BAB II
TINJAUAN PUSTAKA

2.1. Kajian Pustaka

Penelitian terdahulu yang telah dilakukan adalah implementasi pengukuran terhadap performa algoritma *weighted round robin* dengan membandingkan antara *load balancing cluster* dengan *load balancing non cluster* dengan variabel yang diukur adalah *response time* dan *throughput* yang dihasilkan. Penelitian tersebut menghasilkan data bahwa *load balance cluster* memiliki *response time* dan *throughput* yang lebih signifikan. Untuk implementasi di *non cluster* memiliki performa yang lebih rendah, sehingga cocok diimplementasikan untuk *web server* dengan *server* lebih dari satu atau sistem *cluster* [8].

Penelitian sebelumnya telah dilakukan implementasi *load balancing server* menggunakan Haproxy dan Nginx. Implementasi dilakukan pada suatu web server *E-Learning*. Berdasarkan penelitian tersebut diketahui bahwa *load balancing* menggunakan Haproxy rata-rata memiliki *response time* dan *throughput* yang lebih unggul [3].

Penelitian mengenai implementasi *load balancing* server untuk sistem informasi akademik menggunakan Haproxy dan sinkronisasi *file*. Algoritma yang digunakan adalah *round robin* dengan tiga web server Apache dan satu server basis data. Pada pengujian fail over dengan menggunakan Apache Bench menghasilkan data bahwa saat hanya satu server yang tersedia, jumlah *failed request* banyak sedangkan saat semua web server tersedia tidak ada *failed request* [5].

Telah dilakukan penelitian *load balancing* pada *E-Learning* dan dilakukan implementasi *load balancing round robin* dan *least connection*. Dilakukan pengujian dengan melakukan koneksi sebanyak 500 koneksi per 10 detik. Dari penelitian tersebut, *website E-Learning* yang dijalankan menggunakan *server load balancing* dan dijalankan menggunakan dua server *E-Learning* memiliki waktu respon lebih kecil dibandingkan server tunggal [2].

Pada penelitian tentang perbandingan kinerja antara server *load balancing* menggunakan metode *round robin* dan *weighted round robin* menggunakan dua

cluster web server menghasilkan kesimpulan bahwa total *response time* mengalami peningkatan sejalan dengan meningkatnya *trafik* ke web server. Pada algoritma *round robin*, penggunaan dua web server dengan spesifikasi sama menghasilkan *response time* yang berbeda dengan salah satu server mendapatkan *trafik* lebih besar. Total *response time* pada metode *round robin* mengalami peningkatan lebih tinggi dibandingkan metode *weighted round robin* saat *trafik* semakin banyak [10]. Lebih jelasnya hubungan penelitian ini dengan penelitian terdahulu, dapat dilihat dari tabel 2.1. di bawah ini.

Tabel 2.1 Kajian Pustaka

No.	Judul dan Tahun	Objek Yang Dibahas	Metode Yang digunakan	Hasil	Perbedaan dengan Penelitian Sekarang
1	Implementasi *Load Balancing* Dengan Algoritma Penjadwalan *Weighted Round Robin* Dalam Mengatasi Beban Webserver (2021)	Web Server dengan Ipvsadm sebagai *load balancer*	*Weighted round robin*	Kinerja *Load balancing cluster* Lebih baik dalam segi *response time* dan *throughput* dibandingkan dengan *non cluster*	Menggunakan algoritma *weighted round robin* dan *weighted least connection* dan *load balancer* Haproxy dengan studi kasus *website E-Learning*

No.	Judul dan Tahun	Objek Yang Dibahas	Metode Yang digunakan	Hasil	Perbedaan dengan Penelitian Sekarang
2	Analisis Perbandingan Server *Load Balancing* dengan Haproxy & Nginx dalam Mendukung Kinerja Server *E-Learning* (2020)	Server *E-Learning*	*Round robin* dan *least connection*	*Load balancing* menggunakan Haproxy lebih kecil dibandingkan menggunakan Nginx dalam segi *response time* dan *throughput.* dengan data tersebut bisa diketahui jumlah paket yang bisa dilewatkan dengan Haproxy lebih besar	Menggunakan algoritma *weighted round robin* dan *weighted least connection*
3	Implementasi *Load Balancing* Web Server menggunakan Haproxy dan Sinkronisasi *File* pada Sistem Informasi Akademik Universitas Siliwangi (2017)	Sistem Informasi Akademik Universitas Siliwangi	*Round robin*	Web server dapat menampung 1000 *request* jika *request* berhasil di distribusikan ke *node cluster*. *File* berhasil di sinkronisasi antar *node*. *Response time* berbanding lurus dengan *current connection* server.	Menggunakan algoritma *weighted round robin* dan studi kasus *website E-Learning*

