Tabel 2.1 Rangkuman Keterkaitan dengan Penelitian Sebelumnya

Penelitian Oleh	**Algoritma *Load Balancing***		**Tempat Implementasi**			**Parameter QoS**			
	Round Robin	*Weighted Round Robin*	*Server* Fisik	Server Fisik dan Virtualisasi	*Cloud*	*Throughput*	*Delay*	CPU *Ussage*	*Packet Loss*
Febby Triadi Adidrajat dan Astriana Mulyani	✓				✓		✓		
Molion S. P dan Aditya Prapancar		✓	✓			✓		✓	✓
Surahmat dan Alfred Tenggono				✓		✓	✓	✓	✓
Alam Rahmatulloh dan Firmansyah MSN	✓		✓						✓
Saidi Ramadan Siregar				✓				✓	
Vassa Metayasha		✓		✓		✓	✓	✓	✓

2.2 DASAR TEORI

2.2.1 *LOAD BALANCING*

Load balancing bekerja dengan cara mendistribusikan beban trafik yang datang secara merata ke beberapa *server – server* yang terhubung pada mesin *load balancer*, sehingga beban trafik yang diterima akan lebih ringan untuk dilayani. Dengan menggunakan teknik *load balancing* dapat mempersingkat waktu akses terhadap *web server* dan memiliki ketersediaan layanan yang tinggi. Teknik *load balancing* sering kali diperlukan saat membuat solusi untuk menangani permintaan *user* dalam jumlah besar atau yang memiliki tuntutan tinggi pada keamanan dan redudansi [13].



Gambar 2.1 Server dengan teknik *load balancing* [12].

2.2.2 ALGORITMA *WEIGHTED ROUND ROBIN*

Penjadwalan *Weighted Round Robin* (WRR) dirancang untuk mampu menangani *server* dengan ketersediaan sumber daya perangkat yang berbeda - beda. WRR mampu mengatur jumlah bobot pada masing – masing *server*, sehingga WRR mampu menyeimbangkan kedua *server* yang memiliki spesifikasi berbeda. *Server* dengan jumlah bobot yang lebih tinggi akan menerima koneksi baru terlebih dahulu dibandingkan dengan *server* yang diberikan bobot lebih kecil dan *server* dengan spesifikasi yang lebih besar akan menerima jumlah koneksi lebih banyak dibandingkan dengan *server* yang memiliki spesifikasi lebih kecil [7].

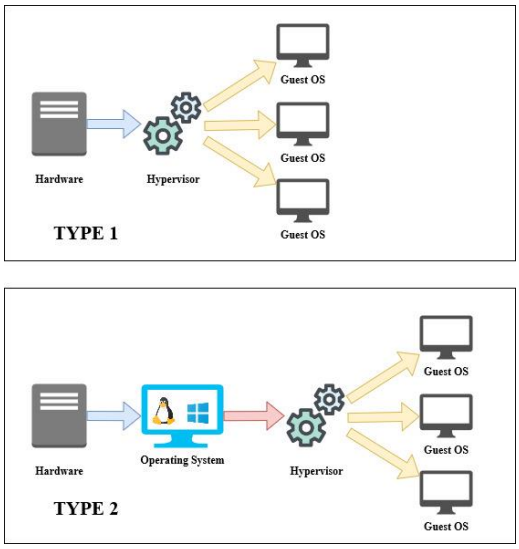
2.2.3 TEKNOLOGI VIRTUALISASI

Teknologi virtualisasi merupakan sebuah teknologi yang memungkinkan sebuah mesin yang berbentuk fisik dijadikan sebuah sumber daya bersama yang dapat dibagi dan digunakan oleh beberapa layanan sekaligus. Dengan kata lain, virtualisasi adalan proses mengubah perangkat fisik (*hardware*) menjadi perangkat lunak (*software*). Pada masing – masing perangkat *virtual* juga dapat dikonfigurasi

tanpa mempengaruhi satu sama lain walaupun dalam satu lingkungan virtualisasi. Masing – masing perangkat *virtual* juga dapat memiliki sistem operasi sendiri – sendiri, sehingga konfigrasi dari masing – masing perangkat pun tidak saling mempengaruhi. Selain itu, masing – masing mesin *virtual* pada lingkungan virtualisasi dapat dimatikan atau dihidupkan sesuai dengan kebutuhan tanpa harus mengganggu layanan lainnya. Sehingga ketersediaan sebuah layanan dapat lebih terjamin meskipun terdapat beberapa layanan yang sedang mengalami masalah atau perbaikan [14].

2.2.4 *HYPERVISOR*

Hypervisor adalah sebuah *firmware* yang bertugas membuat, mengatur, menjalankan, dan memonitor sebuah mesin *virtual*. *Hypervisor* diklasifikasikan menjadi 2 jenis tergantung tempat dimana dia berdiri sendiri yang ditunjukkan pada gambar 2.2.



Gambar 2.2 Arsitektur *Hypervisor* [14].

