BAB II
DASAR TEORI

2.1 KAJIAN PUSTAKA

Pada penelitian ini dalam menguji *webserver Openlitespeed* dan *nginx* pada *Software defined network* (SDN) penulis akan mengungkapkan sedikit refrensi kajian Pustaka. Kajian Pustaka yang relevan dijaman sekarang akan digunakan untuk menjadi bahan acuan untuk penulis dalam Menyusun dan melakukan penelitian ini.

Penelitian yang dilakukan oleh Ronaldi Putra salah satu mahasiswa universitas islam riau pekanbaru yang berjudul "Analisis kinerja perbandingan *webserver nginx* dan *Openlitespeed*" . membahas tentang pengujian pada kedua *server* ini ketika diberi beban dan parameter parameter yang diuji adalah throughtput antara kedua *server* ini, *connection time*, perbandingan *Request*, pengujian *reply section*, kemudian *error section* pada kedua *server* ini, *system log* yang nantinya akan menampilkan informasi yang lengkap, dan *security* dari kedua *server* ini. dari masing masing parameter yang diuji ada baik dan buruknya ada plus minusnya seperti *connection time* pada kedua *server* ini hasilnya *Openlitespeed* lebih unggul dalam mengkoneksikan *client* ke *server* dan *server* ke *client*.[3]

Penelitian yang dilkukan oleh mahasiswa Andri Jiwandono salah satu mahasiswa universitas islam riau pekanbaru yang berjudul " Analisa perbandingan kinerja *webserver apache*, *nginx* dan *Openlitespeed* dengan menggunakan metode *stress test* " . membahas tentang pengujian pada ketiga *server* ini ketika diberi beban dan parameter parameter yang diuji adalah, *connection*, *sent*, *received*, *error*, *log*, dan *web security*. dari masing masing parameter yang diuji ada plus minusnya ada baik dan ada tidak baiknya namun dari ketiga *webserver* yang diuji ini bisa dibilang dimasing masing bidang ada keunggulannya tersendiri seperti pada pengujian *web security* hasilnya ketiga *webserver* ini sudah mengenkripsi semua paket yang dikirim yang pada akhirnya data menjadi terlindungi dan aman dari *hacker*.[4]

Penelitian yang dilakukan mahasiswa Busran Dan Ridwan dari institut teknologi padang dengan penelitian yang berjudul "Analisis perbandingan performa *Apache webserver* dan *nginx* menggunakan jmeter". Membahas tentang menguji kinerja kedua *webserver* tersebut dalam menangani *Request* yang diminta oleh *client* dari mulai 100, 1.000 sampai 10.000 *user*. Dengan parameter yang diuji *time* dan *Throughput* bisa disimpulkan bahwa *nginx* lebih efisien

dalam waktu dan dalam melakukan respon terhadap *client nginx* lebih unggul daripada *Apache*.[5]

Penelitian yang dilakukan oleh Riswandi dari politeknik negeri ujung pandang yang berjudul "Evaluasi kinerja *webserver Apache* menggunakan protocol HTTP2". Penelitian tersebut membahas tentang menguji kinerja *webserver* berdasarkan protocol *respon time.* Ketika menangani http/1 dan http/2 penelitian ini menggunakan aplikasi jmeter dengan hasil kedua *webserver respon time* http2 tersebut akan lebih cepat jika diterapkan di *nginx*.[6]

Penelitian yang dilakukan oleh Albert salah satu mahasiswa mercu buana Yogyakarta dengan penelitian yang berjudul "Analisis performansi antara *Apache* dan *nginx webserver* dalam menangani *client Request*" . penelitian tersebut membahas tentang menguji kinerja *webserver* Ketika menangani *client Request* yang nantinya akan diberikan beban dari 100 *user* hingga 1.000.000 *user* dengan menggunakan *Apache bench*. Kemudian hasil daripada pengujian tersebut nantinya dapat memberikan sebuah pilihan manakah yang optimal dalam menangani banyaknya permintaan *client*.[7]

