MEKANISME DISTRIBUSI BEBAN PADA *LOAD BALANCING WEB SERVER* DENGAN ALGORITMA *LEAST CONNECTION* DAN *MULTI-AGENT SYSTEM* PADA LINGKUNGAN VIRTUAL

Web Server Load Balancing Mechanism with Least Connection Algorithm and Multi-Agent System

AFIYAH RIFKHA RAHMIKA
D082192010

PROGRAM STUDI S2 TEKNIK INFOMATIKA
DEPARTEMEN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS HASANUDDIN
GOWA
2023

MEKANISME DISTRIBUSI BEBAN PADA *LOAD BALANCING WEB SERVER* DENGAN ALGORITMA *LEAST CONNECTION* DAN *MULTI-AGENT SYSTEM* PADA LINGKUNGAN VIRTUAL

AFIYAH RIFKHA RAHMIKA
D082192010

PROGRAM STUDI S2 TEKNIK INFOMATIKA
DEPARTEMEN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS HASANUDDIN
GOWA
2023

PENGAJUAN TESIS

MEKANISME DISTRIBUSI BEBAN PADA *LOAD BALANCING WEB SERVER* DENGAN ALGORITMA *LEAST CONNECTION* DAN *MULTI AGENT SYSTEM* PADA LINGKUNGAN VIRTUAL

Tesis

Sebagai Salah Satu Syarat untuk Mencapai Gelar Magister

Program Studi Teknik Informatika

Disusun dan diajukan oleh

AFIYAH RIFKHA RAHMIKA

D082192010

Kepada

FAKULTAS TEKNIK

UNIVERSITAS HASANUDDIN

GOWA

2023

TESIS

MEKANISME DISTRIBUSI BEBAN PADA *LOAD BALANCING WEB SERVER* DENGAN ALGORITMA *LEAST CONNECTION* DAN *MULTI-AGENT SYSTEM* PADA LINGKUNGAN VIRTUAL

AFIYAH RIFKHA RAHMIKA
D082192010

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi Program Magister Teknik Informatika Fakultas Teknik Universitas Hasanuddin
pada tanggal 16 Februari 2023
dan dinyatakan telah memenuhi syarat kelulusan.

Menyetujui,

Pembimbing Utama,

Dr-Eng. Zulkifli Tahir, S.T., M.Sc.
NIP. 19840403 201012 1 004

Pembimbing Pendamping,

Dr. Eng. Ady Wahyudi Paundu, S.T., M.T.
NIP. 19750313 200912 1 003

Dekan Fakultas Teknik
Universitas Hasanuddin,

Prof. Dr. Eng. Ir. Muhammad Isran Ramli, S.T., M.T.
NIP. 19730926 200012 1 002

Ketua Program Studi
S2 Teknik Informatika

Dr. Ir. Zahir Zainuddin, M.Sc.
NIP. 19640427 198910 1 002

PERNYATAAN KEASLIAN TESIS DAN PELIMPAHAN HAK CIPTA

Yang bertanda tangan di bawah ini:

Nama : Afiyah Rifkha Rahmika

Nomor Mahasiswa : D082192010

Program Studi : S2 Teknik Informatika

Dengan ini menyatakan bahwa, tesis berjudul "Mekanisme Distribusi Beban Pada *Load Balancing Web Server* dengan *Algoritma Least Connection* dan *Multi-Agent System* Pada Lingkungan Virtual" adalah karya saya dengan arahan dari komisi pembimbing (Dr.Eng. Zulkifli Tahir, S.T., M.Sc., dan Dr-Eng. Ady Wahyudi Paundu, S.T., M.T.,). Karya ilmiah ini belum diajukan dan tidak sedang diajukan dalam bentuk apa pun kepada perguruan tinggi mana pun. Sumber informasi yang berasa atau dikutip dari karya yang diterbitkan maupun tidak diterbitkan dari penulis lain telah disebutkan dalam teks dan dicantumkan dalam Daftar Pustaka tesis ini. Sebagian dari isi tesis ini telah dipublikasikan di Jurnal/Prosiding (CommIT (Communication and Information Technology) Journal) sebagai artikel dengan judul *"Web Server Load Balancing Mechanism with Least Connection Algorithm and Multi-Agent System"*.

Dengan ini saya limpahkan hak cipta dari karya tulis saya berupa tesis ini kepada Universitas Hasanuddin.

Gowa, 20 Februari 2023

Yang menyatakan

Afiyah Rifkha Rahmika

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Allah SWT. karena berkat rahmat dan karunia-Nya sehingga tesis yang berjudul "**Mekanisme Distribusi Beban Pada *Load Balancing Web Server* Dengan Algoritma *Least Connection* dan *Multi-Agent System* Pada Lingkungan Virtual**" ini dapat diselesaikan sebagai salah satu syarat dalam menyelesaikan jenjang Strata-2 pada Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin.

Penulis menyadari bahwa dalam penyusunan dan penulisan laporan tesis ini tidak lepas dari bantuan, bimbingan serta dukungan dari berbagai pihak, dari masa perkuliahan sampai dengan masa penyusunan tesis. Oleh karena itu, penulis dengan senang hati menyampaikan terima kasih kepada:

1. Kedua Orang tua penulis, Bapak Alm. Ir. H. Bahrun Rafid dan Ibu Ir. Hj. Ramlawati Ahmad yang selalu menjadi motivasi terbesar dalam penyelesaian perkuliahan ini yang tidak pernah putus memberikan dukungan, doa, dan semangat serta selalu sabar dalam mendidik penulis sejak kecil;
2. Saudara kembar penulis, Afiyah Rifdha Rahmika, S.Ds yang dengan sangat sabar menemani dan memberikan semangat kepada penulis selama penyusunan tesis;
3. Bapak Dr.Eng. Zulkifli Tahir, S.T., M.Sc selaku pembimbing I dan Bapak Dr. Eng. Ady Wahyudi Paundu, ST., M.T. selaku pembimbing II yang telah memberikan waktu, tenaga, pikiran, dukungan moril maupun materil serta perhatian yang luar biasa untuk mengarahkan penulis dalam penyusunan tesis;
4. Bapak Dr. Eng. Muhammad Niswar, S.T., M.InfoTech. Bapak Dr. Amil Ahmad Ilham, S.T., M.IT. dan Bapak Ir. Zahir Zainuddin, M.Sc. selaku dosen penguji yang telah memberikan kritik dan saran yang membangun sehingga laporan tesis ini menjadi lebih baik;
5. Bapak Dr. Ir. Zahir Zainuddin, M.Sc selaku Ketua Program Studi Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah memberikan motivasi, bimbingan, dan semangatnya selama masa perkuliahan penulis;

6. Para sahabat, teman-teman, kakak-kakak dan adik-adik di Laboratorium *Computer Based System* Pascasarjana UNHAS yang telah memberikan begitu banyak bantuan, keceriaan dan pengalaman manis selama proses perkuliahan;
7. Teman-teman Pascasarjana UNHAS Angkatan 1 atas dukungan dan semangat yang diberikan selama ini;
8. Ibu Yuanita serta segenap Staf Departemen Magister Teknik Informatika yang telah banyak membantu penulis selama pengurusan administrasi;
9. Orang-orang terkasih yang tidak sempat dituliskan oleh penulis;

Akhir kata, penulis berharap semoga Allah SWT. Senantiasa berkenan membalas segala kebaikan dari semua pihak yang telah banyak membantu. Semoga Tesis ini dapat memberikan manfaat bagi pengembangan ilmu. Aamiin ya Rabbal Alamin.

Gowa, 20 Februari 2023

Afiyah Rifkha Rahmika

ABSTRAK

Afiyah Rifkha Rahmika. Mekanisme Distribusi Beban Pada *Load Balancing Web Server* Dengan Algoritma *Least Connection* dan *Multi-Agent System* Pada Lingkungan Virtual. (dibimbing oleh **Zulkifli Tahir,** dan **Ady Wahyudi Paundu**).

