PENERAPAN METODE *ROUND ROBIN* DALAM EFEKTIVITAS *LOAD BALANCER* PADA PENDAFTARAN BEASISWA DI UNIVERSITAS MUHAMMADIYAH MAKASSAR

PROPOSAL SKRIPSI

Diajukan Sebagai Salah Satu Syarat Untuk Mendapatkan

Gelar Sarjana Komputer (S.Kom) Program Studi Informatika

ISMI SALSABILA

105841101720

PRODI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH MAKASSAR

2024

UNIVERSITAS MUHAMMADIYAH MAKASSAR

FAKULTAS TEKNIK

GEDUNG MENARA IQRA LT. 3

Jl. Sultan Alauddin No. 259 Telp. (0411) 866 972 Fax (0411) 865 588 Makassar 90221
Website: www.unismuh.ac.id, e_mail: unismuh@gmail.com
Website: http://teknik.unismuh.makassar.ac.id

بسم الله الرحمن الرحيم

PENGESAHAN

Skripsi atas nama ISMI SALSABILA dengan nomor induk Mahasiswa 105 84 11017 20, dinyatakan diterima dan disahkan oleh Panitia Ujian Tugas Akhir/Skripsi sesuai dengan Surat Keputusan Dekan Fakultas Teknik Universitas Muhammadiyah Makassar Nomor : 300/05/A.5-II/VIII/46/2024 sebagai salah satu syarat guna memperoleh gelar Sarjana Komputer pada Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Makassar pada hari Senin tanggal 31 Agustus 2024.

Panitia Ujian :

Makassar, 26 Safar 1446 H / 31 Agustus 2024 M

1. Pengawas Umum
   a. Rektor Universitas Muhammadiyah Makassar
      Dr. Ir. H. Abd. Rakhim Nanda, S.T., M.T., IPU. .....
   b. Dekan Fakultas Teknik Universitas Hasanuddin
      Prof. Dr. Eng. Muhammad Isran Ramli, S.T., MT. .....
2. Penguji
   a. Ketua : Dr. Ir. Zahir Zainuddin, M.Sc. .....
   b. Sekertaris : Fahrim Irhamna Rahman S.Kom., MT. .....
3. Anggota : 1. Lukman Anas, S.Kom., MT. .....
   2. Titin Wahyuni, S.Pd., M.T. .....
   3. Desi Anggreani, S.Kom.,M.T. .....

Mengetahui :

Pembimbing I

Muhyiddin A M Hayat, S.Kom., MT.

Pembimbing II

Lukman, S.Kom., MT.

Dekan

Dr. Ir. Hj. Nurnawaty, S.T., MT., IPM.
NBM. 795 108

UNIVERSITAS MUHAMMADIYAH MAKASSAR
FAKULTAS TEKNIK
GEDUNG MENARA IQRA LT. 3
Jl. Sultan Alauddin No. 259 Telp. (0411) 866 972 Fax (0411) 865 588 Makassar 90221
Website: www.unismuh.ac.id, e_mail: unismuh@gmail.com
Website: http://teknik.unismuh.makassar.ac.id

بسم الله الرحمن الرحيم

HALAMAN PENGESAHAN

Tugas Akhir ini diajukan untuk memenuhi syarat ujian guna memperoleh gelar Sarjana Informatika (S.Kom) Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Makassar.

Judul Skripsi : **PENERAPAN METODE ROUND ROBIN DALAM EFEKTIVITAS LOAD BALANCER PADA PENDAFTARAN BEASISWA DI UNIVERSITAS MUHAMMADIYAH MAKASSAR**

Nama : ISMI SALSABILA
Stambuk : 1058411017 20

Makassar, 31 Agustus 2024

Telah Diperiksa dan Disetujui
Oleh Dosen Pembimbing:

Pembimbing I	Pembimbing II
Muhyiddin A.M Hayat, S.Kom., MT.	Lukman, S.Kom., MT.

Mengetahui,
Ketua Program Studi Informatika

Muhyiddin A.M Hayat, S.Kom., MT.
NBM. 1504 577

Abstrak

Penelitian ini bertujuan untuk menganalisis efektivitas metode *Round Robin* dalam *load balancing* guna meningkatkan kinerja sistem pendaftaran beasiswa di Universitas Muhammadiyah Makassar. Masalah utama yang dihadapi adalah lonjakan akses selama periode pendaftaran, yang menyebabkan kinerja server menurun dan waktu respons yang lambat. Penerapan *load balancing* bertujuan untuk mendistribusikan beban secara merata ke beberapa server, sehingga sistem dapat menangani lonjakan akses secara lebih efisien. Metode yang digunakan dalam penelitian ini melibatkan instalasi dan konfigurasi perangkat lunak *Nginx* sebagai *load balancer* dengan menggunakan metode *Round Robin*. Kinerja sistem diuji menggunakan alat uji beban *Locust* untuk mensimulasikan ribuan pengguna secara bersamaan. Pengujian dilakukan pada dua skenario, yaitu dengan dan tanpa *load balancer*, untuk membandingkan kinerja dari kedua konfigurasi tersebut. Hasil penelitian menunjukkan bahwa penerapan *load balancing* dengan metode *Round Robin* berhasil meningkatkan stabilitas dan efisiensi sistem, dengan waktu respons yang lebih cepat dan konsisten dibandingkan tanpa *load balancer*. Meskipun terdapat sedikit penurunan throughput, sistem dengan *load balancer* menunjukkan distribusi beban yang lebih merata dan penurunan waktu respons yang signifikan. Oleh karena itu, metode ini terbukti efektif dalam mengatasi masalah kinerja sistem selama periode lonjakan akses.