Penelitian yang dilakukan oleh Evin Nofia delta dan Asmunin "*PERFORMANCE TEST* DAN *STRESS WEBSITE* MENGGUNAKAN OPEN *SOURCE TOOLS*" penelitan ini membahas mengenai *Performance test website* if.unesa.ac.id dengan menggunakan metode stress *test*. Dari hasil pengujian *Performance test* dan analisa system diperoleh hasil bahwa *website* yang paling baik adalah *website* unesa.ac.id kedua *website* if.unesa.ac.id ketiga *website* detik.com. Ketika melakukan *Performance test* rata-rata persentase error *website* unesa.ac.id sebesar 23.90 %, *website* if.unesa.ac.id sebesar 35.60% dan *website* detik.com sebesar 69.58%. Namun untuk pengujian stress tes tidak ada respon dari *website* unesa.ac.id. Mungkin dikarenakan masalah yang disebabkan oleh database/kesalahan penggunaan pada mutex dapat mengakibatkan deadlock, rata-rata persentase error *website* unesa.ac.id sebesar 100 %, *website* if.unesa.ac.id sebesar 53.92% dan *website* detik.com sebesar 52.38%. *Website* unesa.ac.id diharapkan dapat menyediakan layanan *web* dengan sistem keamanan yang baik, karena dari scan *vulnerability* dengan *acunetix* terdapat kerentanan blind SQL *Injection* dan *Cross site scripting*

Ringkasan informasi dari hasil penelitian yang telah diuraikan di atas dapat ditunjukkan pada Tabel 2.1

Tabel 2.1 Penelitian Sebelumya

Diteliti Oleh	*Objek*	Skeneario Pengujian	Parameter Penelitian	Jumlah Koneksi	Metode Jaringan	Hasil
Ronaldi Putra	*Nginx* Dan *Openlitespeed*	Penambahan Beban *Request*	*Respon time*, *ram Usage*, *cpu Usage*, *sent*	700, 900,1.100	Konvensional	*Openlitespeed* unggul dalam koneksi *client*
Andri Jiwandono	*Apache*, *Nginx*, Litspeed	Penambahan Beban *Request*	*Throughput*, *cpu Usage*, *ram Usage*, *connection*, *sent*, *received*, *error*, *log*, *web security*	700, 900,1.100 *user*	Konvensional	Pada pengujian *web security* hasilnya ketiga *webserver* ini sudah mengenkripsi semua paket yang dikirim yang pada akhirnya data menjadi terlindungi dan aman dari *hacker*
Busran, Ahmad Ridwan	*Apache*, *Nginx*	Penambahan Beban *Request*	*Respone time*, *Throughput*	100,1.000 ,10.000	Konvensional	Dengan parameter yang diuji *time* dan *Throughput* bisa disimpulkan bahwa *nginx* lebih efisien dalam waktu dan dalam melakukan respon terhadap *client nginx* lebih unggul daripada *Apache*.
Riswandi,Kasim, Fajri Raharjo	*Apache*	Penambahan Beban *Request*	*Respone time*,latensi	1 – 500 *user*	Konvensional	Ketika menangani http/1 dan http/2 *webserver respon time* http2 tersebut akan lebih cepat jika diterapkan di *nginx*
Albert yokobus chandra	*Apache* dan *nginx*	Penambahan Beban *Request*	*Connection time*	100, 1000, 10000,10000	Konvensional	Hasil *benchmarking* yang menunjukkan bahwa dari sisi penggunaan waktu, *nginx* menggunakan waktu lebih sedikit daripada *Apache* dalam menyelesaikan *client request.*

Evin Nofia delta Asmunin	*Website* if.unesa.ac.id	Pengujian presentase	*Performance test, stress website dan scan vulnerability*	5, 10, 50, 100, 150	Konvesional	Dari hasil scan vulnerability dengan acunetix, tiap *website* memiliki kerentanan yang berbeda-beda. Untuk Vulnerability High Severity *website* unesa.ac.id memiliki kerentanan berbahaya karena terdapat kerentanan blind SQL Injection, dimana kerentanan tersebut berpengaruh pada
Diteliti Oleh	***Objek***	**Skeneario Pengujian**	**Parameter Penelitian**	**Jumlah Koneksi**	**Metode Jaringan**	**Hasil**
						integritas dari database *website* atau mengungkapkan informasi sensitif.