Kebutuhan pengguna internet terhadap suatu website terus meningkat seiring dengan pengembangan layanan ini yang dilakukan secara terus menerus. Semakin banyak jumlah pengguna internet yang mengakses sebuah website, maka semakin berat beban web server dalam merespon permintaan tersebut sehingga dapat terjadi kondisi *overload* pada *web server*. Hal ini akan menurunkan kinerja website dalam memberikan kepuasan kepada pengguna internet, sehingga dibutuhkan sebuah mekanisme yang mampu menyeimbangkan beban dari web server agar website dapat bekerja secara optimal. Penelitian ini bertujuan untuk mengembangkan mekanisme *load balancing* berdasarkan kondisi *load* dan jumlah koneksi aktif terkecil dari *web server* menggunakan kombinasi algoritma *Least Connection* dan metode Multi-Agent System (LC-MAS). Agen yang diletakkan pada masing-masing *web server* memastikan kondisi *load* dibawah 80% kemudian algoritma menghitung jumlah koneksi yang sedang ditangani untuk selanjutnya dipilih menjadi *web server* yang merespon permintaan selanjutnya. Pengukuran kinerja dilakukan dengan menguji mekanisme LC-MAS dengan mekanisme yang hanya menggunakan algoritma *Least Connection* (LC) dengan mengirimkan sejumlah HTTP *request*. Hasil penelitian menunjukkan mekanisme *load balancing* LC-MAS memperoleh kinerja yang lebih baik dibandingkan dengan algoritma Least Connection (LC), dengan rata-rata waktu respon sebesar 1338.8 ms, 20.07 % error, dan nilai throughput sebesar 125 tps saat menangani 1500 HTTP *request*. Mekanisme *load balancing* dengan LC-MAS mampu mendistribusikan jumlah HTTP request secara lebih merata tanpa menimbulkan kondisi *idle* atau *overload*.

Kata Kunci: Load Balancing, Web Server, Least Connection, Multi-Agent System, Virtualisasi

ABSTRACT

Afiyah Rifkha Rahmika. Web Server Load Balancing Mechanism with Least Connection Algorithm and Multi-Agent System. (Supervised by **Zulkifli Tahir,** and **Ady Wahyudi Paundu**).

Demands for information over the internet become massively increased through the continuous expansion of web applications. Therefore, generating powerful and efficient server architecture for web servers is a must to satisfy internet users and avoid the system being overloaded. Implementing load balancing methods has proven to tackle the problems. A load balancer is needed to manage and distributes the workload equally among the servers to make the website performance better. The main contribution of this research focuses on developing a new mechanism for load balancing to distribute incoming HTTP request in web applications by combining the Least Connection algorithm and Multi-Agent System (LC-MAS). Proposed mechanism will distribute the request based on load condition and fewest number of active connections where existing research did not consider resources condition in developing load balancing mechanisms. This research was tested in two scenarios using 500, 1000, and 1500 requests. The performance of this proposed mechanism is measured through the values of Average Response Time, Throughput, and Error Percentage. The results show that the proposed mechanism (LC-MAS) distributes the workload more equally than LC with average response time for 1500 request is 1338.8 millisecond, 20.07% error, and 125 transactions per second. The LC-MAS makes the web application performance is much better when the request is increased. The LC-MAS helps in the utilization of system resources and improves system robustness.

Keywords: Load Balancing, Web Server, Least Connection, Multi-Agent System, Virtualization

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Nomor Halaman

DAFTAR TABEL

Nomor Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada era digital ini penggunaan internet dikalangan masyarakat sudah menjadi hal yang biasa. Aplikasi internet seperti *website* terlibat secara luas dalam semua aspek kehidupan sehari-hari karena jutaan pengguna menggunakan internet untuk mengakses *website* dari manapun. Konten yang ditampilkan juga sangat beragam dan interaktif sehingga mendorong pengguna internet untuk mengakses *website* setiap saat. Pengguna ini meminta skalabilitas tinggi, ketersediaan, dan keandalan dalam memberikan respon cepat untuk aplikasi *website* yang mereka akses kapan saja[1].

Website seperti layanan *e-commerce* bahkan menerima jutaan koneksi dari pengguna internet secara simultan dalam satu hari karena suatu kondisi dimana *website* tersebut sering dikunjungi. Contohnya pada layanan *e-commerce* yang mengadakan promo besar-besaran di akhir tahun atau pada tanggal tertentu, sistem informasi akademik suatu universitas ketika pendaftaran mahasiswa baru atau saat *deadline* pengumpulan berkas calon mahasiswa, dan pengumpulan tugas mahasiswa melalui *e-learning*. Kondisi ini membuat permintaan akses dari pengguna internet yang harus ditangani oleh *website* tersebut semakin meningkat.

Web server menjadi bagian penting dari infrastruktur internet sebab sebuah aplikasi *website* hanya dapat berjalan di atas sebuah *web server*. Membangun arsitektur *web server* yang baik dan handal sangat penting untuk mengelola trafik dan *request* dalam jumlah yang besar guna melindungi bisnis dari risiko kegagalan *(down)* sistem, kesalahan pemrosesan pesan, kehilangan data transaksional, dan

penurunan performa. Untuk aplikasi *website* yang ramai, *down* sejenak merupakan masalah yang serius karena bisa menghilangkan banyak uang[2].

Semakin banyak jumlah pengguna yang mengakses suatu *web*site, maka semakin berat pula beban sebuah *web server* dalam menerima *request*. Banyaknya jumlah *request* pada suatu *web server* akan mengakibatkan terjadinya *overloading* bahkan kemungkinan *server* akan mengalami *down*. Disisi lain, sebuah *web server* yang handal seharusnya mampu menangani *request* dari pengguna yang cukup besar. Oleh karena itu dibutuhkan penyeimbang beban *server* yang mengatur dan membagi beban kerja *server* secara merata[3].

Teknologi yang dapat diterapkan untuk menyeimbangkan beban pada *server* yaitu teknologi *load balancing*. *Load balancing* adalah teknik untuk mendistribusikan beban trafik dari sebuah layanan pada sekumpulan *server* atau perangkat jaringan secara seimbang ketika ada permintaan dari pengguna. Dengan ini trafik dapat berjalan secara optimal, memaksimalkan *throughput*, memperkecil waktu respon, memanfaatkan *resources* dengan baik sehingga dapat meningkatkan efisiensi hasil yang dibuat oleh *server*. Selain itu jika ada satu *server* yang gagal, layanan tetap dapat berjalan menggunakan *server* yang masih ada karena terdapat banyak *server* yang bekerja untuk melayani *request*[4].

Proses penyeimbangan beban pada sistem *load balancing* memiliki metode dan algoritma tersendiri. Algoritma *load balancing* dibagi menjadi dua jenis yaitu statis dan dinamis. Algoritma statis membagi beban berdasarkan nilai awal yang ditetapkan sebelum *request* dieksekusi oleh *server*. Algoritma dinamis membagi beban berdasarkan kondisi *real* sebuah *server* setelah *request* dieksekusi. Terdapat beberapa parameter tertentu seperti waktu respon, *throughput*, *resources utilization*,

fault tolerance, skalabilitas, dll yang digunakan untuk analisis efisiensi algoritma *load balancing* yang diterapkan[5].

Penelitian [6] menganalisa performansi dari penerapan teknologi *load balancing* menggunakan algoritma *Least Time First Byte* dan *Multi Agent System* (LFB-MAS). Penelitian ini menggunakan kondisi *resources* dari *server backend* dan waktu respon tercepat terhadap *byte* pertama pada data yang menjadi prioritas dalam melakukan *load balancing*. Simulasi dilakukan dengan membandingkan hasil performansi dari algoritma *Weighted Least Connection* (WLC) pada penelitian sebelumnya. Algoritma LFB-MAS mendapatkan hasil yang lebih baik dibandingkan dengan WLC, dengan waktu respon 16.190 ms, *error* 0.00%, dan nilai *throughput* sebesar 43.0 request per second. Akan tetapi penulis menambahkan bahwa nilai *response time* dan *throughput* sistem membutuhkan peningkatan untuk menghasilkan kinerja sistem yang lebih baik dibandingkan sebelumnya.