Kata Kunci: *Round Robin*, *Load Balancer*, Pendaftaran Beasiswa, Kinerja Sistem, Nginx, Locust

Abstract

This research aims to analyze the effectiveness of the Round Robin method in load balancing to improve the performance of the scholarship registration system at Muhammadiyah University of Makassar. The main issue faced is the surge in access during the registration period, which causes server performance to degrade and response times to slow down. The application of load balancing aims to evenly distribute the workload across several servers, allowing the system to handle access surges more efficiently. The method used in this research involves the installation and configuration of Nginx software as a load balancer using the Round Robin method. System performance was tested using the Locust load testing tool to simulate thousands of users concurrently. The tests were conducted under two scenarios: with and without load balancing, to compare the performance of both configurations. The results of the study show that the application of load balancing with the Round Robin method successfully improved system stability and efficiency, with faster and more consistent response times compared to a system without a load balancer. Although there was a slight decrease in throughput, the system with the load balancer demonstrated better load distribution and significantly reduced response times. Therefore, this method proved effective in addressing system performance issues during periods of high access demand.

***Keywords**: Round Robin, Load Balancer, Scholarship Registration, System Performance, Nginx, Locust*

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Dengan mengucapkan bismillahirrahmanirrahim, saya memulai penulisan kata pengantar ini sebagai ungkapan rasa syukur kepada Allah SWT, yang telah memberikan kemudahan dan kekuatan sehingga saya dapat menyelesaikan skripsi ini dengan judul "**Penerapan Metode *Round Robin* dalam Efektivitas *Load Balancer* pada Pendaftaran Beasiswa di Universitas Muhammadiyah Makassar**".

Skripsi ini merupakan buah dari proses pembelajaran yang panjang dan penuh tantangan di **UNIVERSITAS MUHAMMADIYAH MAKASSAR**, yang tidak mungkin terwujud tanpa dukungan dari berbagai pihak. Oleh karena itu, izinkan saya untuk menyampaikan terima kasih kepada:

1. Ayahanda **Hasbi Hasan**, beliau yang menjadi inti tulang punggung keluarga. Meskipun beliau tidak sempat merasakan pendidikan hingga bangku perkuliahan, namun beliau mampu mendidik penulis menjadi laki-laki yang kuat dan tegar dalam segala rintangan, hingga penulis mampu menyelesaikan tugas akhir ini.
2. Ibunda **Mardiana**, pintu surgaku. Beliau sangat berperan penting dalam menyelesaikan program studi penulis. Beliau juga memang tidak sempat merasakan pendidikan hingga bangku perkuliahan, namun gigih dalam memanjatkan doa yang selalu beliau berikan yang tiada henti meminta kepada Tuhan Yang Maha Esa, hingga penulis mampu menyelesaikan akhir ini.
3. Terima kasih yang tulus kepada bapak **Muhyiddin A M Hayat, S.Kom, M.T.** dan Pak **Lukman, SKM, S.Kom, MT** yang telah menjadi pembimbing yang luar biasa selama proses penulisan tugas akhir ini. Bimbingan, kesabaran, dan wawasan yang diberikan telah menjadi penerang dalam menuntun saya menyelesaikan penelitian ini. Tanpa arahan yang berharga beliau, skripsi ini tidak akan dapat terwujud.

Saya berharap skripsi ini dapat memberikan kontribusi yang berarti bagi pengembangan ilmu INFORMATIKA dan menjadi sumber referensi yang berguna bagi peneliti lainnya.

Akhir kata, semoga skripsi ini dapat diterima dan memenuhi salah satu syarat untuk mencapai gelar sarjana di FAKULTAS TEKNIK

Makassar, 15 Juli 2024

ISMI SALSABILA

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR ISTILAH

Round Robin	Metode penjadwalan atau distribusi beban di mana setiap permintaan atau tugas didistribusikan secara merata ke semua server atau prosesor yang tersedia, dalam urutan yang berulang.
Load Balancer	Perangkat atau perangkat lunak yang mendistribusikan beban jaringan atau aplikasi di beberapa server untuk memastikan tidak ada server yang kewalahan dan untuk meningkatkan ketersediaan serta keandalan layanan.
Nginx	Perangkat lunak server web dan reverse proxy yang juga berfungsi sebagai load balancer, HTTP cache, dan media streaming server. Nginx dikenal karena kinerja dan stabilitasnya yang tinggi.
Overload	Situasi di mana server, jaringan, atau aplikasi menerima lebih banyak permintaan daripada yang dapat ditangani, mengakibatkan penurunan kinerja atau bahkan kegagalan layanan.
Website	Kumpulan halaman web yang dapat diakses melalui internet, yang biasanya berisi teks, gambar, video, dan lainnya, diakses menggunakan browser.
Server	Komputer atau sistem yang menyediakan

	sumber daya, data, layanan, atau program kepada komputer lain, yang disebut klien, melalui jaringan.
Database	Sistem terorganisir untuk menyimpan, mengelola, dan mengambil data secara efisien. Digunakan oleh server untuk menyimpan data aplikasi.
Tools	Perangkat lunak atau program yang digunakan untuk melakukan tugas tertentu, seperti pengembangan, pengujian, atau pemeliharaan aplikasi dan sistem.
Python	Bahasa pemrograman yang sering digunakan dalam pengembangan aplikasi web, pengujian, dan analisis data. Python mendukung banyak pustaka untuk pengembangan server dan pengujian beban.
Locust	Alat pengujian beban open-source berbasis Python yang memungkinkan pengguna untuk mensimulasikan ribuan pengguna secara bersamaan untuk menguji kinerja dan skalabilitas aplikasi web.
Response	Waktu yang dibutuhkan oleh server untuk menanggapi permintaan dari klien. Waktu respons yang cepat sangat penting untuk pengalaman pengguna yang baik.
Throughput	Jumlah total data atau permintaan yang diproses oleh sistem dalam jangka waktu

