BAB II TINJAUAN PUSTAKA

A. Penelitian Terdahulu

Tabel 2.1 Algoritma penjadwalan *load balancing* terintegrasi untuk cluster aplikasi web berbasis *nginx*

Judul	Masalah	Metode	Hasil	Kesamaan dengan penelitian yang dilakukan	Perbedaan dengan penelitian yang dilakukan
Algoritma penjadwalan *load balancing* terintegrasi untuk cluster aplikasi web berbasis *nginx* (Li *et al*. 2018)	Menganalisis penyebaran aplikasi web berdasarkan teknologi wadah, mennghitung metrik beban host yang dikombinasikan kinerja status.	Metode kuantitatif	Algoritma DWLC membantu repon aplikasi web 52,6 % dan 46,4 % lebih cepat dari algoritma *round robin* dan *least connection* biasa.	Menggunakan algoritma *load balancing* berbasis *nginx*.	Pengunaan algoritma DWLC pada *load balancing*.

Tabel 2.2 Prinsip desain dari *intelligent load balancing* untuk layanan web socket skalabel yang digunakan dengan komputasi grid

Judul	Masalah	Metode	Hasil	Kesamaan dengan penelitian yang dilakukan	Perbedaan dengan penelitian yang dilakukan
Prinsip desain dari *intelligent load balancing* untuk layanan web socket skalabel yang digunakan dengan komputasi grid (Alexeev*et al*. 2019)	Pentingnya penskalaan web socket dan load balancing sangat penting untuk tugas-tugas system komputasi grid berorientasi browser berskala besar, karena kenyamanan WebSocket yang belum pernah ada sebelumnya untuk pengiriman pesan antara banyak node web	Metode kuantitatif	Mengusulkan mekanisme negosiasi antara klien dan sistem penyeimbangan beban tentang server target untuk mengatur client. Prinsip-prinsip konstruksi infrastruktur multi-server yang dapat diskalakan dengan komunikasi interserver yang efektif secara intensif menggunakan *instance Redis*	Skema ini berfungsi dengan baik dengan permintaan HTTP, karena penyeimbang tidak perlu menyimpan koneksi di memori, itu menutupnya setelah tanggapan dikembalikan ke klien. Permintaan dari klien mana pun dapat diteruskan ke server mana pun, bergantung pada distribusi beban saat ini.	Pada penelitian perbedaan terletak pada metode penelitian yaitu dengan kualitatif.

Tabel 2.3 Analisis performansi *load balancing* dengan algoritma *round robin* dan *least connection* pada sebuah web server

Judul	Masalah	Metode	Hasil	Kesamaan dengan penelitian yang dilakukan	Perbedaan dengan penelitian yang dilakukan
Analisis performansi *load balancing* dengan algoritma *round robin* dan *least connection* pada sebuah web server (Pratama *et al*. 2015)	Penggunaan *load balancing* sebagai peningkatan performansi untuk sebuah web server.	Metode kuantitatif	Algoritma *round robin* lebih baik dibandingkan dengan *least connection.*	Penggunaan *load balancing round robin* dengan melihat pada nilai parameter *respon time.*	Perbandingan dua algoritma *load balancing* yaitu *round robin* dan *least connection* berdasarkan parameter *throughput* dan *request loss.*

Tabel 2.4 Desain *load balancing* dari *google cloud compute engine* vps dengan metode *round robin* di PT. lintas data Indonesia

Judul	Masalah	Metode	Hasil	Kesamaan dengan penelitian yang dilakukan	Perbedaan dengan penelitian yang dilakukan
Desain *load balancing* dari *google cloud compute engine* vps dengan metode *round robin* di PT. lintas data Indonesia (Desmulyati dan Putra, 2019)	PT. lintas data Indonesia belum mengimplementasikan sistem *load balancing* dan *fail over* pada sistem saat ini.	Metode kuantitatif	Dengan HAProxy terbukti dapat meningkatkan hinga 150% dalam penanganan isu saat ini yang sebelumnya adalah sekitar 30 ms	Penggunaan vps dari *google cloud platform* dan algoritma *load balancing round robin*	Penggunaan tiga web server dengan HAProxy sebagai *load balancing*

