BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Tinjauan pustaka merupakan acuan utama dalam beberapa studi yang pernah dilakukan yang berkaitan dengan penelitian ini. Beberapa penelitian yang digunakan sebagai acuan dalam penelitian ini:

Penelitian ini menggunakan beberapa pustaka yang berkaitan dengan penelitian terkait analisa subjek yang berbeda dengan yang akan dianalisa. Hal ini berfungsi untuk menjadi pedoman dan pembanding penelitian yang akan dilakukan.

Matius Krisna Gunarno (2021) pernah melakukan penelitian tentang Analisis Perbandingan *Load Balancing* Antara *Nginx* dan *Traefik* Pada Aplikasi *Wordpress* Berbasis *Docker Container*. Penelitian ini menghasilkan kesimpulan bahwa *Nginx* sebagai *load balancer* mengungguli *Traefik* hampir seluruh skenario pengujian. Terjadi perbedaan signifikan terhadap *latency* pada *Traefik*, meningkatnya distribusi *latency* pada persentil 90% keatas. Nginx juga unggul dalam *request per second*, yang mampu menyesuaikan peningkatan jumlah permintaan. *Error rate* pada Nginx dan Traefik tidak terlalu banyak perbedaan dan Traefik unggul dalam kondisi khusus (*concurrency* 750 dan 1000).

Sampurna Dadi Riskiono dan Donaya Pasha (2020) pernah melakukan penelitian tentang Analisis Perbandingan *Server Load Balancing* dengan *Haproxy* & *Nginx* dalam Mendukung Kinerja *Server E-Learning*. Penelitian ini

menghasilkan kesimpulan bahwa Haproxy memiliki *response time* yang lebih kecil dari Nginx. Selain itu *Throughput* dari Haproxy juga lebih besar daripada Nginx.

Fahmi Apriliansyah, Iskandar Fitri, dan Agus Iskandar (2020) melakukan penelitian tentang Implementasi *Load Balancing* Pada *Web Server* Menggunakan Nginx. Penelitian ini menghasilkan kesimpulan bahwa kemampuan sistem *load balancing* Nginx dengan 3 metode algoritma *Round Robin, Least Connection*, dan *IP Hash* yang terbaik adalah *Least Connection*. Hasil yang didapatkan *Least Connection* mendapatkan *response time* 116ms, 2300.96 req/s dan *throughput* 17380.01 kbps Sedangkan *Round Robin* mendapatkan 140ms, 2335.36 req/s dan *throughput* 17098.05 kbps. Kemampuan sistem *network load balancing* Nginx dalam melayani *request* lebih besar dibandingkan dengan *single server*.

Albert Yakobus Chandra (2019) melakukan penelitian tentang Analisis Performansi Antara Apache & Nginx *Web Server* dalam Menangani *Client Request* yang pada penelitiannya mendapat kesimpulan bahwa Nginx memiliki rata-rata waktu penyelesaian *request* yang lebih cepat dibandingkan dengan Apache. Hasil ini didapatkan setelah proses pengujian dengan jumlah *request* mulai dari 100 sampai 1000000 dengan menggunakan *tool Apache Bench*.

Andri Jiwandoro (2021) pernah melakukan penelitian tentang Analisa Perbandingan Kinerja Web Server Apache, Nginx, Dan Litespeed Dengan Menggunakan Metode Stress Test. Penelitian ini menghasilkan beberapa kesimpulan, termasuk pengujian pada *Error, Sent* dan *Received*. Pada *Error,* didapat kesimpulan bahwa OpenliteSpeed memiliki tingkat kegagalan yang lebih

kecil dibandingkan *web server* Apache dan Nginx, menunjukkan bahwa OpenliteSpeed memiliki kinerja yang paling optimal. Untuk *Sent* dan *Received web server* Apache lebih unggul dibanding Nginx. Hal ini menunjukkan Apache lebih baik dalam mengelola permintaan client dan responsif dalam menanggapi.

Perbandingan pada penelitian di atas dengan penelitian yang akan dikerjakan saat ini dapat dilihat pada Tabel 2.1.