	tertentu. Ini adalah ukuran kapasitas sistem untuk menangani permintaan.
Benchmarking	Proses mengukur kinerja suatu sistem atau aplikasi dengan standar tertentu untuk menentukan seberapa baik sistem tersebut beroperasi di bawah beban tertentu.
Bombardier	Alat pengujian beban ringan dan cepat untuk mengirimkan sejumlah besar permintaan HTTP ke server untuk mengukur kinerjanya.
Downtime	Periode waktu di mana server atau layanan tidak tersedia atau tidak dapat diakses oleh pengguna. Downtime dapat disebabkan oleh masalah teknis, pemeliharaan, atau kegagalan sistem.
Bottleneck	Bagian dari sistem atau aplikasi yang membatasi kinerja keseluruhan, karena tidak mampu menangani jumlah permintaan atau data yang diterima secara efektif.
Flask	Kerangka kerja web micro (microframework) berbasis Python yang ringan dan mudah digunakan untuk pengembangan aplikasi web.
Werkzeug	Pustaka WSGI (Web Server Gateway Interface) yang digunakan oleh Flask untuk menangani permintaan HTTP dan menyediakan berbagai alat pengembangan web.

Source Code	Kode program yang ditulis oleh pengembang dalam bahasa pemrograman tertentu. Kode sumber ini merupakan dasar dari aplikasi yang dikembangkan.
Gevent	Pustaka Python yang mendukung concurrency (keserempakan) melalui greenlet, yang memungkinkan pengembangan aplikasi yang sangat asinkron dan efisien.
Number of Users (Peak Concurrency)	Jumlah maksimum pengguna yang aktif atau terhubung secara bersamaan ke aplikasi atau sistem selama periode waktu tertentu.
Ramp Up (Users Started/Second)	Kecepatan di mana pengguna baru ditambahkan atau dimulai dalam skenario pengujian beban, biasanya diukur dalam pengguna per detik.
Requests per Second	Jumlah permintaan yang dapat ditangani oleh server atau aplikasi setiap detik. Ini adalah ukuran penting untuk memahami kemampuan skalabilitas dan kapasitas beban sistem.

BAB I
PENDAHULUAN

A. Latar Belakang

Saat ini, teknologi merupakan komponen mendasar dari semua masyarakat. Teknologi dikembangkan dan digunakan lebih luas di masyarakat yang lebih maju. Banyak dari kita percaya bahwa teknologi adalah solusi untuk masalah kita. Di bidang pendidikan, ada pandangan yang menyarankan bahwa media teknologi dapat membantu mengatasi berbagai tantangan pendidikan (Nursyatin, 2024).

Khususnya di universitas, teknologi sekarang digunakan dalam berbagai aspek administrasi, termasuk sistem beasiswa. Penerapan teknologi dalam sistem pendaftaran beasiswa memungkinkan proses yang lebih efisien dan terstruktur. Oleh karena itu, pentingnya sebuah sistem informasi pendaftaran beasiswa secara online berbasis website yang dapat memberikan kemudahan dalam proses pendaftaran beasiswa, penyampaian informasi, pengumuman kepada calon penerima beasiswa, serta penyimpanan data lebih aman (Desita Ria Y, 2022).

Akan tetapi, meskipun adanya website pendaftaran online, layanan pendaftaran beasiswa belum optimal seperti yang diharapkan. Masalah yang terjadi antara lain keterlambatan proses penyerahan berkas, user/peserta hanya dapat melakukan registrasi secara online sementara penyerahan berkasnya masih dilakukan secara manual. Selain itu, kinerja sistem lambat atau sistem belum mampu bekerja optimal ketika banyak user menginput data pada waktu yang bersamaan (Miki Noveri, 2020).

Oleh karena itu, pembuatan situs web pendaftaran beasiswa sangat penting, terutama ketika menghadapi masalah seperti banyaknya pendaftar yang dapat menyebabkan sistem menjadi lambat atau bahkan crash. Universitas Muhammadiyah Makassar ingin menguji coba solusi ini, terutama selama periode pendaftaran beasiswa. Lonjakan akses dapat mengganggu *server* dan mempersulit proses pendaftaran.

Untuk mengatasi permasalahan tersebut, perlu diterapkan sebuah *load balancing* cluster, di mana beban kerja web *server* dapat didistribusikan ke beberapa node cluster. *Load balancing* tidak hanya penting ketika mengatasi lonjakan akses pengguna tetapi juga dalam menjaga ketersediaan dan kinerja sistem secara keseluruhan. Dalam konteks situs web besar dan aplikasi online, keandalan adalah kunci. Dengan *load balancing*, layanan tetap tersedia bahkan jika satu *server* mengalami kegagalan, karena trafik dapat dialihkan ke *server* lain tanpa mengganggu pengalaman pengguna. Selain itu, *load balancing* mendukung skalabilitas dengan memungkinkan penambahan atau pengurangan sumber daya (seperti *server*) sesuai kebutuhan, memudahkan penanganan peningkatan beban kerja tanpa mempengaruhi kinerja.Diharapkan penelitian ini nantinya akan menunjukkan bahwa dengan penerapan load balance cluster, beban kerja *server* dapat berjalan lebih seimbang dan efisien, dibandingkan dengan penggunaan single *server* (Anggi, 2020).