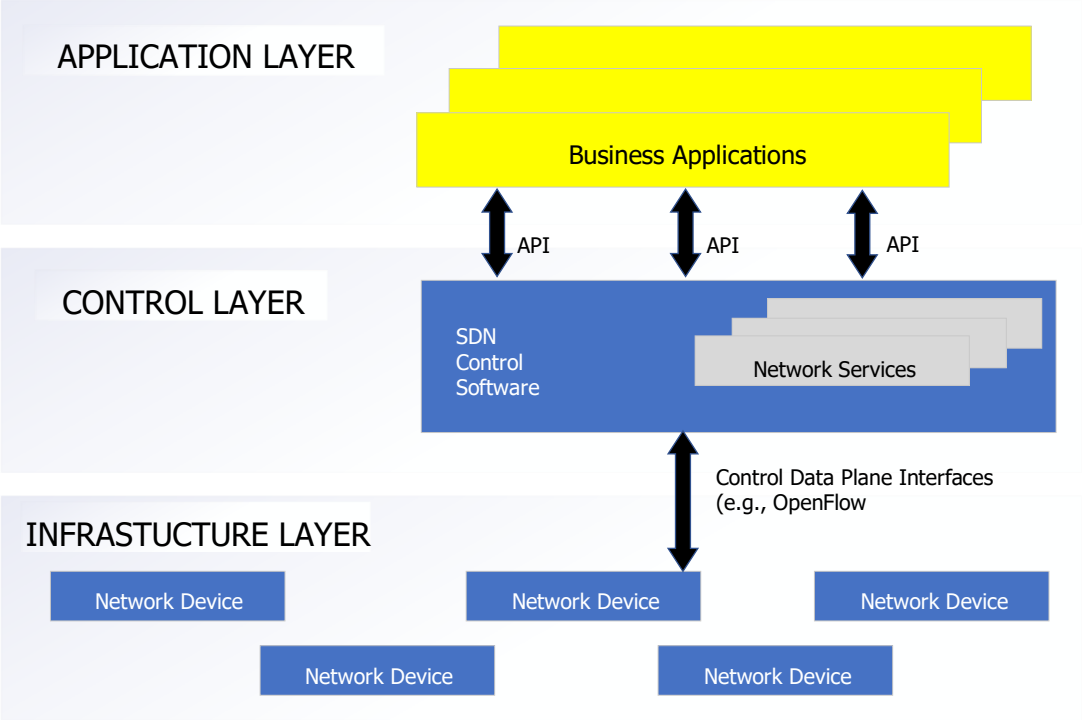
.

2.2 DASAR TEORI

2.2.1 *SOFTWARE DEFINED NETWORK*

Software-Defined Networking (SDN) adalah pendekatan yang mengubah cara jaringan komputer tradisional diatur, dikendalikan, dan dikelola. Cara kerja SDN melibatkan pemisahan antara lapisan pengendali (*control plane*) dan lapisan data (*data plane*) dalam jaringan. Perbedaan SDN dengan paradigma lainnya dalam jaringan komputer, seperti model tradisional berbasis perangkat keras, adalah pada Sentralisasi dan Desentralisasi. SDN mendorong sentralisasi pengendalian jaringan, sedangkan model tradisional memiliki pengambilan keputusan yang tersebar di perangkat keras. SDN menggunakan kontroler sentral untuk mengendalikan seluruh jaringan, sementara model tradisional memiliki pengambilan keputusan di setiap perangkat. [8]. *Software-Defined Networking* (SDN) dianggap sebagai kunci pengembangan teknologi masa depan. Hal ini disebabkan adanya perbedaan arsitektur SDN dengan ar*site*ktur jaringan sebelumnya. Perkembangan SDN ini karena kebutuhan untuk bergerak bebas dan memfasilitasi perluasan jaringan komputer, yang tidak dapat dicapai dengan teknologi jaringan "tradisional"; ini dicapai dengan membuat manajemen jaringan lebih mudah. Di seluruh dunia, penelitian sedang dilakukan untuk pengembangan jaringan di masa depan. Namun, berdasarkan beberapa penelitian, semakin banyak peneliti percaya bahwa pengembangan jaringan di masa depan akan terhambat oleh keterbatasan jaringan saat ini. Ini mengarah pada perubahan ar*site*ktur dalam teknologi masa depan. Berdasarkan Gambar 2.1, konsep *Software defined network* (SDN) menggunakan *application programming Interface* (API) sebagai penghubung antara *application layer* dan *control layer*. Dalam hal ini diperlukan bahasa pemrograman khusus untuk mendefinisikan *controller* yang akan dihubungkan dengan infrastruktur jaringan yang akan dibuat. Di sisi lain, lapisan kontrol dan lapisan infrastruktur dihubungkan oleh protokol *OpenFlow*. SDN juga memungkinkan pengguna untuk mengembangkan aplikasi manajemen jaringan. Bidang kontrol mengontrol jaringan sedangkan bidang data bertanggung jawab untuk mengirim paket