No.	Judul dan Tahun	Objek Yang Dibahas	Metode Yang digunakan	Hasil	Perbedaan dengan Penelitian Sekarang
4.	Analisis Metode *Load Balancing* Dalam Meningkatkan Kinerja *Website E-Learning* (2020)	*Website E-Learning*	Metode *round robin* dan *least connection* pada server tunggal dan *cluster*	*Least connection* memiliki *response time* lebih kecil dan *throughput* lebih besar dibandingkan *round robin*. Server tunggal tidak mampu menangani 500 koneksi per 20 detik.	Menggunakan algoritma *weighted round robin* dan w*eighted least connection*
5.	Analisa Kinerja *Load Balancing* Menggunakan Metode *Round Robin* Dan *Weighted Round Robin* (2021)	Analisis perbandingan *response time* pada *load balancing* web server Apache.	Metode *round robin* dan *weight round robin*	Algoritma *round robin* memiliki total *response time* lebih tinggi dibandingkan *weighted round robin* ketika semakin banyak *trafik*.	Menggunakan algoritma *weighted least connection* dan mengukur *throughput*, *packet loss* dan *bandwidth*.

2.2. Dasar Teori

2.2.1. *Cluster*

Cluster atau *Clustering* adalah sebuah solusi yang bisa digunakan untuk mencegah atau menanggulangi masalah tentang terlalu banyak *trafik* ke satu web server. Cara yang digunakan adalah dengan menggunakan lebih dari satu web server digabungkan menjadi *cluster* yang saling bekerja sama tetapi menjadi satu kesatuan seolah sistem tunggal. Walaupun tunggal, tapi beban dibagi ke seluruh

server yang terhubung menjadi *cluster* [13]. *Clustering* membuat server menjadi dua fungsi utama yaitu fungsi *fail over* atau penggantian server saat salah satu server mengalami kerusakan dan fungsi *load balancing* atau pembagian beban kerja kepada beberapa server yang tergabung dalam *cluster* agar tetap menjaga kestabilan server [5].

2.2.2. Apache

Apache merupakan sebuah perangkat lunak yang digunakan untuk menjalankan sebuah *file* web dari sisi server. Apache dalam sebuah server bertindak untuk menyediakan web agar dapat diakses melalui *browser* oleh klien. Apache dapat ditemukan di banyak sistem operasi seperti Windows, Linux, *BSD* dikarenakan merupakan perangkat lunak terbuka (*Open Source*) [14].

2.2.3. Haproxy

Haproxy (*High Availability Proxy)* merupakan sebuah perangkat lunak bebas untuk dimodifikasi yang digunakan untuk *reverse proxy* atau *HTTP load balancer*. Perangkat lunak ini bekerja dengan melakukan *forwarding* ke server yang memiliki layanan dalam hal ini *web server* [3]. Haproxy bekerja dengan cara mengintegrasikan dengan arsitektur sistem yang sudah ada sehingga dalam pemasangannya tidak terlalu mengubah sistem yang sudah ada atau hanya perlu membuat rute baru [10].

2.2.4. CentOS

CentOS (*Community Enterprise Operating System*) adalah sebuah distro Linux yang biasa dipakai dalam sekala *Enterprise* (perusahaan) tetapi walaupun tingkat perusahaan, sistem operasi ini tidak berbayar. Dokumentasi dari CentOS sudah cukup banyak di internet sehingga mengurangi kebingungan jika memakai perangkat lunak ini untuk membangun server. Dikarenakan sekalanya yang diperuntukkan untuk server, maka keamanan dari perangkat lunak ini, cukup tinggi [15].

2.2.5. *Load Balancing*

Load balancing merupakan sebuah metode yang digunakan untuk mendistribusikan banyak beban kerja dalam hal ini *trafik* permintaan klien ke beberapa komputer server dengan tujuan untuk menciptakan kestabilan dalam

menangani permintaan *user* dan meningkatkan *throughput* serta menghindari server tersebut mengalami *overload* karena beban yang melebihi kemampuan. Respon server saat diimplementasikan *load balancing* akan meningkat dikarenakan pembagian beban tersebut [14].