Pada *hypervisor* sendiri memiliki beberapa tipe yang pertama, *native / baremetal hypervisor*. *Hypervisor* ini dapat mengakses langsung perangkat keras pada suatu mesin fisik tanpa melalui sebuah sistem operasi. Contoh dari hypervisor pada tipe ini adalah Proxmox VE, VMware ESX/ESXi, Microsoft Hyper-V 2008/2012. Selanjutnya *hypervisor type* 2, *hypervisor* ini hanya dapat mengakses perangkat keras pada suatu mesin fisik melalui sebuah sistem operasi yang ada dibawahnya. Contoh dari *hypervisor type* 2 ini adalah Virtual Box, VMware Workstation, VMware Player [14].

2.2.5 *CLOUD COMPUTING*

Cloud computing merupakan komputasi dimana kapabilitas terkait teknologi informasi dapat disajikan dalam bentuk layanan (*as a service*), sehingga pengguna dapat mengaksesnya melalui jaringan internet tanpa mengetahui apa yang ada didalamnya beserta kendali terhadap infrastruktur teknologi yang membantunya. *Cloud computing* mengacu pada paradigma berorientasi layanan dimana penyedia layanan menawarkan komputasi awan mengacu pada paradigma berorientasi layanan. Dimana penyedia layanan menawarkan komputasi sumber daya seperti perangkat keras, perangkat lunak, penyimpanan dan *platform* sebagai layanan sesuai dengan permintaan pengguna [15].

2.2.6 PROXMOX VE

Proxmox VE (*Virtual Environtment*) adalah distro linux *server* dengan kode sumber terbuka untuk membangun lingkungan virtualisasi berbasis distribusi linux Debian. Proxmox VE digunakan untuk mengimplementasikan dan mengelola mesin *virtual*. Proxmox VE menggunakan kernel dari *Red Hat Enterprise Linux* (RHEL) yang sudah dimodifikasi. Ada dua teknologi *hypervisor* yang didukung oleh Proxmox VE yaitu *container-based virtualization* dan *Kernel-based Virtual Machine* (KVM). Beberapa fitur utama yang dimiliki oleh Proxmox VE, diantaranya :

1.) *Open source* : Proxmox VE sepenuhnya *open source* atau bersifat gratis dibawah *General Public License*, version 3 (GNU AGL, v3) yang artinya dapat digunakan dengan bebas.
2.) *High Availability* : beberapa server yang di-*install* Proxmox VE dapat digabungkan menjadi satu *cluster*. Dalam *mode cluster*, ketika terjadi kegagalan pada salah satu *node* maka *node* lain yang berada pada satu *cluster* akan mem-*backup node* tersebut untuk meminimalisir gangguan layanan.
3.) *Flexible storage* : memiliki banyak pilihan penyimpanan yang tersedia termasuk penyimpanan lokal dan penyimpanan berbasis jaringan, seperti LVM, iSCSI, NFS, GFS, dan CEPH.

4.) *Live migration* : *Live migration* memungkinkan untuk memindahkan mesin *virtual* dari satu *node* proxmox ke *node* proxmox lainnya dengan waktu *downtime* yang sangat kecil atau tanpa adanya *downtime.*

5.) *Bridged networking* : Proxmox VE memungkinkan pengguna untuk membuat jaringan *private* antara mesin *virtual*. Selain itu, VLAN juga tersedia.

6.) OS *Template* : Proxmox VE memungkinkan pengguna untuk membangun *template* sistem operasi sendiri atau dapat meng-*upload file* ISO yang sudah dimiliki ke *node* proxmox.

7.) *Scheduled backup* : Tersedia antarmuka untuk pengguna yang digunakan untuk mengatur strategi *backup*. *File backup* dapat disimpan secara lokal atau penyimpanan lain yang sudah dikonfigurasi.

8.) *Command-line* (CLI) *tool* : Proxmox VE menyediakan cara manajemen lain untuk pengguna mengatur sumber daya mesin *virtual* dan lainnya [16].

2.2.7 HAPROXY

HAProxy menawarkan layanan *load balancing* ke layanan berbasis HTTP dan TCP, seperti layanan yang terhubung ke internet dan aplikasi berbasis *web*. Bergantung pada algoritma penjadwalan sebagai penyeimbang beban yang dipilih, sehingga HAProxy dapat melayani ribuan koneksi yang terjadi [17].

2.2.8 *WEB SERVER*

Web server adalah sebuah *software* yang memberikan layanan berbasis data dengan menggunakan protokol HTTP atau HTTPS dari *user* yang mengakses melalui *web browser*. *Request* data yang diberikan oleh *user* akan diproses pada *server* dan *server* akan mengirimkan kembali data ke *user* dalam bentuk halaman *web* dan pada umumnya berbetuk dokumen HTML [12].