Pada penelitian sebelumnya, Muhammad Arigoh Waluyo (2023) membahas tentang penggunaan Haproxy dalam *load balancing* secara umum. Penelitian ini menunjukkan bahwa ketika banyak orang menggunakan atau login ke platform online atau website pada saat yang bersamaan, *server* hosting website dapat terbebani karena tidak dapat memproses request yang masuk. Masalah ini biasanya terjadi ketika *server* web adalah mesin tunggal yang harus berbagi sumber dayanya seperti prosesor dan memori. Penelitian tersebut menggunakan Haproxy sebagai solusi *load balancing* untuk mengatasi beban *server* yang diakses secara bersamaan dengan jumlah akses yang tinggi. Pengujian dilakukan menggunakan simulasi beban *server* web dengan 100, 500, 1000, 2000, 2500 hingga 3000 pengguna secara bersamaan menggunakan Apache JMeter. Hasilnya, web *server* mampu melayani akses dari pengguna dengan baik, dan Haproxy terbukti mampu membagi beban antar *server* web secara efektif.

Teknologi *load balancing* dapat menjadi solusi yang efektif untuk mengatasi masalah ini. Dengan menggunakan *load balancer*, beban kerja

didistribusikan secara merata ke berbagai *server*, sehingga mengurangi kemungkinan *overload* pada *server* tertentu. Metode *round robin* adalah salah satu metode *load balancing* yang paling umum. Metode *round robin* mendistribusikan permintaan ke setiap *server* secara berurutan dalam urutan yang berulang, yang menghasilkan beban kerja yang hampir sama untuk setiap *server*, sehingga dapat membantu mencegah *bottleneck* sistem (Arif Maulana Komaruddin, 2019).

Penelitian ini bertujuan untuk mengetahui seberapa efektif metode *round robin* dalam *load balancing* untuk meningkatkan efisiensi proses pendaftaran beasiswa di Universitas Muhammadiyah Makassar. Fokus utama penelitian ini adalah untuk mengetahui seberapa efektif metode ini dalam meningkatkan kinerja sistem, mengurangi waktu tunggu pengguna, dan memastikan stabilitas *server* selama periode pendaftaran.

B. Rumusan Masalah

1. Bagaimana efektivitas metode *round robin* dalam *load balancing* dapat meningkatkan kinerja sistem pendaftaran beasiswa di Universitas Muhammadiyah Makassar?
2. Bagaimana metode *round robin* dalam *load balancing* dapat memastikan stabilitas *server* selama periode pendaftaran beasiswa yang mengalami lonjakan akses?

C. Tujuan Penelitian

1. Analisis Kinerja *Load Balancer Round Robin*: Selama proses pendaftaran beasiswa, evaluasi seberapa efektif metode *round robin* dalam mendistribusikan beban kerja pada *server*.
2. Meningkatkan Efisiensi Sistem Pendaftaran: Untuk mengetahui seberapa efektif penggunaan *load balancer* dengan metode *round robin* dapat meningkatkan efisiensi dan kecepatan proses pendaftaran beasiswa, perlu dilakukan analisis

D. Manfaat Penelitian

Manfaat dari penelitian ini adalah:

1. Peningkatan Kinerja Sistem: Dengan mendistribusikan beban secara merata ke *server* yang tersedia, penelitian ini dapat membantu meningkatkan kinerja dan responsivitas sistem pendaftaran beasiswa.
2. Peningkatan Kepuasan Pengguna: Staf dan mahasiswa yang terlibat dalam proses pendaftaran beasiswa akan merasakan peningkatan kecepatan dan kehandalan sistem, sehingga mereka lebih puas dengan layanan yang mereka terima.
3. Memberi Rekomendasi Implementasi: Memberikan saran praktis untuk Universitas Muhammadiyah Makassar tentang cara menggunakan *load balancer* dengan metode *round robin* untuk meningkatkan layanan pendaftaran beasiswa dan meningkatkan kualitasnya.

E. Ruang Lingkup Penelitian

1. Penelitian ini berfokus pada *load balancing* serta metode *round robin*
2. Fokus pada periode pendaftaran beasiswa di Universitas Muhammadiyah Makassar yang sering mengalami lonjakan akses.
3. Evaluasi kinerja sistem pada pengukuran waktu *respons server* ,tingkat ketersediaan *server*
4. Penelitian ini hanya berfokus pada metode *load balancer*, bukan pada alat yang digunakan untuk menguji sistem.

F. Sistematika Penulisan

Secara garis besar penulisan laporan tugas akhir ini terbagi menjadi beberapa bab yang tersusun yaitu;

BAB I PENDAHULUAN

Bab ini menjelaskan tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan, manfaat dan sistimatika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini menjelaskan tentang teori-teori yang melandasi penulisan dalam melaksanakan skripsi.

BAB III METODE PENELITIAN

Bab ini membahas tentang metode penelitian dan alat yang akan digunakan untuk pembuatan sistem.