Waktu respon yang diperoleh jauh lebih lama dibandingkan penelitian sebelumnya dengan algoritma WLC. Hal ini terjadi karena sistem harus mengecek waktu respon *byte* pertama dari setiap *request* yang masuk. LFB-MAS memang menghasilkan waktu respon yang lebih lama dikarenakan LFB-MAS memproses lebih banyak *request* yang dapat diproses sehingga harus menunggu sampai pemrosesan *request* tersebut selesai, sedangkan LFB tanpa agen dan WLC mendapatkan waktu respon lebih singkat karena ketika *request* yang banyak tidak dapat diproses, pemrosesan *request* langsung selesai sehingga nilai error yang dihasilkan lebih tinggi.

Seperti yang diketahui bahwa kebutuhan aplikasi *web* saat ini masih membutuhkan performansi yang lebih tinggi, tidak hanya memberikan waktu respon yang cepat ke pengguna namun sebuah aplikasi *web* dengan performa yang sangat baik juga harus mampu memproses semua permintaan dari pengguna guna meminimalkan nilai *error* dalam menangani permintaan sehingga dibutuhkan mekanisme *load balancing* yang lebih baik.

Berdasarkan permasalahan pada penelitian [6], maka pada penelitian ini peneliti mengusulkan sebuah mekanisme pendistribusian beban pada aplikasi *web* dengan menerapkan teknologi *load balancing* menggunakan algoritma adalah *Least Connection* dan *Multi Agent System* (LC-MAS) untuk meningkatkan kinerja aplikasi web yang lebih baik. Pemilihan algoritma bertujuan untuk menghilangkan proses dimana sistem harus mengecek waktu respon *byte* pertama dari setiap *request* yang masuk yang menyebabkan waktu respon dari *backend server* semakin meningkat.

Algoritma LC mendistribusikan beban kerja secara efektif ke seluruh *server* sesuai dengan kapasitasnya. *Server* dengan kapasitas yang lebih kuat akan memenuhi *request* lebih cepat sehingga pada saat tertentu kemungkinan memiliki jumlah koneksi yang masih diproses jauh lebih sedikit daripada *server* dengan kapasitas yang rendah. Akan tetapi ketika hanya mempertimbangkan jumlah koneksi aktif, penggunaan sumber daya (*resources*) dari setiap *server* tidak dapat dimaksimalkan sehingga dibutuhkan metode untuk memantau kondisi *resources* dari setiap *server*. Penelitian [7] memperlihatkan hasil metrik kinerja beberapa algoritma dalam melakukan *load balancing*. Algoritma LC unggul dalam metrik

performance, resource utilization, dan *throughput* akan tetapi kurang baik pada metik *resources utilization*.

Pada beberapa algoritma seperti algoritma WLC dan Fixed Weighting, bobot beban yang harus ditangani *server* telah ditentukan sebelumnya sedangkan kondisi *real* yang sering terjadi adalah spesifikasi *server* tidak merata sehingga dapat terjadi ketidakseimbangan beban antara *server* dengan spesifikasi rendah dan *server* dengan spesifikasi tinggi. Pemberian bobot pada *server* tidak dapat dikira-kira dalam artian tidak diketahui kapan *server* dengan spesifikasi rendah sedang menangani permintaan yang besar dan kapan *server* dengan spesifikasi tinggi sedang menangani permintaan yang kecil. Agar penggunaan *resources* lebih efisien, maka metode MAS ditambahkan untuk mengecek kondisi setiap *backend server* secara berkala.

Pada umumnya *load balancing* tidak menyediakan metode apapun untuk memonitoring kondisi *resources server*. *Load balancing* dapat diterapkan dengan menggunakan metode *Multi Agent Sistem* (MAS). Metode ini menggunakaan agen-agen yang memiliki kemampuan khusus untuk terus-menerus melakukan tugas yang sebelumnya diberikan dalam suatu lingkungan yang khusus. Agen dalam sistem ini dapat berpindah dari satu *node* menuju *node* yang lainnya secara bebas sesuai dengan tugas yang telah diberikan. Agen bergerak ini dapat diprogram untuk memonitor keadaan *resources* dari *server*. Para agen berkomunikasi melalui *node* dalam kelompok jaringan dalam melakukan tugasnya untuk mendapatkan informasi *resources*. Dengan integrasi metode ini, proses untuk mendapatkan informasi menjadi lebh cepat dan efisien[8].

Penelitian ini juga mengusulkan desain arsitektur mekanisme *load balancing* yang dapat diterapkan pada lingkungan virtual dengan mengatur beban lalu lintas pada jalur koneksi sebuah *node* (mesin virtual/vm). Seperti yang kita ketahui bahwa layanan internet saat ini rata-rata telah berpindah dari manual ke lingkungan virtual. Hal ini bertujuan untuk memudahkan proses pembangunan sistem serta meminimalisir biaya yang harus dikeluarkan ketika harus menyewa perangkat fisik. Lingkungan virtual dapat memudahkan dalam mensetup lingkungan sistem yang diharapkan. Dalam arsitektur ini terdapat satu vm sebagai *server load balancer* dan *server* utama untuk menyimpan informasi dari para agen dan beberapa vm sebagai *web server*. Agen ditempatkan di semua vm dengan tugas yang berbeda. *Agen server* utama bertugas meminta informasi *resources* dari vm yang berperan sebagai *web server*. Informasi ini kemudian dijadikan sebagai acuan untuk *server load balancer* memutuskan kondisi *web server* layak atau tidak untuk menerima *request* berdasarkan parameter yang diatur. Parameter tersebut adalah nilai ambang (*threshold*) dari CPU dan Memori sebesar 80%. Ketika *resources* tidak melebihi *threshold* maka dikategorikan **normal** dan dapat menerima *request*. Ketika *resources* melebihi *threshold* maka dikategorikan ***overload*** dan tidak dapat menerima *request*. Setelah mengecek kondisi *web server*, maka algoritma *least connection* akan menghitung jumlah koneksi aktif terkecil yang sedang ditangani dan *request* tersebut dialihkan ke *web server* yang memenuhi kedua syarat ini. Penelitian ini menggunakan dua skenario pengujian yaitu membandingkan antara *load balancing* menggunakan algoritma *least connection* tanpa *agen* dan *load balancing* menggunakan algoritma *least connection* dan *multi agent system*.

Mekanisme yang disulkan kemudian diuji untuk memperoleh analisa performasi *load balancing* dengan algoritma LC-MAS. Hasil pengujian dari setiap skenario akan dijadikan sebagai bahan analisa kuantitatif dengan mengambil nilai dari beberapa parameter berupa waktu respon, *throughput* (*request/*s), jumlah error, dan *resources utilization*. Penelitian ini berfokus pada aspek efisiensi dari mekanisme yang diusulkan sehingga diharapkan dapat memberikan performansi yang lebih baik dibandingkan dengan metode dan algoritma *load balancing* yang telah diuji pada penelitian sebelumnya dan mampu menjadi referensi baru dalam menerapkan teknologi *load balancing*.

1.2 Rumusan Masalah

Berdasarkan pada latar belakang di atas, maka rumusan masalah pada penelitian ini yaitu

1. Bagaimana merancang dan mengembangkan mekanisme *load balancing* pada *web server* di lingkungan virtual dalam memenuhi kebutuhan dari website yang sampai saat ini masih membutuhkan kinerja yang lebih tinggi?
2. Bagaimana kinerja mekanisme distribusi *load balancing* pada *web server* di lingkungan virtual yang telah dirancang dan dikembangkan?
3. Bagaimana pengaruh kinerja mekanisme distribusi *load balancing* pada *web server* di lingkungan virtual ketika ditambahkan metode Multi-Agent System?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah maka tujuan yang akan dicapai dari penelitian ini:

1. Merancang dan mengembangkan mekanisme *load balancing* pada *web server* di lingkungan virtual dengan kombinasi algoritma *Least Connection* dan metode Multi-Agent System untuk memenuhi kebutuhan dari *website* yang masih membutuhkan kinerja yang lebih tinggi.
2. Melakukan pengujian dan analisis terhadap kinerja mekanisme distribusi *load balancing* pada *web server* di lingkungan virtual yang telah dirancang dan dikembangkan.
3. Mengetahui pengaruh penambahan metode Multi-Agent System berdasarkan kinerja mekanisme distribusi *load balancing*.