Tabel 2.1 Tabel Perbandingan

Penulis	**Objek Penelitian**	**Tools**	**Perbedaan dengan topik yang sedang diteliti**
Matius Krisna Gunarno (2021)	Menganalisis perbandingan *Load Balancing* antara *Nginx* dan *Traefix* pada aplikasi *Wordpress* berbasis *Docker Container*	-	Perbedaan utama terletak pada objek penelitian yaitu pada penelitian ini menganalisa perbandingan antara *Nginx* dan *Caddy*
Sampurna Dadi Riskiono, Donaya Pasha (2020)	Menganalisis perbandingan *Load Balancing* antara *Haproxy* dan *Nginx* dalam mendukung kinerja *server E-learning*	-	Perbedaan utama terletak pada objek penelitian yaitu pada penelitian ini menganalisa perbandingan antara *Nginx* dan *Caddy*
Fahmi Apriliansyah, Iskandar Fitri, Agus Iskandar (2020)	Mengimplementasi *Load Balancing* pada web server Nginx	-	Perbedaan utama terletak pada subjek penelitian yaitu pada penelitian ini melakukan perbandingan antara Nginx dengan Caddy dengan mengetahui performa Load Balancing dari keduanya.

Albert Yakobus Chandra (2019)	Menganalisa performa *web server Apache* dan *Nginx* dalam menangani *client request.*	Apache Bench	Perbedaan dengan penelitian ini terdapat pada objek penelitian dimana pada penelitian ini menganalisa performansi dari Nginx dengan performansi dari Caddy
Andri Jiwandoro (2021)	Menganalisa perbandingan kinerja *web server Apache, Nginx*, dan *Litespeed* dengan metode *stress test*.	Apache JMeter	Perbedaan utama dengan penelitian ini terletak pada perbandingan antara Nginx dan Caddy.

2.2 Dasar Teori

2.2.1 Pengertian Analisis

Menurut Kamus Besar Bahasa Indonesia, "Analisis adalah penguraian suatu pokok atas berbagai bagiannya dan penelaahan bagian itu sendiri serta hubungan antara bagian untuk memperoleh pengertian yang tepat dan pemahaman arti keseluruhan ".

Menurut Nana Sudjana (2016:27) "Analisis adalah usaha memilah suatu integritas menjadi unsur-unsur atau bagian-bagian sehingga jelas hirarkinya dan atau susunannya ".

Menurut Abdul Majid (2013:54) "Analisis adalah kemampuan menguraikan menguraikan satuan menjadi unit-unit terpisah, membagi satuan menjadi sub-sub atau bagian, membedakan antara dua yang sama, memilih dan mengenali perbedaan di antara beberapa yang dalam satu kesatuan".

Dari beberapa pendapat di atas, dapat disimpulkan bahwa analisis adalah suatu kegiatan untuk menemukan temuan baru terhadap objek yg akan diteliti

ataupun diamati oleh peneliti dengan menemukan bukti-bukti yg akurat pada objek tersebut.

2.2.2 Website

Menurut Sibero (2013:11) ”*Web* adalah suatu sistem yang berkaitan dengan dokumen yang digunakan sebagai media untuk menampilkan teks, gambar, *multimedia*, dan lainnya pada jaringan *internet*”.

2.2.3 Web Server

Menurut Sibero (2013:11) ”*Web server* adalah sebuah komputer yang terdiri dari perangkat keras dan perangkat lunak”.

Sedangkan menurut Kustiyahningsih dan Devie (2011:8) ”*Web server* adalah komputer yang digunakan untuk menyimpan dokumen-dokumen *web*, komputer ini melayani permintaan dokumen *web* dari kliennya”.

Dari penjelasan teori di atas, dapat disimpulkan bahwa *web server* adalah komputer yang digunakan untuk menyimpan dokumen dengan mengakses dan menampilkan halaman web tersebut dari komputer client.

2.2.4 NGINX

Menurut Albert (2019:02) ”Nginx atau biasa disebut “*Engine-x*”, adalah *opensource web server*”. Nginx selain digunakan sebagai web server juga memiliki fitur untuk digunakan sebagai *reverse proxy, HTTP cache, dan load balancer*.

Nginx dibuat oleh Igor Sysoev dan dirilis ke publik pada bulan Oktober 2004. Nginx menawarkan penggunaan memori yang lebih rendah dibandingkan

web server lainnya dan juga beberapa fitur seperti: *reverse proxy, IPv6, load balancing, FastCGI support, web sockets, handling static files, TLS/SSL.*

2.2.5 Caddy

Menurut O'Reilly (2018:02) "Caddy atau biasa juga disebut Caddy *Web Server* adalah sebuah aplikasi *web server opensource* yang ditulis dalam bahasa Go dan menawarkan fitur utama dukungan ke protokol *HTTP/2* dan *HTTPS by default*."