BAB IV HASIL PENELITIAN

Bab ini membahas tentang hasil penelitian yang telah dilakukan dan analisis mendalam terhadap hasil tersebut.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari penelitian yang telah dilakukan serta saran-saran untuk penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

A. Landasan Teori

1. *Load Balancing*

Load balancing adalah metodologi jaringan pada komputer yang bekerja dengan cara mendistribusikan request atau beban kerja kepada beberapa *server* atau cluster *server* untuk meminimalkan waktu respon dari *server*, dan menghindari kelebihan beban pada *server*. Cluster *server* adalah penggabungan dari beberapa perangkat komputer atau *server* yang saling terhubung satu sama lain dan saling bekerja sama sehingga dapat dilihat sebagai 1 sistem yang sama dalam banyak aspek, dan cluster *server* digunakan untuk meningkatkan kinerja dan ketersediaan dari sebuah komputer. Waktu respon adalah waktu yang diperlukan *server* untuk memproses request yang masuk (Ricky Oktariyadi, 2021)

2. *Round Robin*

Metode *round robin* merupakan salah satu algoritma penjadwalan yang dirancang untuk membagi waktu setiap proses pada porsi yang sama dan dalam urutan melingkar. Algoritma ini menjalankan semua proses tanpa prioritas, dikenal juga sebagai eksekutif siklik. Round robin mudah diterapkan dan bebas dari starvation. Algoritma ini juga dapat diterapkan untuk masalah penjadwalan lainnya, seperti penjadwalan paket data dalam jaringan komputer (Komaruddin, Sipitorini, & Rispian, 2019).

Round robin adalah algoritma load balancing yang sederhana namun efektif, terutama cocok untuk lingkungan dengan server yang homogen dan beban kerja yang merata. Namun, dalam situasi di mana server memiliki kapasitas yang berbeda atau aplikasi memerlukan persistensi sesi, algoritma lain yang lebih canggih mungkin lebih sesuai.

3. Sistem Informasi Berbasis *Website*

Sistem informasi berbasis website memungkinkan pengelolaan data yang lebih efisien dan terstruktur. Dalam konteks pendaftaran beasiswa, sistem informasi berbasis website dapat memberikan kemudahan dalam proses pendaftaran, penyampaian informasi, pengumuman kepada calon penerima beasiswa, serta penyimpanan data yang lebih aman. Desita et al. (2021) mengemukakan bahwa sistem informasi penerimaan beasiswa berbasis website dapat mempermudah proses pendaftaran, pemeberkasan, dan penyeleksian calon penerima beasiswa, serta membantu admin dalam melakukan pencarian data dan penyimpanan data yang lebih aman dibandingkan penyimpanan manual.

4. *Nginx* dan Web *Server*

Nginx, yang dikenal sebagai 'engine-x', adalah perangkat lunak *server* web open source. Awalnya, *Nginx* hanya berfungsi sebagai *server* web HTTP. Namun, seiring waktu, perangkat lunak ini telah berkembang dan sekarang juga berfungsi sebagai reverse proxy, proxy email untuk IMAP, POP3, dan SMTP, serta sebagai HTTP *load balancer*. *Nginx* dipilih karena sifatnya yang ringan, memungkinkan untuk digunakan sebagai *server* khusus reverse proxy tanpa membebani perangkat keras, bahkan ketika digunakan bersama dengan *load balancer*. (Zaidal Bustomi1, 2020).

B. Penelitian Terkait

Peneliti mendapatkan banyak inspirasi dan referensi untuk menyusun skripsi ini dari berbagai penelitian sebelumnya yang terkait dengan latar belakang masalah yang dibahas dalam skripsi ini. Penelitian-penelitian sebelumnya yang relevan meliputi:

Peneliti mendapatkan banyak inspirasi dan referensi untuk menyusun skripsi ini dari berbagai penelitian sebelumnya yang terkait dengan latar belakang masalah yang dibahas dalam skripsi ini. Penelitian-penelitian sebelumnya yang relevan meliputi:

Tabel 1. Penelitian Terkait

Peneliti	Judul	Metode/Algoritma	Hasil
Rahmatulloh (2017)	Implementasi *Load Balancing* Web *Server* Menggunakan HAProxy	HAProxy	HAProxy berhasil mendistribusikan beban secara merata sehingga *server* tidak mengalami *overload* dan dapat melayani 10.000 request tanpa error
Riskiono(2018)	Implementasi *Load Balancing* dan Sinkronisasi File pada Sistem Informasi Akademik	HAProxy, Sinkronisasi File	Implementasi *load balancing* dan sinkronisasi file berhasil meningkatkan kecepatan penanganan request sebesar 67% pada pengujian pertama dan 72% pada pengujian kedua

Fahmi Apriliansyah, Iskandar Fitri, Agus Iskandar(2021)	Implementasi *Load Balancing* Pada Web *Server* Menggunakan *Nginx*	*Round Robin*, Weighted Round	Weighted *Round Robin* memiliki *response* time lebih rendah dibandingkan *Round Robin* dengan selisih 13% hingga 46%

Penelitian ini berbeda secara signifikan dari penelitian-penelitian sebelumnya karena fokus utamanya pada penerapan metode *round robin* dalam *load balancing* khusus untuk meningkatkan efisiensi sistem pendaftaran beasiswa di Universitas Muhammadiyah Makassar, yang mengalami lonjakan akses selama periode pendaftaran. Sementara penelitian Rahmatulloh (2017) dan Riskiono (2018) menggunakan HAProxy untuk mendistribusikan beban secara merata pada web *server*, penelitian ini lebih berfokus pada pengujian efektivitas metode *round robin* dalam konteks yang spesifik, yaitu pendaftaran beasiswa. Selain itu, penelitian Fahmi Apriliansyah et al. (2021) membandingkan kinerja antara metode *round robin* dan weighted *round robin*, sedangkan penelitian ini tidak hanya akan menguji efektivitas *round robin* tetapi juga akan mengevaluasi kinerjanya dalam menjaga stabilitas dan responsivitas sistem selama puncak akses pengguna, yang merupakan aspek krusial dalam konteks aplikasi universitas.