1.4 Manfaat Penelitian

Penelitian ini memberikan manfaat berupa pengetahuan (*knowledge*) tentang mekanisme dalam mengelola jumlah request yang besar pada aplikasi *web* dengan teknologi *load balancing*. Pengetahuan ini dapat dijadikan sebagai referensi bagi organisasi atau perusahaan dalam meningkatkan keandalan sistem aplikasi *web* mereka. Hasil analisa kinerja yang dilakukan secara komprehensif dan akurat memberikan manfaat bagi akademisi sebagai literatur ketika akan meneliti pada topik serupa. Manfaat lain dari penelitian ini yaitu tersedianya rekomendasi rancangan mekanisme pendistribusian beban menggunakan teknologi *load balancing* pada aplikasi *web* dengan beberapa skenario.

1.5 Batasan Masalah

Adapun batasan masalah dalam penelitian ini yaitu:

1. Mekanisme *load balancing* berfokus pada pendistribusian permintaan atau *request* pada *website* yang diterapkan di lingkungan virtualisasi.

2. Parameter pengujian untuk menentukan kinerja dari mekanisme yang dikembangkan hanya waktu respon, *throughput*, dan persentasi error yang dihasilkan.
3. *Website* yang digunakan dibangun menggunakan CMS Wordpress dengan konten sederhana.
4. Skenario pengujian dibantu dengan tools Apache Jmeter untuk menciptakan jumlah *request*.

1.6 Sistematika Penulisan

Adapun sistematika penulisan pada penelitian ini yaitu:

Bab I Pendahuluan

Bab ini berisi penjelasan tentang latar belakang yang menjabarkan alasan dilakukannya penelitian terkait *load balancing* berdasarkan peluang penelitian dan uraian penelitian awal tentang *load balancing* yang dilakukan, terkait rumusan masalah, tujuan, manfaat, ruang lingkup serta sistematika penulisan penelitian dibahas pada bagian ini.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan tentang landasan teori yang digunakan dalam penelitian seperti manajemen virtualisasi, teknologi *load balancing*, algoritma *least connection,* metode *Multi-Agent System,* aplikasi website, web server, dan beberapa landasan teori lainnya. Diuraikan pula tentang tinjauan pustaka yang merupakan penjelasan tentang hasil-hasil penelitian sebelumnya yang berkaitan dengan penelitian yang dilakukan. Dalam bab ini juga diuraikan tentang kerangka pemikiran yang merupakan penjelasan tentang kerangka berpikir untuk

memecahkan masalah yang sedang diteliti, termasuk menguraikan objek penelitian serta state of the art dari beberapa penelitian terkait.

Bab III Metodologi Penelitian

Bab ini berisi tentang tahapan penelitian, waktu dan lokasi penelitian, instrumen penelitian, tahap persiapan, gambaran umum sistem, struktur dataset, skenario pengujian.

Bab IV Hasil dan Pembahasan

Bab ini berisi penjabaran hasil penelitian berdasarkan teknik evaluasi kinerja sistem yang digunakan. Pada bagian ini hasil dan pembahasan terbagi atas dua yaitu berdasarkan hasil perancangan mekanisme *load balancing* dan kinerja mekanisme yang dikembangkan. Hasil analisis kinerja mekanisme dirangkum dalam bentuk tabel, grafik dan gambar.

Bab V Kesimpulan dan Saran

Bab V berisi kesimpulan terhadap hasil yang didapatkan dalam penelitian ini, yang merujuk pada rumusan masalah dan saran pengembangan dari penelitian ini untuk menyempurnakan kekurangan-kekurangan atau capaian-capaian yang belum tercapai pada penelitian ini, sehingga kedepannya penelitian yang dilakukan dapat dikembangkan dan bisa memperoleh hasil yang jauh lebih baik.

BAB II
TINJAUAN PUSTKA

II.1. Penelitian Terdahulu

2.1 Virtualisasi

Virtualisasi adalah abstraksi sumber daya komputer dan memisahkan lapisan perangkat lunak dari lapisan perangkat keras serta mengisolasi aplikasi yang sedang berjalan dari perangkat keras yang digunakan. Pengertian virtualisasi dalam lingkungan IT secara mendasar adalah melakukan isolasi terhadap satu sumber daya komputasi dengan yang lainnya. Dengan memisahkan *layer-layer* yang berbeda dalam *logic stack* memungkinkan fleksibilitas yang lebih tinggi karena tidak diperlukan lagi konfigurasi tiap elemen untuk dapat berkerja bersama-sama. Virtualisasi memberikan fleksibilitas yang lebih baik untuk penyediaan sumber daya IT dibandingkan dengan penyediaan di lingkungan non-virtual karena dapat membantu mengoptimalkan pemanfaatan sumber daya dan mengirimkan sumber daya dengan lebih efisien. Pengumpulan sumber daya didukung dalam virtualisasi, di mana sumber daya dapat dikelola secara terpusat untuk meningkatkan fleksibilitas dengan mendukung perubahan dinamis dalam kebutuhan bisnis[9].

Menurut [10], virtualisasi dilakukan dengan menambahkan lapisan virtualisasi yang terdiri dari hypervisor dan monitor mesin virtual yang ditempatkan diantara perangkat keras dan sistem operasi yang menggunakannya. Menggunakan lapisan ini untuk virtualisasi sistem, satu mesin komputer dapat menjalankan beberapa sistem operasi secara bersamaan dalam mesin virtual, di mana sumber daya komputer seperti CPU, memori, dan penyimpanan secara dinamis dipartisi dan dibagi antara mesin virtual yang berbeda.

Pengenalan teknologi virtualisasi terutama berfokus pada tiga bidang: *Hardware Virtualization, Presentation Virtualization, dan Application Virtualization* [11].

a) *Hardware Virtualization*: Konsep dasar dari virtualisasi perangkat keras sangat sederhana yaitu gunakan perangkat lunak untuk membuat mesin virtual (*virtual machine* / VM) yang "meniru" komputer fisik. Dengan menyediakan beberapa VM sekaligus memungkinkan menjalankan beberapa sistem operasi secara bersamaan pada satu mesin fisik. Ketika digunakan pada mesin klien, pendekatan ini sering disebut virtualisasi desktop, sedangkan menggunakannya pada sistem *server* dikenal sebagai virtualisasi *server*.

b) *Presentation Virtualization*: Pendekatan ini memungkinkan pembuatan sesi virtual, masing-masing berinteraksi dengan sistem desktop jarak jauh. Aplikasi yang dijalankan dalam sesi tersebut mengandalkan virtualisasi presentasi untuk memproyeksikan antarmuka pengguna mereka dari jarak jauh. Setiap sesi mungkin hanya menjalankan satu aplikasi, atau mungkin menyajikan penggunanya dengan desktop lengkap yang menawarkan beberapa aplikasi. Dalam kedua kasus tersebut, beberapa sesi virtual dapat menggunakan salinan aplikasi yang sama.

c) *Application Virtualization*: Pendekatan ini memungkinkan pengguna untuk mengakses dan menggunakan aplikasi dari komputer yang terpisah dari komputer tempat aplikasi diinstal. Cara paling umum untuk memvirtualisasikan aplikasi adalah pendekatan berbasis *server*. Artinya, administrator IT menerapkan aplikasi jarak jauh dari *server* di dalam pusat

data organisasi atau melalui layanan *hosting*. Admin IT kemudian menggunakan perangkat lunak virtualisasi aplikasi untuk mengirimkan aplikasi ke *desktop* pengguna atau perangkat lain yang terhubung. Pengguna dapat mengakses dan menggunakan aplikasi seolah-olah itu dipasang secara lokal di komputer mereka, dan tindakan pengguna disampaikan kembali ke *server* untuk dieksekusi.