Tabel 2.5 Desain *load balancing* dari *google cloud compute engine* vps dengan metode *round robin* di PT. lintas data Indonesia

Judul	Masalah	Metode	Hasil	Kesamaan dengan penelitian yang dilakukan	Perbedaan dengan penelitian yang dilakukan
Analisis perbandingan *load balancing web server* tunggal dengan *web server cluster* menggunakan linux *virtual server* (Lukitasari dan Oklilas., 2010)	Memecah bagian web server tunggal dan cluster dengan *load balancing*, pengambilan dari membanding gkan data trafik bandwidth pada web server.	Metode kuantitatif. .	LVS mampu membuat kinerja dari sebuah server menjadi lebih ringan dan lebih cepat. Web server cluster dapat meningkatka n kecepatan sehingga waktu yang dibutuhkan lebih cepat.	Mengguna kan sistem operasi Linux Ubuntu.	Pada penelitian ini menggunakan llinux *virtual server* yang dibangun di atas sebuah *cluster* dari beberapa *real server*.

Tabel 2.6 Teknik *Load Balancing* : Sebuah *Study* yang Komprehensif

Judul	Masalah	Metode	Hasil	Kesamaan dengan penelitian yang dilakukan	Perbedaan dengan penelitian yang dilakukan
Teknik *Load Balancing* : Sebuah *Study* yang Komprehensif (Pandey et al, 2015)	Membahas teknik *load balancing* yang digunakan pada SIP kluster server, *cluster computing* , dan *cloud computing*	Metode kualitatif	Dengan *load balancing*, memberikan waktu respons yang baik, meningkatkan *throughput*, memanfaatkan pemulihan secara efektif.	Penggunaan *load balancing* untuk distribusi beban kerja ke beberapa server	Pada penelitian ini hanya membahas tentang teori dari algoritma *load balancing*

Tabel 2.7 Implementasi Load Balancing Web Server menggunakan Haproxy dan Sinkronisasi File pada Sistem Informasi Akademik Universitas Siliwangi

Judul	Masalah	Metode	Hasil	Kesamaan dengan penelitian yang dilakukan	Perbedaan dengan penelitian yang dilakukan
Implementasi Load Balancing Web Server menggunakan Haproxy dan Sinkronisasi File pada Sistem Informasi Akademik Universitas Siliwangi (Rahmatulloh dan Nursuwars, 2017)	Saat ini SIA Universitas Siliwangi dengan arsitektur server tunggal sering terjadi *overload*	Metode kuantitatif	*Load balancing* dapat bekerja dengan baik ketika *request* datang dari *client* telah berhasil didistribusikan *balancer* secara merata kepada setiap node cluster	Penggunaan load balancing dan metode kuantitatif	Pada penelitian ini menggunakan Haproxy sebagai *load balancing*

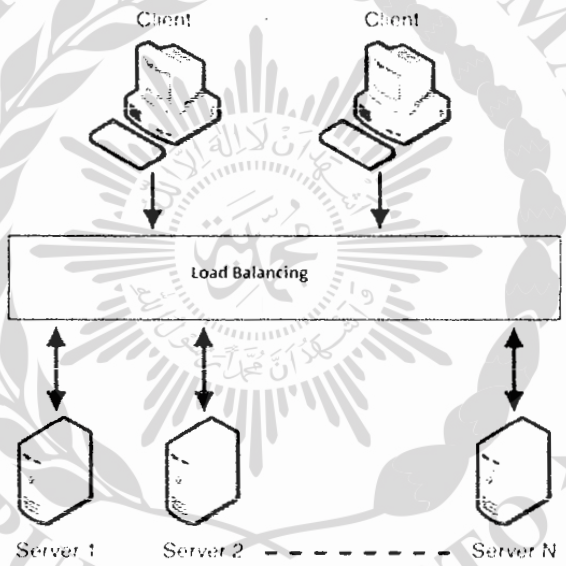
Tabel 2.8 *Load Balancing Web Server* Berbasis Cloud dengan Menggunakan Algoritma *Round Robin* padaSampoerna *University*