Caddy *Web Server* dirilis oleh Matthew Holt sekitar April 2015. Saat ini Caddy *Web Server* sendiri sudah dapat digunakan pada hampir semua platform seperti Windows, Mac, Linux, Android dan BSD, untuk arsitektur 32bit, 64bit, dan ARM.

2.2.6 Wordpress

"*Wordpress* tidak hanya sebagai *blog tool* yang dapat dipasang ke dalam *server* pengguna, namun juga menyediakan layanan *hosting blog* gratis sebagai layanan blogger" (Budiarto, 2010:18)

WordPress.com merupakan situs layanan *blog* yang menggunakan mesin *WordPress*, didirikan oleh perusahaan Automattic. Dengan mendaftar pada situs WordPress.com, pengguna tidak perlu melakukan instalasi atau konfigurasi yang cukup sulit.

2.2.7 Load Balancing

"Sistem *load balancing* bertugas untuk mendistribusikan beban kerja ke banyak *server*, dengan mempertimbangkan kapasitas dari setiap *server* yang ada" (Dani & Suryawan, 2012)

"Sistem *load balancing* dapat bekerja dengan baik ketika *request* datang dari klien telah berhasil didistribusikan *balancer* secara merata kepada setiap *node cluster*. Sehingga *server* tidak mengalami *overload* dan kemampuan *web server* bisa melayani hingga 10.000 request dengan tidak mengalami *error request"* (Rahmatulloh & MSN, 2017).

2.2.8 Docker Container

Menurut Sugianto (2016) "Docker adalah suatu *platform* terbuka bagi pengembang perangkat lunak dan pengelola sistem jaringan untuk membangun, mengirimkan dan menjalankan aplikasi-aplikasi terdistribusi"

Definisi tersebut membawa pengertian praktis bahwa *Docker* merupakan suatu cara memasukkan layanan ke dalam lingkungan terisolasi bernama *container*, sehingga layanan tersebut dapat dipaketkan menjadi satu bersama dengan semua pustaka dan *software* lain yang dibutuhkan.

2.2.9 Podman

Menurut M. A. Aziz (2020), "Podman merupakan *platform container open-source* yang digunakan untuk menjalankan maupun memanajemen ekosistem *container* seperti *pods*, *containers*, *container images*, dan *container volumes* menggunakan *libpod library*."

Dengan Podman, sebuah layanan dapat berjalan sebagai proses terisolasi, independen terhadap lingkungan sekitarnya.

2.2.10 Algoritma Round Robin

Round Robin merupakan algoritma paling sederhana dan paling banyak digunakan oleh perangkat *load balancing*. *Round Robin* bekerja dengan cara membagi beban secara bergiliran dan berurutan dari satu *server* ke *server* lainya (Diarjo & Mulyana, 2017)

2.2.11 Algoritma Least Connection

Least Connection melakukan pembagian beban berdasarkan banyaknya koneksi yang sedang dilayani oleh sebuah *server*. *Server* dengan koneksi yang paling sedikit akan diberikan beban berikutnya (Arman, Wijaya, & Irsyad, 2017).

2.2.12 k6

Grafana k6 adalah alat *open-source load testing* yang membuat pengujian performa mudah dan produktif untuk tim teknis. k6 adalah gratis, berpusat pada *developer*, dan mudah dikembangkan. Dengan k6, kamu dapat menguji keandalan dan performa sistem dan menangkap penurunan performa dan masalah lebih awal. k6 akan membantu menbangun aplikasi yang tangguh dan memiliki performa yang baik. (https://k6.io/docs/#what-is-k6).

2.2.13 RED Method

RED Method mendefinisikan 3 metrik utama yang digunakan untuk menghitung, yaitu Request Rate, Request Error, dan Request Duration. Request Rate yaitu tingkat permintaan yang diproses dalam hitungan detik. Untuk Request

Error berupa jumlah permintaan yang gagal untuk per detik. Request Duration berupa ditribusi jumlah waktu yang digunakan dalam setiap permintaan. Metode RED digunakan untuk menyamakan metrik *service* yang dihasilkan, sehingga memudahkan skalabilitas operasional tim. (https://www.weave.works/blog/the-red-method-key-metrics-for-microservices-architecture/).