Penerapan virtualisasi *server* memberikan banyak keuntungan diantaranya, penghematan biaya karena terjadi penurunan anggaran yang dikeluarkan untuk pembelian *server* baru, daya listrik, perangkat pendingin, serta biaya *service* dan *maintenance*, mulai ulang otomatis mesin virtual jika terjadi kegagalan perangkat keras *server* host lengkap (*zero downtime maintenance*), *load balancing*, manajemen penyimpanan, isolasi antar mesin virtual berarti infrastruktur yang mendasari dapat dibagikan antara kelompok yang berbeda (*collabortion*).

2.2 Load Balancing

Load balancing adalah teknik untuk mendistribusikan beban trafik terhadap sebuah layanan yang ada pada sekumpulan *server* atau perangkat jaringan ketika ada permintaan dari pemakai. Dengan *load balancing* trafik dapat berjalan secara optimal, memaksimalkan *throughput* dan memperkecil waktu tanggap dan menghindari *overload* pada salah satu *server*. *Load balancing* dapat diterapkan pada lingkungan fisik dan lingkungan virtual.

Berikut beberapa definisi dari *load balancing*:

a) Menurut [12], *load balancing* adalah metode jaringan komputer untuk mendistribusikan beban kerja di berbagai sumber daya komputasi, seperti komputer, *cluster* komputer, tautan jaringan, unit pemrosesan pusat, atau

drive disk. Teknik load balancing bertujuan untuk mengoptimalkan penggunaan sumber daya, memaksimalkan throughput, meminimalkan waktu respons, dan menghindari *overload* salah satu sumber daya. Menggunakan beberapa komponen dengan *load balancing* sebagai ganti satu komponen dapat meningkatkan keandalan melalui redundansi.

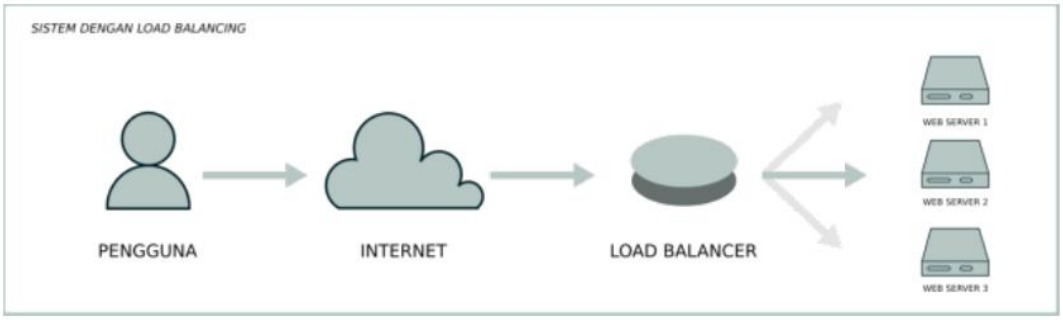
b) Menurut [13], *load balancing* adalah teknik untuk mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara seimbang, agar trafik dapat berjalan optimal, memaksimalkan *throughput*, memperkecil waktu tanggap dan menghindari *overload* pada salah satu jalur koneksi. *Load balancing* digunakan pada saat sebuah *server* telah memiliki jumlah user yang telah melebihi maksimal kapasitasnya.

c) Menurut [14], *load balancing* merupakan teknik yang dapat mendistribusikan beban kerja, jaringan, atau lalu lintas aplikasi merata di beberapa *server*. *Load balancing* mampu meningkatkan fungsi sistem dengan memungkinkan peningkatan keandalan dan kinerja karena menghilangkan terjadinya satu titik kegagalan.

d) Menurut [15], *load balancing* adalah proses membagi beban total ke masing-masing *node* dari sistem terdistribusi untuk melakukan pemanfaatan sumber daya secara lebih cepat dan efisien dengan menghindari situasi dimana beberapa *node* banyak dimuat sementara *node* lain melakukan sedikit pekerjaan. *Load balancing* memastikan semua *node* dalam jaringan akan bekerja kurang lebih sama.

Gambar 1 merupakan gambaran tentang layanan *website* yang menggunakan sistem *load balancing*. *Load balancer* akan mengatur trafik yang datang dan

menentukan *web server* mana yang akan melayani trafik tersebut berdasarkan algoritma yang digunakan.



Gambar 1 Proses Load Balancing

Algoritma *load balancing* berkonsentrasi pada meminimalkan konsumsi sumber daya, meningkatkan skalabilitas, menghindari kemacetan (*bottleneck*), serta mengurangi penyediaan sumber daya yang berlebih. Algoritma *load balancing* pada dasarnya ada 2 jenis tergantung dari metode implementasi yang digunakan [16]:

a) Algoritma static *load balancing* merupakan algoritma *load balancing* yang tidak bergantung pada status sistem saat ini. Pada tahap awal, sebelum permintaan masuk ke *server*, telah diputuskan bahwa di*server* mana permintaan akan dieksekusi. Beberapa contoh dari algoritma statik *load balancing* yaitu Min-min, Round-Robin, OLB dan Max-Min.

b) Algoritma dinamik *load balancing* bekerja dengan cara penyeimbang beban menganalisis statistik beban saat ini dari setiap *server* yang tersedia dan mengeksekusi permintaan pada *server* yang sesuai. Beberapa contoh dari algoritma dinamik *load balancing* yaitu Least Connection, Honey Bee Foraging, Ant Colony Optimization dan Throttled.

Menurut [7] beberapa metrik kualitatif yang dianggap penting untuk *load balancing* dalam mengukur performa algoritma yang digunakan adalah:

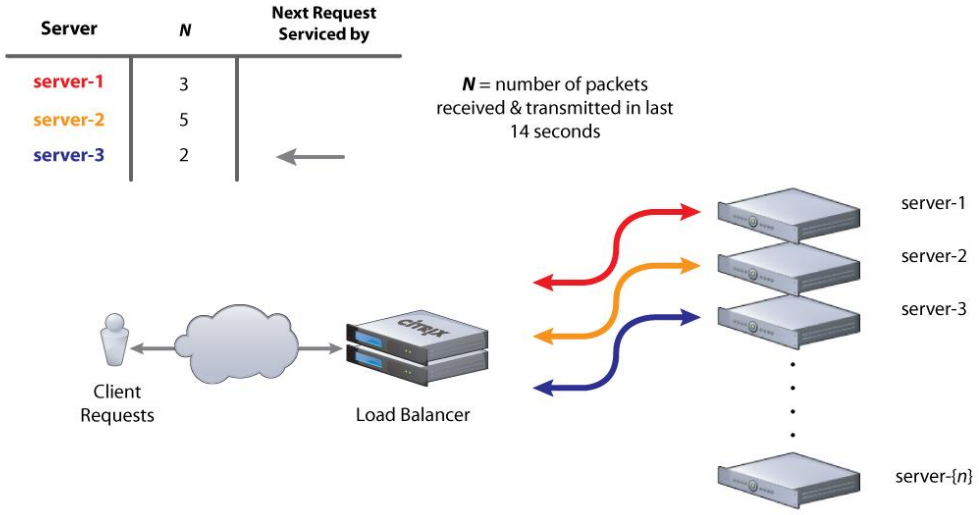
a) Throughput: Jumlah total tugas yang telah menyelesaikan eksekusi disebut *throughput*. *Throughput* yang tinggi diperlukan untuk kinerja sistem yang lebih baik.
b) Fault Tolerant: Kemampuan algoritme untuk bekerja dengan benar dan seragam bahkan dalam kondisi kegagalan di sembarang node dalam sistem.
c) Migration Time: Waktu yang dibutuhkan dalam migrasi atau transfer tugas dari satu mesin ke mesin lain dalam sistem.
d) Response Time: Waktu minimum yang diperlukan sistem terdistribusi yang menjalankan algoritme penyeimbangan beban tertentu untuk merespons.
e) Resource Utilization: Sejauh mana sumber daya sistem digunakan.
f) Scalability: Kemampuan untuk menyeimbangkan beban sistem dengan jumlah simpul yang terbatas. Metrik ini harus ditingkatkan.
g) Performance: Mewakili efektivitas sistem setelah melakukan load balancing. Jika semua di atas parameter terpenuhi secara optimal maka akan sangat tinggi meningkatkan kinerja sistem.