Judul	Masalah	Metode	Hasil	Kesamaan dengan penelitian yang dilakukan	Perbedaan dengan penelitian yang dilakukan
Load Balancing Web Server Berbasis *Cloud* dengan Menggunakan Algoritma *Round-Robin* pada Sampoerna *University* (Adidrajat dan Mulyani, 2019)	Universitas Sampoerna menggunakan *server* biznet *cloud* sering terjadi *down time*	Metode kuantitatif	Dengan *cloud computing* mengurangi *down time* sebuah web server yang dialami oleh Sampoerna *University*.	Penggunaan *cloud computing* GCP dan *round robin*	Pada penelitian ini menggunakan apache sebagai *load balancing*

B. Landasan Teori

1. Load Balancing

Menurut Rahmatulloh & Nursuwars (2017) *load balancing* adalah sebuah teknik untuk mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara seimbang, agar trafik dapat berjalan optimal, memaksimalkan *throughput*, memperkecil waktu tanggap dan menghindari *overload* pada salah satu jalur koneksi. Topologi *Load Balancing* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Konfigurasi arsitektur load balancing

Pada Gambar 2.1 *Load balancing* digunakan pada saat sebuah *server* telah memiliiki jumlah user yang telah melebihi maksimal kapasitasnya. *Load balancing* juga mendistribusikan beban kerja secara merata di dua atau lebih komputer, link jaringan, CPU, hard drive, atau sumber daya lainnya, untuk mendapatkan pemanfaatan sumber daya yang optimal.

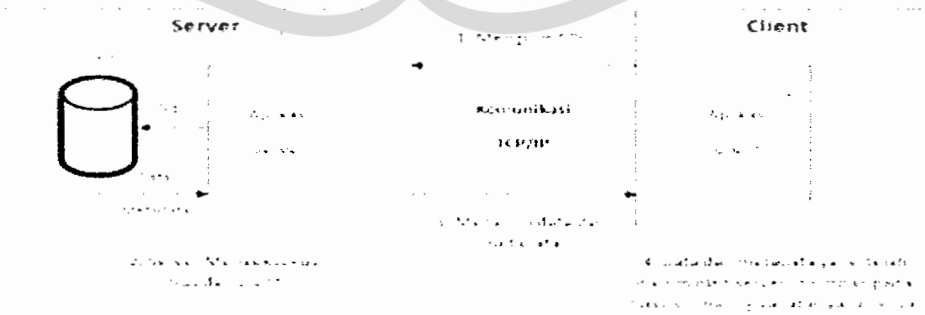
Sedangkan menurut Membrey, *et al.* (2012) menjelaskan fungsi dari *load balancing* dari kasus perusahaan besar yang menjual berbagai jenis mainan berbulu secara online. Masalahnya adalah bahwa seperti kebanyakan bisnis mereka memiliki periode sibuk dan sunyi. Rata-rata mereka akan mengambil sekitar 2.000 pesanan per minggu. Namun, pada waktu-waktu tertentu. *Server* mereka tidak bisa menangani beban sebanyak ini. Dengan banyak pesanan itu, kita bisa membayangkan berapa banyak halaman web dan grafik yang perlu dihasilkan dan dikirim kembali ke pelanggan. Untuk setiap pesanan yang dilakukan, puluhan orang bisa menjelajahi situs web. Selama periode normal, server mereka benar-benar tidak punya masalah mengatasi beban. Namun, selama periode tinggi, *server* akan mulai mengalami masalah.