2.2.6. *Weighted Round Robin* (WRR)

Algoritma penjadwalan *weighted round robin* merupakan salah algoritma penjadwalan pada server *load balancing*. Cara kerjanya dengan membuat beban kerja server dapat berjalan seimbang dengan memberikan sebuah nilai sebagai jumlah bobot ke masing-masing web server atau *node cluster*. Semakin besar nilai yang diberikan, maka semakin sering server tersebut diberikan pekerjaan oleh server *load balancing*. Algoritma *load balancing weighted round robin* merupakan pengembangan dari algoritma *round robin* yang dapat mempertimbangkan seberapa besar beban server berdasarkan kapasitas yang ditentukan. Algoritma penjadwalan *weighted round robin* memasukkan bobot dengan menambahkan parameter secara manual kepada masing-masing web server atau *node cluster* berdasarkan *resource* atau spesifikasi perangkat keras yang dimiliki oleh masing-masing web server, sehingga penjadwal kerja dalam hal ini server *load balancing* akan memprioritaskan pekerjaan untuk server tersebut lebih dari pekerjaan server yang lainnya yang ditentukan bobotnya lebih kecil [8].

2.2.7. *Weighted Least Connection* (WLC)

Algoritma ini merupakan algoritma penjadwalan server *least connection* bedanya dalam konfigurasinya ditentukan bobot kinerja yang bisa diterima oleh sebuah server. Semakin tinggi nilai bobot yang ditentukan, maka akan semakin besar persentase mendapat bagian kerja dari koneksi-koneksi aktif dari satu waktu. jadi penentuan bobot kerja pada masing-masing server nyata atau *backend server* dapat ditentukan secara spesifik dan dijadwalkan juga koneksi jaringan masing-masing *backend server* dengan persentase koneksi aktif masing-masing sesuai bobot yang ditentukan dimana bobot awal tanpa dikonfigurasi dimulai dari satu [16].

2.2.7. Httperf

Httperf merupakan sebuah perangkat lunak yang digunakan sebagai alat untuk mengukur kinerja dari web server. Dari alat ini bisa didapatkan data yang dapat digunakan untuk mengevaluasi dan memperbaiki masalah yang mungkin terjadi terhadap performa web server [8]. Httperf dapat digunakan untuk mengukur *throughput* dan *response time* dari suatu server. *Throughput* adalah *bandwidth* sebenarnya dari suatu jaringan. Jadi *throughput* dihitung dari mulai pengiriman *request* ke server hingga mendapatkan respon sehingga biasanya lebih rendah dari *bandwidth* yang tertulis pada spesifikasi perangkat jaringan. *Response time* merupakan waktu yang diperlukan oleh server menangani *request* dari klien dan biasanya dalam satuan ms (*micro second*) [7].

2.2.9. Iperf

Iperf adalah sebuah *tools* yang digunakan melalui terminal melalui perintah Linux yang digunakan untuk mengukur kinerja *bandwidth* melalui protokol TCP dan UDP untuk menampilkan nilai *throughput, delay*, *jitter*, dan *packet loss*. Berdasarkan hasil yang diberikan, Iperf dapat digunakan untuk mengukur performa jaringan baik dalam satu jaringan maupun antar jaringan melalui terminal [17]. *Delay* merupakan jeda pada saat data lewat pada beberapa titik pemberhentian pada jaringan internet dan *jitter* merupakan gabungan dari keseluruhan jeda waktu yang dihasilkan dari waktu awal memberikan *request* hingga mendapatkan balasan dari *request* yang dikirim. *Packet loss* adalah sebuah persentase dari paket data yang hilang saat pengiriman yang biasanya diakibatkan oleh tabrakan data atau kemacetan dalam pengiriman data [18].

2.2.10. Metode Prototipe

Metode ini adalah salah satu metode pengembangan perangkat lunak yang berdasarkan pada pembuatan model secara fisik sebuah sistem yang dijadikan sebagai bentuk awal dari sistem yang akan dibuat [19]. Metode ini digunakan untuk membuat prototipe yang fungsinya sebagai perantara antara pengembang dengan pengguna sistem sehingga dapat diciptakan sistem yang sesuai dengan kebutuhan pengguna. Prototipe yang telah dibuat bisa diabaikan atau ditambahkan komponen-komponen baru sesuai dengan kebutuhan yang diperlukan dan sesuai dengan

perencanaan dan analisis yang oleh pengembang berdasarkan saran pengguna. Selama proses pengembangan, juga dilakukan uji coba secara bersamaan [20].

Prototipe memiliki kelebihan dapat digunakan pada pengembangan sistem besar maupun kecil dan dalam penggunaan metode tersebut dapat berjalan lancar dan selesai tepat waktu. Penggunaan metode prototipe dapat memberikan keuntungan bagi semua pihak seperti pimpinan, pengembang dan pengguna karena keterlibatan pengguna secara menyeluruh [21].

Tahap utama dalam metode prototipe adalah analisis kebutuhan dengan cara pengumpulan data, desain cepat, membangun prototipe kemudian melakukan evaluasi dan perbaikan. Berikut ini langkah-langkah *prototyping* dapat dilihat pada gambar 2.1 berikut.

Gambar 2.1 Metode Prototipe [20]