2.3 Algoritma Least Connection (LC)

Algoritma *load balancing least connection* bekerja dengan mengarahkan *request* menuju *server* yang paling sedikit memiliki koneksi yang sedang tersambung dengan terus mengupdate catatan koneksi tersambung secara *real time*, sehingga dapat diketahui *server* mana yang sedang menangani banyak *request* dan *server* mana yang tidak. Algoritma dapat bekerja dengan baik pada *web server* dengan spesifikasi *hardware* yang sama.

Penjadwalan ini termasuk salah satu algoritma penjadwalan dinamik, karena memerlukan perhitungan koneksi aktif untuk masing-masing *real server* secara

dinamik. Metode penjadwalan ini baik digunakan untuk melancarkan pendistribusian ketika *request* yang datang sangat banyak. Ilustrasi pendistribusian beban *request* dalam algoritma ini dapat dilihat pada Gambar II.2.



Gambar 2 Distribusi Beban Pada Algoritma Least Connection

Ilustrasi diatas menggambarkan *cluster server* yang saling terkoneksi untuk mengelola permintaan atau *request* dari pengguna internet yang mengakses sebuah layanan. *Server* tersebut menangani jumlah permintaan yang dikirim dan diterima sebanyak (N) dalam waktu 14 detik terakhir. *Server-1* sedang menangani 3 *request*, *Server-2* sedang menangani 5 *request*, *Server-3* sedang menangani 2 *request*. Algoritma LC kemudian menghitung jumlah koneksi aktif yang sedang ditangani oleh *server* untuk kemudian menentukan *request* selanjutnya akan diteruskan ke *server* mana. Dari gambar dilihat bahwa *request* selanjutnya diteruskan ke *Server-2* yang dalam waktu tersebut menangani *request* yang paling sedikit.

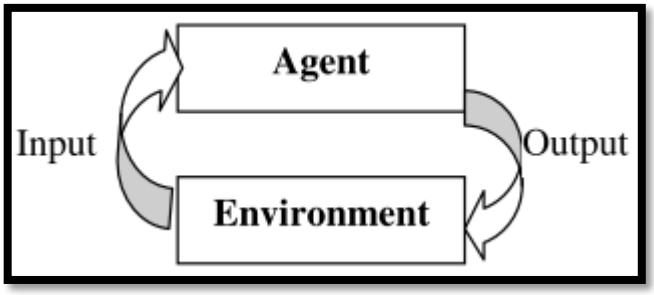
Penelitian [17] menjelaskan bahwa *server* dengan spesifikasi *hardware* yang bervariasi, kerja algoritma ini bisa menjadi tidak optimal, disebabkan oleh adanya TCP_WAIT_TIME. Kondisi *wait time* menyebabkan koneksi yang belum diproses disimpan dalam antrian. *Server* dengan spesifikasi rendah akan mengalami waktu

proses yang lebih lama dibandingkan dengan *server* spesifikasi tinggi, sehingga menyebabkan waktu *reply* yang tinggi untuk koneksi yang dilayani oleh *server* dengan spesifikasi rendah.

Algoritma ini menetapkan koneksi pengguna ke setiap server yang terhubung dengan pengguna. Arsitektur keseimbangan beban dirancang berdasarkan jumlah koneksi pengguna. Server distributor terhubung ke permintaan HTTP, dan setiap pengguna baru mengirimkan permintaan HTTP yang akan ditetapkan ke jumlah minimum koneksi dengan server [18] [14].

2.4 Multi-Agent System (MAS)

Multi-Agent System dibangun dari agent-agent yang cerdas (*intelligent agents*). *Intelligent agent* adalah sebuah sistem komputer yang independen, yang mampu melakukan *action* terhadap kebutuhan lingkungannya, khususnya kebutuhan dirinya sendiri, sehingga sebuah *agent* dapat menyelesaikan pekerjaan yang didelegasikan kepadanya secara lebih baik (*delegation*), serta mampu secara cerdas mengelola pengetahuannya (*intelligence*), juga memberikan kontribusi pada lingkungannya (*interconnection*). Setiap *agent* memiliki kemampuan dapat menerima informasi dari luar, yang selanjutnya dapat memberikan aksi ke lingkungan [19]. Sebuah *agent* mampu menerima masukan dari lingkungan di sekitarnya dan memberikan keluaran sesuai dengan masukan tersebut secara otomatis. Interaksi keduanya dapat dilihat pada Gambar 2. Agent memiliki kemampuan atau karateristik seperti kooperatif, mandiri, dapat berkomunikasi antar agent, dan bergerak (mobile). Agent dapat berpindah dari satu node ke node yang lain dengan bebas sesuai dengan tugas yang diberikan.



Gambar 3 Interaksi Agent dan Lingkungannya

Agent tidak selamanya beroperasi sendirian tapi kadang-kadang bisa melibatkan *agent-agent* lain sehingga disebut *multi-agent system*. Dalam interaksi antar *agent* dibutuhkan sebuah protokol yang menangani interaksi tersebut. Contoh protokol komunikasi dua *agent* yang yaitu:

- Menawarkan sebuah aksi yang harus dijalankan
- Menerima tawaran aksi tersebut
- Menolak tawaran aksi tersebut

Berdasarkan contoh sederhana dari protokol yang ada di atas, maka interaksi antara *agent* A dan *agent* B bisa berupa:

- *Agent* A menawarkan sebuah aksi yang harus dijalankan oleh *Agent* B
- *Agent* B mengevaluasi tawaran dari *Agent* A
- *Agent* B menerima atau menolak tawaran tersebut

Multi-Agent System (MAS) merupakan sebuah perangkat lunak berbasis sistem terdistribusi menggunakan agent dengan kemampuan khusus dan *autonomous* (mandiri) pada sebuah jaringan [20]. Sistem *multi-agent* dibangun atas 3 kategori yang umum, yaitu, *provider agent, service requester agent*, dan *middle agent. Provider agent* (penyedia layanan) seperti pencarian informasi atau mememcahkan masalah yang spesifik, *Requester Agent* adalah agent yang membutuhkan layanan tersebut dari *provider agent*, dan *Middle Agent* digunakan

untuk mencari kecocokan antar agent. Contoh sistem *multi-agent* keuangan, Pemantauan perawatan kesehatan berbasis agen seluler dirancang untuk memberi tahu penyedia perawatan yang bertanggung jawab tentang kelainan secara otomatis.

II.2. Metode Penyelesaian Masalah

2.5 State of the Art Penelitian

Penelitian-penelitian terkait yang telah dilakukan sebelumnya dapat dilihat pada Tabel 1 dibawah ini.

Tabel 1 State of the Art

No	Judul Karya Ilmiah, Nama, Tahun Terbit, dan Penerbit	Objek Dan Permasalahan	Metode Penyelesaian	Kinerja
1.	**Topik yang diusulkan:** Mekanisme Distribusi Beban Pada Load Balancing Web Server Dengan Algoritma Least Connection dan Multi-Agent System Pada Lingkungan Virtual	**Objek:** Web Server dan Algoritma Load Balancing **Permasalahan:** Bagaimana merancang dan menguji mekanisme pendistribusian beban dengan teknologi *load balancing* untuk memenuhi kebutuhan dari aplikasi web yang sampai saat ini masih membutuhkan performansi yang lebih tinggi?	Algoritma Least Connection dan Multi-Agent System	Distribusi permintaan yang masuk dibagi dengan hanya mengecek kondisi resources server dan koneksi aktif terkecil yang ditangani server pada saat itu sehingga diharapkan mampu menghasilkan performansi yang lebih baik dibandingkan metode penyelesaian pada penelitian terdahulu, khususnya waktu respon dan nilai error
2.	**Judul:** Round-robin Algorithm in HAProxy and Nginx Load Balancing Performance Evaluation a Review **Penulis:**	**Objek:** Web Server dan Algoritma Load Balancing **Permasalahan:** Bagaimana evaluasi kinerja load balancer yang	Round Robin Least Connection HAProxy NGINX NGINX KA	HAProxy menunjukkan kinerja yang lebih baik di antara NGINX dalam konfigurasi default dengan menerapkan algoritma Round-Robin. Jika