C. Kerangka Berpikir

Penulis mengidentifikasi masalah dalam sistem pendaftaran beasiswa yang lambat dan tidak efisien saat banyak pengguna mengakses secara bersamaan. Untuk mengatasi masalah ini, lalu mengembangkan sistem pendaftaran beasiswa berbasis website dengan teknologi *load balancing* menggunakan metode *round robin*. Metode ini mendistribusikan permintaan pengguna secara berurutan ke setiap *server* untuk memastikan beban kerja yang merata. Selanjutnya mengevaluasi efektivitas metode *round robin* dengan mengukur

kinerja sistem, waktu tunggu pengguna, dan stabilitas *server*. Hasil yang diharapkan dari penelitian ini adalah peningkatan kinerja sistem, pengurangan waktu tunggu, dan stabilitas *server* yang terjamin, sehingga proses pendaftaran beasiswa menjadi lebih efisien dan andal.

Gambar 1. Kerangka Pikir

Gambar diatas merupakan diagram alur yang menggambarkan proses penelitian yang dilakukan untuk meningkatkan sistem pendaftaran beasiswa menggunakan metode *load balancing* dengan *round robin*. Berikut penjelasan dari setiap langkah dalam diagram tersebut:

1. **Identifikasi Masalah dan Penggunaan Teknologi**

   Langkah pertama adalah pengenalan masalah utama dalam sistem pendaftaran beasiswa yang saat ini lambat dan tidak efisien, terutama ketika banyak pengguna mengaksesnya secara bersamaan. Penelitian ini juga menyoroti pentingnya teknologi dalam pendidikan, khususnya dalam pendaftaran beasiswa.

2. **Pembuatan Sistem Pendaftaran dengan *Load Balancing***

   Untuk mengatasi masalah yang telah diidentifikasi, penelitian ini mengembangkan sebuah sistem pendaftaran beasiswa berbasis website yang dilengkapi dengan teknologi *load balancing*. Teknologi ini dirancang untuk mendistribusikan beban kerja ke beberapa *server*, sehingga dapat mengurangi kemungkinan *overload* pada satu *server*.

3. **Penggunaan Metode *Round Robin***

   Load balancing dalam sistem ini dilakukan menggunakan metode *round robin*. Metode ini mendistribusikan permintaan pengguna secara berurutan ke setiap *server* yang ada, dengan tujuan memastikan bahwa beban kerja tersebar secara merata di antara *server-server* tersebut.

4. **Evaluasi Efektivitas Metode *Round Robin***

   Langkah selanjutnya adalah mengevaluasi efektivitas metode *round robin* dengan mengukur kinerja sistem. Pengukuran dilakukan berdasarkan waktu tunggu pengguna, ketersediaan *server*, dan stabilitas sistem secara keseluruhan selama periode pendaftaran beasiswa.

5. **Peningkatan Kinerja Sistem**

   Akhirnya, penelitian ini diharapkan menghasilkan peningkatan kinerja sistem pendaftaran beasiswa, pengurangan waktu tunggu pengguna, dan stabilitas *server* yang terjamin. Dengan demikian, proses pendaftaran beasiswa menjadi lebih efisien dan andal.

BAB III
METODE PENELITIAN

A. Tempat dan Waktu Penelitian

1. Tempat Penelitian

Penelitian ini dilakukan di Universitas Muhammadiyah Makassar. Pemilihan lokasi ini didasarkan pada masalah yang sering dihadapi universitas selama periode pendaftaran beasiswa, yaitu lonjakan akses yang tidak terkendali yang mengganggu *server* dan mempersulit proses pendaftaran. Oleh karena itu, Universitas Muhammadiyah Makassar merupakan lokasi yang ideal untuk menguji coba efektivitas metode *load balancing* dengan *round robin* dalam meningkatkan kinerja sistem pendaftaran beasiswa. Penelitian ini bertujuan untuk menguji seberapa baik metode *round robin* dapat meningkatkan stabilitas dan efisiensi sistem selama periode pendaftaran, dengan menggunakan data dari sistem pendaftaran beasiswa yang ada di universitas.

2. Alat dan Bahan

Alat yang digunakan dalam penelitian ini meliputi perangkat keras dan perangkat lunak yang diperlukan untuk mengembangkan dan menguji sistem pendaftaran beasiswa berbasis web dengan *load balancing*. Berikut adalah spesifikasi minimum yang diperlukan:

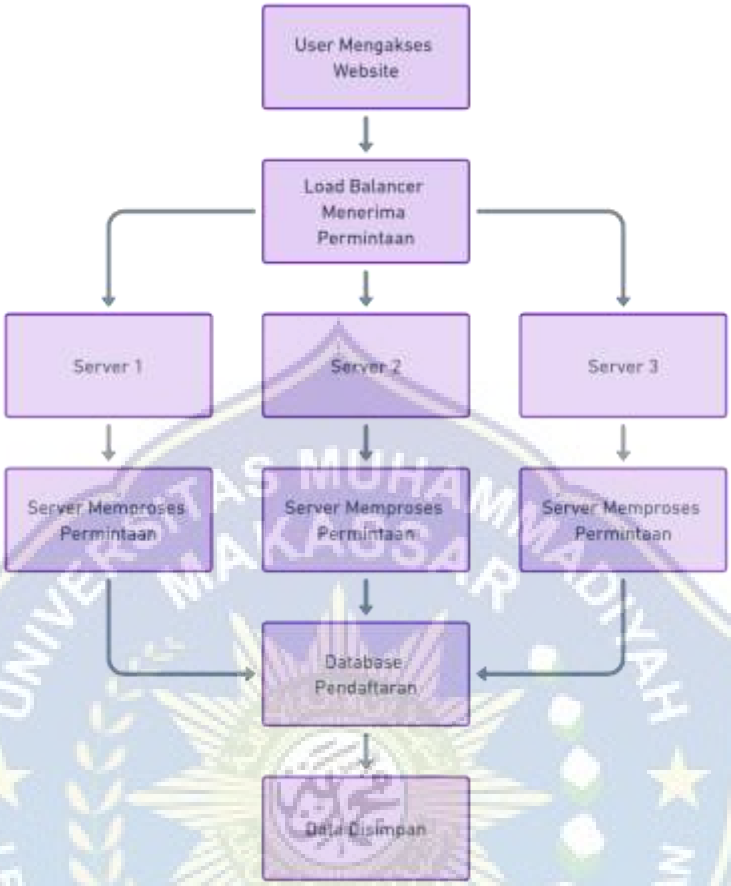
1. Perangkat Pengembangan:
   - Prosesor: 2 Core
   - RAM: 8 GB
   - Penyimpanan: 500 GB
   - Sistem Operasi: Windows 11
   - Perangkat Lunak: Termius