Hasilnya adalah orang-orang yang hanya menjelajah menemukan situs itu sangat lambat sehingga mereka menyerah dan pergi ke tempat lain. Mereka yang melewati proses pemesanan hanya untuk mendapatkan batas waktu pada halaman pembayaran. Solusi yang jelas adalah meningkatkan sistem sehingga dapat menangani beban sebanyak ini. Namun, mereka melihat harga dan menemukan bahwa mesin seperti itu akan menjadi beberapa kali lipat dari harga *server* mereka saat ini. Diputuskan bahwa daripada membeli satu mesin besar, mereka akan membeli dua mesin dengan spesifikasi yang sama dengan *server* yang ada dan kemudian menggunakan solusi *load balancing*.

Load balancing mempunyai beberapa manfaat. Pertama, secara signifikan lebih murah untuk membeli dua mesin spek yang lebih masuk akal daripada membeli *server* spek yang jauh lebih tinggi yang dapat menangani beban dengan sendirinya. Ini juga berarti lebih banyak kapasitas untuk menangani pertumbuhan di masa depan karena secara umum, dua *server* menyediakan lebih banyak sumber daya daripada satu *server* tunggal. Jika satu *server* gagal, mereka masih bisa kembali menggunakan satu server. Ini juga berarti bahwa mereka dapat menurunkan *server* untuk pemeliharaan tanpa harus menurunkan situs *web*.

2. Socket

Socket adalah suatu abstraksi yang mana aplikasi dapat mengirim dan menerima data sama halnya dengan membuka suatu file untuk dibaca dan ditulis pada tempat penyimpanan file. *Socket* memungkinkan untuk masuk ke dalam jaringan dan berkomunikasi dengan aplikasi lain yang juga masuk ke dalam jaringan yang sama. Informasi yang ditulis ke dalam *socket* pada suatu aplikasi pada suatu mesin dapat dibaca oleh aplikasi lain pada mesin yang berbeda dan sebaliknya. *Socket* Jaringan dapat dilihat pada Gambar 2.2.



Gambar 2.2 Socket Jaringan

Pada Gambar 2.2 mengambarkan aplikasi *server* menunggu permintaan dari aplikasi *client*, hal ini dilaksanakan dengan menggunakan objek *Socket*. Ketika koneksi antara *server* dan *client* telah terhubung, aplikasi *client* dapat mengirimkan perintah sql ke aplikasi *server*. Aplikasi *server* merupakan aplikasi yang berjalan pada komputer yang memberikan layanan aplikasi. *Server* bersifat pasif, artinya bila tidak ada permintaan layanan dari *client*, maka program *server* menunggu tanpa memerlukan sumber daya CPU Cycle. Aplikasi *client* merupakan aplikasi yang berjalan pada komputer yang meminta layanan kepada *server*. Permintaan ini disampaikan kepada *server* dalam bentuk pesan sql. Ketika aplikasi *client* mengirim perintah sql, aplikasi *server* menerima permintaan dari *client* dan mengesekusi sql dari *client* yang terhubung dengan database access. Setelah teresekusi sesuai perintah sql dari *client*, database access memberikan data serta metadata ke aplikasi *server* dan apliikasi *server* mengirim metadata dan data tersebut kepada *client* sesuai permintaan *client* (Raven et al., 2015).