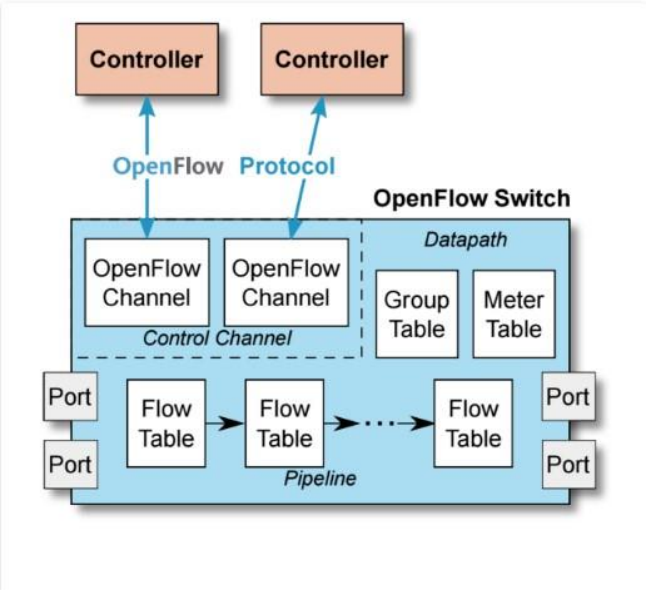
dan menentukan respons dari paket yang diterima atau diteruskan. Tujuan utama SDN adalah untuk memungkinkan kontrol infrastruktur jaringan yang lebih efisien [9].



Gambar 2. 1 Arsitektur SDN

2.2.2 *OPENFLOW*

OpenFlow adalah protokol komunikasi terbuka standar antara data plane dan *control plane* di SDN. Tidak seperti jaringan tradisional di mana fungsi bidang data dan bidang kontrol berada di sakelar/*router* yang sama, jaringan yang dipasang oleh *OpenFlow* memungkinkan fungsi bidang data dan bidang kontrol berada di perangkat yang berbeda, memungkinkan perluasan atau penambahan fungsionalitas ke jaringan yang ringan. Sakelar *OpenFlow* adalah perangkat khusus data yang hanya melakukan operasi penerusan paket berdasarkan kriteria yang digunakan untuk memilih paket dan serangkaian tindakan atau operasi yang akan dilakukan pada paket. Kesatuan seperangkat kriteria dan tindakan mereka disebut aliran. Meskipun perangkat *control plane OpenFlow* dapat menjadi *server* yang menyediakan logika dan kecerdasan jaringan atau yang disebut *controller* atau sistem operasi jaringan [10].