	Luthfian Hadi. Pramono, Robby Cokro Buwono, Yanuar Galih Waskito **Tahun:** 2018 **Penerbit:** Journal of Chemical Information and Modeling	menggunakan algoritma Round Robin dan Least Connection?		konfigurasi Keep Alive pada NGINX aktif, maka NGINX memiliki performa yang sangat baik dan lebih cepat dari yang lain. Algoritma Least Connection menghasilkan hasil yang lebih baik untuk beberapa pengujian, seperti permintaan per detik dan kecepatan transfer (KB/detik).
3	**Judul:** Performance comparisons of web server load balancing algorithms on HAProxy and Heartbeat **Penulis:** Agung B. Prasetijo, Eko D. Widianto, Ersya T. Hidayatullah **Tahun:** 2017 **Penerbit:** IEEE International Conference on Information Technology, Computer, and Electrical Engineering, ICITACEE	**Objek:** Web Server dan Load Balancer **Permasalahan:** Bagaimana merancang sebuah sistem yang dapat membagi beban secara merata dengan memastikan ketersediaan web server ketika server utama mengalami down?	Least Connection, Round Robin, Source, Heartbeat, HAProxy	Algoritma Least Connection terbukti berperan lebih baik dalam hal throughput, waktu respons, jumlah koneksi, dan mengurangi jumlah total permintaan yang gagal. Meskipun, Round Robin adalah yang terbaik ketika sistem menghadapi lalu lintas tinggi.
4.	**Judul:** Web Server Farm Design Using Personal Computer (PC) Desktop **Penulis:** Iqsyahiro Kresna A, Yusep Rosmansyah **Tahun:** 2018 **Penerbit:** IEEE International Conference on Information Technology	**Objek:** Web Server Farm atau Clustering Web Server dan Load Balancer **Permasalahan:** Bagaimana merancang arsitektur cluster web server pada sebuah PC Desktop dan bagaimana performansi yang dihasilkan dari penerapan	Least Connection, Round Robin	Hasil penelitian sangat baik dalam penerapan menggunakan hardware dengan spesifikasi yang tinggi. Dari hasil pengujian Load Balancer, performansi yang dihasilkan oleh algoritma Least Connection lebih baik daripada algoritma Round Robin. Rata-rata waktu Downtime

	and Electrical Engineering (ICITEE)	algoritma load balancing pada PC Desktop		pada pengujian High Availability atau failover adalah 6,587 detik.
5.	**Judul:** Load Balancing Evaluation Tools for a Private Cloud: A Comparative Study **Penulis:** Sahand Kh. Saeid, Tara Ali Yahiya **Tahun:** 2018 **Penerbit:** The Scientific Journal of Koya University Volume VI, No 2(2018)	**Objek:** Penerapan load balancing pada web server di lingkungan private cloud. **Permasalahan:** Bagaimana mengatasi kemacetan lalu lintas pada server yang menyediakan layanan di cloud?	Least Connection, NGINX, HAProxy	Hasil penelitian sangat baik karena diterapkan pada private cloud dan memiliki banyak skenario pengujian. Waktu respons HAProxy sebesar 3.124 s sedangkan NGINX sebesar 3.297 s terhadap permintaan yang masuk dan HAProxy dapat menerima lebih banyak permintaan dengan status beban berat (3201 *request*). Namun CPU Utilization NGINX hanya 20% dan lebih rendah daripada HAProxy yang hampir mencapai 50%
6.	**Judul :** Load Balancing Algorithms Round-Robin (RR), Least-Connection and Least Loaded Efficiency **Penulis:** Dr. Mustafa ElGili Mustafa **Tahun:** 2017 **Penerbit:** International Journal of Computer and Information Technology	**Objek:** Penerapan teknologi load balancing pada 8 HTTP Server **Permasalahan:** Bagaimana meningkatkan efisiensi pendistribusian trafik dengan load balancing pada HTTP Server yang disimulasikan dengan Opnet?	Round Robin, Least Connection, Least Loaded	Hasil penelitian kurang baik karena masih berupa simulasi. Penggunaan CPU pada web server akan lebih tinggi jika menggunakan algoritma Least Loaded dibandingkan dengan dua algoritma lainnya yang memiliki tingkat konsumsi CPU yang sama. Round robin mendistribusikan beban kira-kira sama kecuali server pertama yang menerima jumlah permintaan

				tertinggi dibandingkan dengan 7 server lainnya. Algoritma Least Connection membagi beban paling adil daripada dua algoritma lainnya.
7.	**Judul :** Model of load balancing using reliable algorithm with Multi-Agent System **Penulis:** Muhammad Faizal Afriansyah, Maman Somantri, Munawar Agus Riyadi **Tahun:** 2016 **Penerbit:** International Conference on Electrical Engineering, Computer Science and Informatics	**Objek:** Penerapan teknologi load balancing pada web server di lingkungan virtual **Permasalahan:** Bagaimana menangani masalah peningkatan aktivitas trafik jaringan yang berpengaruh pada beban kerja sistem web pada skala besar?	Least Time First Byte, Multi Agent System, NGINX	Hasil penelitian kurang baik karena salah satu parameter penting load balancing yaitu respon time pada penetilian ini masih cukup besar.
8.	**Judul :** Web Server Load Balancing Based On Memory Utilization Using Docker Swarm **Penulis:** Mochamad Rexa Mei Bella, Mahendra Data, Widhi Yahya **Tahun:** 2018 **Penerbit:** International Conference on Sustainable Information Engineering and Technology	**Objek:** Load Balancing pada Docker **Permasalahan:** Bagaimana mendistribusikan beban server pada Docker Swarm berdasarkan resources utilization agar terhindar dari distribusi beban yang tidak merata?	Memory Utilization, Docker Swarm	Hasil penelitian menunjukkan nilai CPU dan Memory utilization yang dihasilkan lebih tinggi pada node worker 1 namun perbedaannya tidak jauh dari nilai ambang batas yang ditentukan. Hasil ini menunjukkan bahwa masih diperlukan perbaikan untuk memperoleh nilai utilization dibawah ambang batas
9.	**Judul :** Methodology for Load Balancing in Multi-Agent System Using SPE Approach	**Objek:** Algoritma Load Balancing pada SPE (Software	Multi-Agent System, Round Robin, Random,	Hasil penelitian menunjukkan nilai respon time dari algoritma yang diusulkan mencapai

	Penulis: S. Ajitha **Tahun:** 2021 **Penerbit:** Journal of Security Issues and Privacy Concerns in Industry 4.0 Applications	Performance Engineering) **Permasalahan:** Bagaimana menyeimbangkan beban kerja agen pada MAS dengan mengembangkan algoritma pada JADE dan NetLogo?	Mixed Selection	nilai terbaiknya saat diimplementasikan pada JADE, begitupun nilai server utilization menunjukkan hasil yang lebih baik dibandingkan algoritma yang telah ada.
10.	**Judul :** Multi-Agent Cognitive System for Optimal Solution Search **Penulis:** Victor, Ababii Viorica, Sudacevschi Silvia, Munteanu Dimitrie, Bordian Dmitri, Calugari Ana, Nistiriuc Sergiu, Dilevschi **Tahun:** 2018 **Penerbit:** International Conference on DEVELOPMENT AND APPLICATION SYSTEMS	**Objek:** Optimasi pada teori permainan **Permasalahan:** Bagaimana menemukan solusi paling optimal dari sebuah Multi-Agent System berdasarkan konsep Nash Equilibrium?	Multi-Agent System, Nash Equilibrium	Hasil penelitian menunjukkan Semakin banyak agen yang terlibat dalam aksi di lingkungan, semakin cepat dan semakin nyata pengaruh ini.

Tabel diatas merupakan sebuah ringakasan yang memuat beberapa penelitian terdahulu yang memiliki keterkaitan dengan penelitian yang akan dilakukan. Tabel diatas berupa judul, penulis, penerbit, tahun, objek dan permasalahan, metode penyelesaian, kinerja dan korelasi.