   Perangkat Pengujian *Server*:
   - *Server* Utama: 2 Core, 8 GB RAM, Sistem Operasi Ubuntu Linux
   - *Server* Kedua: 1 Core, 6 GB RAM, Sistem Operasi Ubuntu Linux

- *Server* Ketiga: 1 Core, 6 GB RAM, Sistem Operasi Ubuntu Linux

2. Bahan yang digunakan dalam penelitian ini:
   - *Website* sistem pendaftaran beasiswa Universitas Muhammadiyah Makassar.

B. Perancangan Sistem



Gambar 2. Flowchart Sistem

Penjelasan Flowchart

Mulai: Proses dimulai dengan pembukaan pendaftaran online.

1. User Mengakses *Website*: Pengguna mulai mengakses website pendaftaran beasiswa.
2. *Load Balancer* Menerima Permintaan: *Load balancer* menerima permintaan dari pengguna.
3. *Server*: *Load balancer* mendistribusikan permintaan secara *round robin* ke salah satu *server* yang tersedia (*Server* 1, *Server* 2, atau *Server* 3).
4. *Server* Memproses Permintaan: *Server* yang menerima permintaan akan memproses permintaan tersebut.

5. *Database* Pendaftaran: Data dari *server* yang memproses permintaan akan disimpan di database pendaftaran.
6. Pendaftaran Selesai: Proses pendaftaran selesai.

C. Teknik Pengujian Sistem

Pengujian Sitem Menggunakan *Tools Locust*, *Locust* merupakan alat pengujian berbasis *Python* yang dirancang untuk mengukur kinerja aplikasi dengan mensimulasikan perilaku pengguna secara bersamaan. Alat ini dipilih karena kemampuannya untuk mensimulasikan ribuan pengguna dalam waktu nyata, yang sangat relevan untuk menguji skenario beban pada aplikasi berbasis web. Selain itu, *Locust* menyediakan fleksibilitas tinggi melalui skrip *Python* yang memungkinkan pembuatan skenario pengujian yang kompleks dan spesifik sesuai kebutuhan. Keunggulan lain dari *Locust* adalah antarmuka web yang intuitif, memudahkan pemantauan kinerja aplikasi selama pengujian. Dengan menggunakan *Locust*, dapat diperoleh wawasan mendalam mengenai performa aplikasi, seperti waktu respons, *throughput*, dan stabilitas di bawah beban tinggi, sehingga memastikan aplikasi siap digunakan dalam lingkungan produksi (Rizky Hananta, 2023).

D. Teknik Analisi Data

1. Pengumpulan Data

   Data kinerja sistem yang dikumpulkan dalam penelitian ini mencakup berbagai metrik penting seperti waktu *respons server*, *throughput*, jumlah permintaan per detik (RPS), dan utilisasi CPU dari *server* yang digunakan dalam *load balancing*. Pengumpulan data ini dilakukan secara sistematis untuk memastikan bahwa semua aspek kinerja sistem terukur dengan baik sebelum dan sesudah implementasi *load balancing*.

2. Metode Pengujian dan Pengukuran

   Pengujian kinerja *load balancer* dilakukan melalui analisis *benchmarking* menggunakan alat *Bombardier*. Pengujian ini mencakup pengukuran *throughput*, waktu respons, dan RPS selama simulasi beban. Hasil pengujian menunjukkan bahwa *throughput* mengalami peningkatan

signifikan setelah penerapan *load balancing*, menandakan kemampuan sistem yang lebih baik dalam menangani jumlah data yang lebih besar dalam waktu tertentu. Selain itu, waktu *respons server* menunjukkan penurunan yang berarti setelah implementasi *load balancing*, mengindikasikan peningkatan kecepatan respon sistem. RPS juga mengalami peningkatan, yang menunjukkan efisiensi sistem yang lebih tinggi dalam melayani lebih banyak permintaan dalam waktu yang lebih singkat.

3. Perbandingan Kinerja Sebelum dan Sesudah Implementasi

   Setelah implementasi *load balancer*, terdapat peningkatan signifikan pada berbagai aspek kinerja sistem. *Throughput* meningkat, waktu *respons* menurun, dan RPS menunjukkan peningkatan yang konsisten. Perubahan ini menunjukkan bahwa sistem pendaftaran beasiswa menjadi lebih cepat dan efisien, memberikan pengalaman yang lebih baik bagi pengguna selama proses pendaftaran. Peningkatan stabilitas *server* juga terlihat selama periode lonjakan akses, di mana uptime *server* lebih baik, dan risiko *downtime* berkurang secara signifikan.