Gambar 2. 2 Cara kerja *OpenFlow*

Sakelar *OpenFlow* hanya dapat berfungsi dengan kerja kolaborasi dari tiga elemen penting: tabel aliran yang diinstal pada sakelar, pengontrol, dan protokol *OpenFlow* berpemilik agar pengontrol dapat berbicara secara aman dengan sakelar. Tabel aliran diatur pada sakelar. Pengontrol berbicara dengan sakelar melalui protokol *OpenFlow* dan menerapkan kebijakan pada aliran. Pengontrol dapat mengatur jalur melalui jaringan yang dioptimalkan untuk karakteristik tertentu, seperti kecepatan, jumlah lompatan paling sedikit, atau pengurangan latensi.

2.2.3 ONOS

Open Network Operating System (ONOS) merupakan sistem operasi SDN sumber terbuka. Tujuan utama pembuatan ONOS adalah untuk memberikan ketersediaan tinggi, skalabilitas, dan performa yang dibutuhkan pada jaringan. Selain itu, ONOS telah mengembangkan lapisan abstraksi dan antarmuka Northbound agar pengembangan aplikasi menjadi lebih mudah, serta lapisan abstraksi dan antarmuka *Southbound* yang memungkinkan kontrol *OpenFlow* pada perangkat keras. ONOS memiliki peran penting dalam arsitektur SDN dengan berada di lapisan pengendali (*control plane*) dari model SDN Dapat dilihat pada gambar 2.2.[11]

Gambar 2. 3 Logo ONOS *Open Network Operating System* Berikut adalah kelebihan kontroler ONOS:

A. Membawa fitur kelas *carrier provider* (skala, ketersediaan, dan performa) untuk pesawat *control* SDN
B. Memungkinkan untuk me-*monitoring*

2.2.4 *WEBSERVER*

Webserver adalah perangkat lunak yang membentuk tulang punggung world wide *web* (www). *Server web* menunggu permintaan dari klien menggunakan browser seperti *Netscape Navigator, Internet Explorer, Modzilla*, dan program browser lainnya. Jika ada permintaan dari browser maka *web server* akan memproses permintaan tersebut kemudian memberikan hasil proses berupa data yang diinginkan kembali ke browser. Data ini memiliki format standar yang disebut format SGML (standard general markup language). Data dalam bentuk format inilah yang kemudian akan ditampilkan oleh browser sesuai dengan kemampuan browser tersebut.[12]

2.2.5 *WEBSERVER NGINX*

Nginx adalah perangkat lunak sumber terbuka berkinerja tinggi sebagai *server* HTTP dan proxy terbalik. *Nginx* dengan cepat menyediakan statistik konten dengan penggunaan sumber daya sistem yang efisien. Itu dapat menyebarkan konten HTTP dinamis di jaringan menggunakan penangan FastCGI untuk skrip, dan dapat berfungsi sebagai perangkat lunak penyeimbang muatan yang berkemampuan tinggi. *Nginx* dibangun secara modular dan oleh karena itu mampu mendukung fitur seperti Load Balancing dan Reverse Proxying, Virtual host berbasis Nama dan IP, Fast CGI, akses cache langsung, SSL, Flash Video Streaming dan sejumlah fitur standar lainnya. *Nginx* dapat dieksekusi dan tersedia untuk Unix, Linux, varian BSD, MacOS X, Solaris, dan Microsoft Windows.[12]

2.2.6 *WEBSERVER OPENLITESPEED*

Openlitespeed Webserver (LSWS), adalah perangkat lunak *server web* berpemilik. Ini adalah *server web* terpopuler keempat, digunakan oleh sekitar 3,5% situs *web* pada Oktober 2018. LSWS dikembangkan oleh perusahaan swasta *Openlitespeed* Technologies. Perangkat lunak ini menggantikan *Apache*, yang artinya menggunakan format konfigurasi yang sama dengan *Apache* .[13] *Openlitespeed Webserver* (LSWS) kompatibel dengan fitur *Apache* yang umum digunakan termasuk mod_rewrite, .htaccess dan mod_security. LSWS dapat memuat file konfigurasi *Apache* secara langsung dan berfungsi sebagai pengganti *Apache* dengan integrasi penuh dengan panel kontrol populer. LSWS menggantikan semua fungsionalitas *Apache*, tetapi memproses permintaan dengan cara berbasis peristiwa. OpenLiteSpeed juga diklaim memiliki performa yang baik dan cocok untuk mengatasi beban tinggi. *LiteSpeed Cache* adalah fitur bawaan yang dapat mempercepat waktu respon situs. Sedangkan Nginx dikenal memiliki performa tinggi dan efisiensi dalam menangani banyak koneksi bersamaan. Cocok untuk digunakan sebagai server proxy dan load balancer. Pada *OpenLiteSpeed* juga menyediakan panel kontrol web (*LiteSpeed WebAdmin Console*) yang mudah digunakan, memudahkan konfigurasi dan administrasi bahkan untuk pengguna yang kurang berpengalaman. Sedangkan Nginx menggunakan konfigurasi melalui file teks, yang memerlukan pengetahuan teknis lebih mendalam dalam administrasi server. [14].