3. Web Server Toolkit 8

Delta dan Asmunin, (2016) menyatakan Webserver Stress Tool mensimulasikan sejumlah pengunjung/user yang mengakses situs Web pada waktu yang bersamaan dengan cepat dan mudah. Hal terpenting lagi adalah WST akan menunjukkan bagaimana kinerja dari infrastruktur bila server sedang padat. Webserver Stress Tool adalah sebuah aplikasi ampuh untuk menguji HTTP-client/server yang dirancang untuk menentukan kinerja Web ketika sedang mengalami masalah kritis bila sebuah Website atau pun server

milik website tersebut yang sedang mengalami lonjakan pengunjung. Pengertian yang ada di webserver tools 8:

1. Click: Merupakan pilihan yang baik untuk menguji URLs yang berurutan.
2. Time: Pengujian yang berjalan dalam jumlah menit yang telah ditentukan. Pengujian ini sering dilakukan untuk "burn in tests", contohnya untuk membiarkan server berada pada full load sepanjang beberapa jam.
3. Ramp: Ramp tests juga berjalan pada waktu yang telah ditentukan, tetapi dengan menyertakan peningkatan.

4. Linux

Linux adalah varian dari system operasi Unix yang sangat popular saat ini. Umumnya pemakaian Linux tidak ingin system operasi ini dianggap sebagai varian Linux, tetapi jika dirahasiakan dengan memakai, maka akan terlihat bahwa dari sisi pemakai semua yang ada dalam Linux adalah Unix, karena ternyata memang Linux diinspirasi oleh Minix (Mini Unix), dan dilandasi oleh beberapa *software* dari *GNU* (Sidik, 2004).

5. Statistik

Statistika yaitu bagian yang menjelaskan bagaimana data dikumpulkan dan di ringkas pada hal-hal yang penting dalam data tersebut. Dari sudut pandang statistika data dapat dibagi menjadi dua:

1. Data Kualitatif

   Data kualitatif adalah data yang dinyatakan dalam bentuk bukan angka. Misalnya jenis pekerjaan (petani, nelayan, pegawai, dsb.), status perkawinan, gender (jenis kelamin), kepuasan seseorang (tidak puasat,cukup puas, sangat puas), dsb.

2. Data Kuantitatif

   Data kuantitatif adalah data yang dinyatakan dalam bentuk angka. Misalnya usia seseorang, tinggi seseorang, penjualan dalam sebulan, dsb (Taniredja dan Mustafidah, 2011)

6. Uji Persyaratan

a. Uji Normalitas

Uji Normalitas bertujuan untuk menguji apakah dalam model regresi variabel pengganggu atau residual memiliki distribusi normal. Seperti diketahui bahwa uji t dan F mengansuKbpsikan bahwa nilai residual mengikuti distribusi normal. Kalau asuKbpsi ini dilanggar maka uji statistik menjadi tidak valid untuk jumlah sampel kecil. Ada dua cara untuk mendeteksi apakah residual berdistribusi normal atau tidak yaitu dengan analisis grafik dan uji statistik (Ghozali, 2011).

b. Uji Homogen

Uji homogenitas sebagai salah satu uji syarat analisa, selain uji normalitas dimaksudkan untuk memperlihatkan bahwa dua atau lebih kelompok data sampel berasal dari populasi yang memiliki varians yang sama. (Priyono, 2018).

7. Uji t Test Berpasangan

Siregar (2013) Uji t sample berpasangan sering kali disebut sebagai paired-sampel t test. Uji t untuk data sampel berpasangan membandingkan rata-rata dua variabel untuk suatu grup sampel tunggal. Uji ini menghitung selisih antara nilai dua variabel untuk tiap kasus dan menguji apakah selisih rata-rata tersebut bernilai nol.

8. SPSS

SPSS merupakan paket *software* statistika untuk analisis data. Program SPSS di buat pertama kali tahun 1968 oleh norman H. Nie bekerja sama dengan dua mahasiswa pascasarjananya di Stanford University bernama C. hadlai Hull dan Dale Bent. Program itu mereka sebut "statistical Package for the Social Sciences," atau disingkat "SPSS" Paket program SPSS kemudian berkembang menjadi produk multinasional dengan tetap menggunakan nama "SPPS" dan perusahaannya mereka sebut *SPSS Inc.*, yang bermarkas di Chicago, Illinois, USA (Stanislaus dan Uyanto, 2009).