2.2.9 *DATABASE SERVER*

Database server merupakan aplikasi komputer yang menyediakan layanan data ke komputer atau sebuah program komputer. *Database server* memiliki fungsi sebagai tempat penyimpanan sebuah data, baik berbentuk *file* dokumen maupun *file*

gambar. *Database server* juga dapat dikatakan sebagai sebuah komputer yang didedikasikan untuk menjalankan program *server database*. Beberapa aplikasi *database* yang sering digunakan adalah MySQL, MariaDB, Oracle [12].

2.2.10 APACHE2

Apache2 merupakan *web server* yang paling umum digunakan hingga saat ini. Apache2 telah digunakan sekitar 50% dari semua situs *web* yang ada. Selain itu Apache2 bersifat lintas *platform*, ringan, dan digunakan di perusahaan kecil maupun perusahaan besar. Apache2 juga bersifat gratis dan memiliki sumber yang terbuka [18].

2.2.11 MARIADB

MariaDB adalah implementasi *Relational Database Management System* (RDBMS) yang didistribukan dibawah lisensi GPL (*General Public License*) dan dapat didistribusikan sebagai perangkat lunak berbasis *open source* dibawah lisensi GPL. MariaDB dapat digunakan oleh banyak pengguna secara bersamaan dan dapat memproses lebih banyak SQL dalam satu waktu [19].

2.2.12 WIRESHARK

Wireshark adalah *tool* yang ditujukan untuk penganalisaan paket data jaringan. Wireshark juga dapat dikatakan sebagai *network packet analyzer* yang berfungsi untuk *capture* paket – paket jaringan sesuai dengan protokol yang diinginkan. Pada dasarnya wireshark akan menangkap paket dengan protokol TCP [20].

2.2.13 APACHE BENCHMARK

Apache Benchmark adalah salah satu *tools* atau *software* yang digunakan untuk mengukur *performance* sebuah *web server*. Dengan memberikan beban *request* dalam jumlah yang banyak dan melihat seberapa mampu *server* dapat melayani jumlah beban *request* yang diberikan melalui *tools* Apache Benchmark [9].

2.2.14 QUALITY OF SERVICE (QOS)

Quality of Service (QoS) adalah kemampuan sebuah jaringan untuk dapat menyediakan layanan yang baik dengan memperhatikan parameter *throughput, delay, jitter, packet loss, latency*, MOS, *echo cancellation* dan PDD. QoS sangat ditentukan oleh kualitas dari sebuah jaringan yang digunakan. Adanya beberapa factor yang dapat mempengaruhi kualitas dari nilai QoS, seperti redaman, distorsi dan *noise* [21].

2.2.14.1 *THROUGHPUT*

Throughput adalah nilai kecepatan dari *transfer* data efektif yang diukur dalam satuan bps. *Throughput* merupakan jumlah total kedatangan paket yang berhasil diterima dengan *interval* waktu tertentu dan dibagi oleh durasi *interval* waktu tersebut [21].

$$Throughput = \frac{\text{Jumlah data yang dikirim}}{\text{Waktu pengiriman data}} \quad (2.1)$$

2.2.14.2 *DELAY*

Delay adalah total waktu yang dilalui oleh suatu paket dari pengirim ke penerima pada sebuah jaringan. *Delay* antar *user* pengirim ke penerima pada dasarnya tersusun atas *hardware latency*, *delay* transmisi dan *delay* akses [21].

$$Delay \text{ rata – rata} = \frac{\text{Total } Delay}{\text{Total paket yang diterima}} \quad (2.2)$$

Pengelompokkan standarisasi *delay* berdasarkan standar TIPHON ditunjukkan pada tabel 2.1.

Tabel 2.2 Standarisasi *Delay* [22]**.**

Nilai	Besar *delay* (ms)	Indeks
4	< 150	Sangat Bagus
3	<250	Bagus
2	<350	Kurang Bagus
1	<450	Jelek

2.2.14.3 *PACKET LOSS*

Packet loss adalah parameter yang menggambarkan sebuah kondisi yang menunjukkan total paket yang hilang pada *transfer* data yang terjadi. *Packet loss* terjadi karena *collision* dan *congestion* pada jaringan [23].

$$\textit{Packet loss} = \frac{\text{paket data yang dikirim} - \text{paket data yang diterima}}{\text{paket data yang dikirim}} \text{ x } 100\% \quad (2.3)$$

Pengelompokkan standarisasi *packet loss* berdasarkan standar TIPHON ditunjukkan pada tabel 2.3.

Tabel 2.3 Kriteria *Packet Loss* [22]**.**

Nilai	Nilai *packet loss* (%)	Indeks
4	0	Sangat Bagus
3	3	Bagus
2	15	Kurang Bagus
1	25	Jelek

2.2.14.4 *RESOURCE UTILIZATION*

Resource utilization berfungsi untuk menunjukkan tingkat pengunaan sumber daya perangkat tertentu, seperti penggunaan CPU dan *memory*. Tujuannya adalah melihat seberapa sibuk perangkat yang dimiliki dalam menjalankan sistem yang sedang berjalan diatasnya [24].