2.2.7 GNS 3

GNS3 adalah Program simulator jaringan grafis yang dapat mensimulasikan topologi jaringan yang lebih kompleks dibandingkan dengan simulator lainnya. Program ini dapat digunakan di beberapa sistem operasi seperti Windows, Linux atau MacOS X. Dynamis merupakan *Software* buatan Christophe Fillot. Perangkat lunak ini dirancang untuk mensimulasikan *router* IOS untuk seri Cisco 1700, 2600, 3600, 3700 dan 7200. Dynamips dirancang untuk pelatihan, pengujian, eksperimen dan pengujian kualitas konfigurasi IOS pada *router* nyata. Perangkat lunak ini didasarkan pada CLI dan tidak memiliki mode GUI, jadi Anda perlu memahami perintahnya. Dynamips dapat berjalan di beberapa sistem operasi seperti Linux dan Windows. Ini adalah emulator pc/simpul. Prinsip kerja GNS3 adalah menyalin Cisco IOS di komputer, sehingga dengan mengaktifkan fungsi Ethernet*Switch*Card, komputer dapat bertindak sebagai satu atau lebih *router* dan bahkan sebagai *switch*. ..[15]

2.2.8 *STRESS TEST*

Sresst Test Testing atau ujicoba, adalah proses menjalankan aplikasi dengan tujuan menemukan masalah. Pengujian kinerja dilakukan dengan membuat permintaan dalam jumlah besar, seperti mengakses sistem dengan banyak pengguna dalam waktu bersamaan. *Stress testing* adalah bagian dari *Performance testing* yang berfokus pada pengujian aplikasi atau sistem dalam kondisi ekstrim. Kondisi yang dikatakan ekstrim antara lain beban yang berat, concurrency yang tinggi, atau sumber daya komputasi yang terbatas. *Stress testing* yang baik juga berguna untuk menemukan *bug* terkait sinkronisasi, masalah *interlock*, dan *bug* kehilangan sumber daya. *Stress testing* dilakukan untuk memastikan bahwa sistem tidak akan *crash* dalam situasi krisis. Penggunaan *stress testing* yang paling mencolok adalah untuk menentukan batas-batas, di mana sistem rusak. Cara kerja dari *Stress test* adalah mengetahui response *time* atau latency, *Throughput*, resource utilisasi dan beban kerja. Stress test pada *webserver* melibatkan simulasi beban tinggi atau lonjakan lalu lintas dengan tujuan menguji batas kapasitas dan kinerja server. Proses ini dilakukan dengan mengirimkan sejumlah besar permintaan HTTP secara bersamaan atau berurutan kepada server, menciptakan situasi overload untuk mengidentifikasi titik kerentanan atau kegagalan sistem. Metrik seperti waktu respons, latensi, throughput, dan penggunaan sumber daya seperti CPU dan RAM dipantau selama pengujian.[16]

2.2.9 *Respone time*

Response time atau Waktu Respon mengacu pada jumlah waktu yang diperlukan sistem atau aplikasi untuk menanggapi permintaan atau tindakan yang dilakukan terhadapnya. Lebih khusus lagi, dalam konteks *server web*, waktu respons mengacu pada waktu yang dibutuhkan *server* untuk menerima permintaan HTTP dari klien dan mengirim respons ke klien. Waktu respons sering diukur sejak permintaan dikirim ke *server* hingga klien menerima respons penuh. Ini termasuk waktu yang dibutuhkan untuk memproses permintaan, mendapatkan informasi yang diperlukan, melakukan perhitungan dan mengembalikan respon ke pelanggan. [17] . Rumus dari *Respone time* :

Response Time = *Network Latency* + *Server Processing Time* + *Transmission Time* (2.1)

Processing Time = waktu yang diperlukan untuk memproses permintaan atau tugas.