Berdasarkan uraian beberapa keaslian penelitian pada Tabel II.1, maka pada penelitian ini akan merancang sebuah mekanisme load balancing untuk menyeimbangkan beban permintaan (*request*) pada aplikasi web. Dalam penerapannya, *load balancer* akan menggunakan sebuah algoritma untuk membagi

beban secara merata dan dikombinasikan dengan metode yang dapat mengecek kondisi *resources server* secara berkala guna mencapai pemanfaatan *resources* yang maksimal. Algoritma dan metode yang digunakan adalah Least Connection dan Multi-Agent System.

Dengan usulan rancangan mekanisme ini, *request* yang masuk dibagi hanya dengan mengecek kondisi *resources server* dan koneksi aktif terkecil yang ditangani server pada saat itu sehingga sistem tidak perlu lagi menghitung jumlah *byte* data yang masuk seperti yang dilakukan pada penelitian sebelumnya (LFB-MAS) dan diharapkan mampu menghasilkan performansi yang lebih baik, khususnya waktu respon dan nilai *error*. Penelitian ini diuji pada lingkungan virtual dengan menggunakan *software* manajemen virtualisasi. Penelitian ini memiliki dua skenario pengujian dengan tiga kali percobaan yang berbeda. Hasil pengujian dari setiap skenario akan dianalisa untuk mengetahui sejauh mana kinerja yang dihasilkan oleh rancangan mekanisme yang diusulkan.

2.6 Tingkat Keaslian (Level Orisinalitas) Topik Penelitian yang diusulkan

Pengembangan topik penelitian yang diusukan ini dapat dilihat dari Tabel 2. Tabel ini berisi kesesuaian antara referensi pada tabel State of the Art dengan metode penyelesaian yang digunakan dalam penelitian.

Tabel 2 Matriks Metode Penyelesaian

No. Referensi dari Tabel II.1	Metode Penyelesaian									
	a	b	c	d	e	f	g	h	i	j
1 (Topik yang diusulkan)				✓		✓				
2		✓		✓			✓			
3	✓	✓		✓						
4				✓						
5		✓	✓	✓						
6					✓	✓				
7					✓	✓				
8										✓
9		✓				✓		✓	✓	
10						✓				

Keterangan:

Nomor Referensi:

1. Analisis Performansi Load Balancing Web Server Dengan Algoritma Least Connections dan Multi-Agent System Pada Lingkungan Virtual
2. Round-robin Algorithm in HAProxy and Nginx Load Balancing Performance Evaluation a Review
3. Performance comparisons of web server load balancing algorithms on HAProxy and Heartbeat
4. Web Server Farm Design Using Personal Computer (PC) Desktop
5. Load Balancing Evaluation Tools for a Private Cloud: A Comparative Study
6. Load Balancing Algorithms Round-Robin (RR), Least-Connection and Least Loaded Efficiency
7. Model of load balancing using reliable algorithm with Multi-Agent System
8. Web Server Load Balancing Based On Memory Utilization Using Docker Swarm
9. Methodology for Load Balancing in Multi-Agent System Using SPE Approach

10. Multi-Agent Cognitive System for Optimal Solution Search

Metode Penyelesian:

a. Heartbeat
b. Round Robin
c. Least Loaded
d. Least Connection
e. Least Time First Byte
f. Multi Agent System
g. Source
h. Random Selection
i. Mixed Selection
j. Memory Utilization

Berdasarkan pengembangan topik penelitian tentang load balancing pada web server yang dapat dilihat dari Tabel 2 maka topik penelitian ini memenuhi level keaslian atau level orisinalitas tingkat 4 sesuai dengan tingkat keaslian dari pedoman penulisan proposal yang dikeluarkan oleh Program Studi Magister Teknik Informatika UNHAS dilihat pada Tabel 3. Topik penelitian yang diusulkan memenuhi tingkat keaslian 4 karena metode penyelesaian yang diusulkan yaitu algoritma Least Connection dan metode Multi-Agent System belum digunakan untuk menyelesaikan masalah kelebihan beban pada web server. Metode yang diusulkan juga dikategorikan sebagai metode penyelesaian baru karena menggabungkan beberapa metode lama (poin b) untuk menyelesaikan objek dan permasalahan yang sama.

Tabel 3 Level Orisinalitas Berdasarkan Standar Proposal Program Studi Magister Teknik Informatika Universitas Hasanuddin

Level orisinalitas	**Objek penelitian**	**Jenis permasalahan**	**Metode penyelesaian masalah**	**Status orisinal proposal/penelitan**
1	xxxxx	Tidak jelas, sangat minim untuk level S2	xxxxxx	Orisinalitas tidak ada
2	Terpakai (sudah digunakan sebelumnya)	Serupa dengan objek yang sama	Sudah digunakan untuk objek dan jenis permasalahan sebelumnya	Sangat tidak orisinal (replikasi)
3	Baru	Serupa dengan sebelumnya	Sama untuk masalah yang serupa	Kurang
4	Terpakai (sudah digunakan sebelumnya)	Serupa dengan objek yang sama	Lain yang tersedia (belum digunakan untuk masalah yang sama)	Minimalis
5	Baru	Serupa dengan sebelumnya	Lain yang tersedia	Minimalis
6	Terpakai	Baru	Tersedia	Orisinal
7	Baru	Baru	Tersedia	Orisinal
8	Terpakai	Serupa	Baru	Orisinal+Novelty
9	Baru	Serupa	Baru	Orisinal+Novelty
10	Terpakai	Baru	Baru	Sangat Orisinal+Novelty
11	Baru	Baru	Baru	Sangat Orisinal+Novelty

Kategori metode penyelesaian baru:

a. Modifikasi dari metode yang lama
b. Penggabungan beberapa metode lama
c. Metode yang baru dikembangkan dan atau belum pernah digunakan sebelumnya
d. Selalu menghasilkan keluaran yang lebih baik

C. Target Hipotesis Penelitian

Target hipotesis pada penelitian ini adalah nilai dari metrik parameter performansi yang dihasilkan aplikasi web setelah diterapkan mekanisme yang

diusulkan dapat lebih baik. Penelitian ini ditargetkan memiliki nilai-nilai parameter yang lebih baik dibandingkan penelitian terdahulu, seperti nilai waktu respon yang lebih kecil, nilai *throughput* yang lebih besar, nilai *error* yang dihasilkan sangat minim sehingga jumlah *request* yang berhasil ditangani oleh sistem dapat lebih banyak dan penggunaan *resources* dari setiap *server* memiliki nilai yang stabil atau dengan kata lain tidak *overload* atau *underload*. Penelitian ini juga ditargetkan menghasilkan analisis performansi yang detail dan akurat sehingga mampu menjadi referensi dalam implementasi *real* dilapangan.

D. Kerangka Pikir

Kerangka pikir dapat dilihat pada Tabel 4 yang menjelaskan alur penelitian yang akan dilakukan.

Tabel 4 Kerangka Pikir Penelitian

Masalah • Kinerja website saat banyak pengguna internet yang mengunjunginya semakin menurun karena kelebihan beban kerja pada web server • Aplikasi web sampai saat ini masih membutuhkan kinerja yang lebih tinggi • Mekanisme *load balancing* yang terdahulu kurang memperhatikan aspek *resource* pada *server*
Solusi Merancang dan mengembangkan mekanisme *load balancing* pada lingkungan virtualisasi dengan memperhatikan aspek *resource* dalam mendistribusikan beban kerja *web server* untuk mencapai kinerja maksimum pada website.
Metode Penyelesaian Kombinasi algoritma *Least Connection* dan *Multi-Agent System* dalam mendistribusikan *request* dari pengguna internet berdasarkan kondisi *load* dan jumlah koneksi aktif terkecil yang sedang ditangani *web server*
Hasil Mekanisme *load balancing* dapat mendistribusikan beban *web server* secara merata tanpa menyebabkan kondisi *overload* atau *idle* pada *web server* yang membuat kinerja *website* semakin tinggi.