Network Latency = waktu yang dibutuhkan untuk data bergerak dari klien ke server dan kembali lagi..

Browser Rendering Time = waktu yang dihabiskan oleh browser untuk merender dan menampilkan halaman web.

2.2.10 *Throughput*

Throughput adalah ukuran kuantitatif dari jumlah pekerjaan yang dapat dilakukan oleh sistem atau aplikasi dalam periode waktu tertentu. Dalam konteks pemrosesan data, *Throughput* mengacu pada jumlah tugas atau permintaan yang dapat diproses atau dijalankan oleh sistem dalam satu unit. [18]. Rumus dari *Throughput* adalah :

Throughput = Jumlah Beban user / Waktu (2.2) Di

mana:

Jumlah Beban = total permintaan atau transaksi beban user yang berhasil diproses oleh *server* dalam periode waktu tertentu.

Waktu = periode waktu dalam satuan yang sama dengan yang digunakan untuk menghitung jumlah permintaan (misalnya, detik, menit, jam).

2.2.11 *Ram Usage*

Ram Usage, atau penggunaan memori RAM mengacu pada jumlah memori fisik yang digunakan oleh sistem operasi, program, atau proses yang berjalan di komputer. RAM (*Random Access Memory*) adalah tempat penyimpanan sementara yang digunakan untuk menjalankan program dan menyimpan data aktif. Penggunaan RAM yang tinggi dapat memengaruhi kinerja sistem. Ketika RAM mendekati kapasitas maksimumnya, sistem dapat melambat dan pertukaran (penggantian data dalam RAM dengan data dari *hard drive*) dapat terjadi, yang dapat mempengaruhi kinerja secara serius. Pemantauan penggunaan RAM penting untuk mengelola kinerja sistem dan memastikan alokasi sumber daya yang tepat. Dengan memahami penggunaan RAM, pengguna atau administrator sistem dapat mengidentifikasi program atau proses yang menghabiskan banyak memori dan mengambil tindakan jika diperlukan, seperti menutup program yang tidak diperlukan atau menambah kapasitas RAM. Satuan yang umum digunakan untuk mengukur penggunaan RAM (*Random Access Memory*) adalah "*byte*" dan turunannya [19]. Rumus dari dari RAM *Usage* adalah :

RAM *Usage* = Total RAM - *Available* RAM (2.3) Di mana:

Total RAM: Jumlah total RAM fisik yang ada pada *server*.

Available RAM: Jumlah RAM yang tersedia atau tidak digunakan pada saat pengujian.

2.2.12 *Cpu Usage*

Cpu Usage/ penggunaan CPU atau pemanfaatan CPU mengacu pada berapa banyak daya pemrosesan yang dikonsumsi oleh sistem komputer atau aplikasi di *Central Processing Unit* (CPU). CPU adalah komponen utama dari sistem komputer yang bertanggung jawab untuk mengeksekusi instruksi dan memproses data. Penggunaan CPU diukur sebagai persentase dan menunjukkan berapa banyak waktu CPU yang diperlukan untuk menjalankan instruksi dari program atau proses yang sedang berjalan. Penggunaan CPU dapat bervariasi berdasarkan beban kerja dan tugas sistem [20]. Rumus CPU *Usage* adalah : CPU *Usage* (%) = (CPU *Time Used* / Total CPU *Time*) × 100 (2.4) Di mana:

CPU *Time Used*: Waktu pemrosesan yang digunakan oleh sistem atau proses dalam periode waktu tertentu.

Total CPU *Time*: Total waktu pemrosesan yang tersedia dalam periode waktu yang sama.