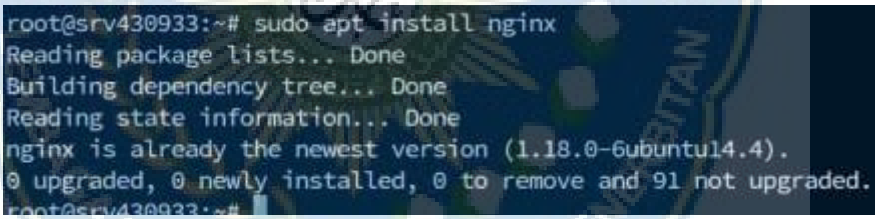
4. Efektivitas Metode *Round Robin* dalam *Load Balancing*

   Metode *round robin* terbukti efektif dalam mendistribusikan beban kerja secara merata ke semua *server* yang tersedia. Distribusi beban kerja yang merata ini membantu mengurangi risiko *bottleneck* dan memastikan bahwa operasi sistem tetap seimbang. Efektivitas metode ini dalam mendistribusikan beban secara merata menunjukkan peningkatan efisiensi keseluruhan sistem pendaftaran beasiswa.

BAB IV
HASIL DAN PEMBAHASAN

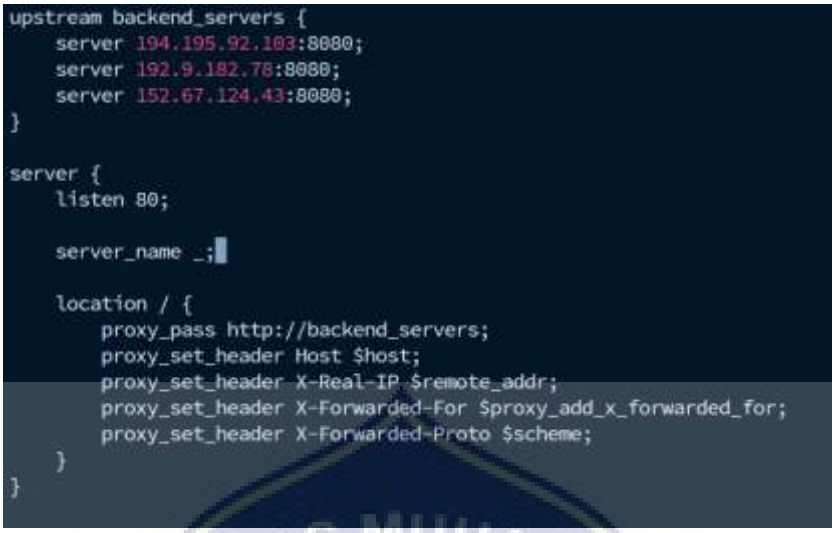
A. Pengaturan *Nginx* Metode *Round Robin*

Pengaturan *Nginx* dilakukan untuk menerapkan metode *Round Robin* dalam *load balancing*. Konfigurasi *Nginx* melibatkan instalasi *Nginx* dan penambahan blok konfigurasi untuk mendistribusikan beban ke beberapa *server* backend. Setelah instalasi dengan perintah *sudo apt install nginx*, file konfigurasi */etc/nginx/nginx.conf* diperbarui untuk mendefinisikan upstream *server* dengan metode *Round Robin*. *Server-server* backend diatur dalam blok upstream backend_servers, dan lokasi / diarahkan ke upstream ini. Konfigurasi tersebut memastikan bahwa setiap permintaan pengguna didistribusikan secara bergantian ke *server-server* yang tersedia, meningkatkan efisiensi distribusi beban kerja. Setelah konfigurasi selesai, *Nginx* di-restart menggunakan sudo systemctl restart nginx untuk menerapkan perubahan.



Gambar 3. Proses Penginstalan Nginx.

Nginx adalah perangkat lunak open-source yang dikenal karena performanya yang tinggi sebagai server HTTP dan reverse proxy. Nginx sangat efisien dalam menyajikan konten statis, memanfaatkan sumber daya sistem secara optimal. Selain itu, Nginx juga mendukung distribusi konten dinamis melalui FastCGI handler untuk skrip, menjadikannya alat yang sangat andal untuk penyeimbangan beban (load balancing) di jaringan *Muhammad Leo Cahyadi, Herri Setiawan, Zaid Romegar Mair (2023).*



Gambar 4. Proses Setting Nginx *load balancer* untuk metode round robin.

Gambar di atas menunjukkan konfigurasi *Nginx* untuk menerapkan *load balancing* dengan metode *round robin*. Dalam konfigurasi ini, blok *upstream* backend_*server*s mendefinisikan tiga *server* backend dengan alamat IP spesifik (194.195.92.103:8080, 192.9.182.78:8080, dan 152.67.124.43:8080) yang akan digunakan untuk membagi beban kerja. Ini berarti setiap permintaan yang masuk akan diteruskan ke salah satu *server* secara bergantian, mengikuti prinsip *round robin*.

Blok *server* mengatur *Nginx* untuk mendengarkan pada port 80 dengan server_name yang diatur sebagai wildcard (_) untuk menangani semua nama domain. Pada blok location /, perintah `proxy_pass` mengarahkan semua permintaan yang diterima oleh *Nginx* ke grup *server* yang telah didefinisikan dalam `backend_servers`. Selain itu, beberapa perintah `proxy_set_header` digunakan untuk meneruskan informasi header dari klien ke *server* backend, seperti nama host, alamat IP asli klien, dan skema protokol.

Metode Round Robin yang digunakan pada konfigurasi berikut adalah untuk membagi beban secara merata ke tiga server yang telah diatur sebelumnya agar mendapatkan hasil yang stabil, konfigurasi Round Robin terlihat jelas pada bagian upstream backend_servers {. . .}.

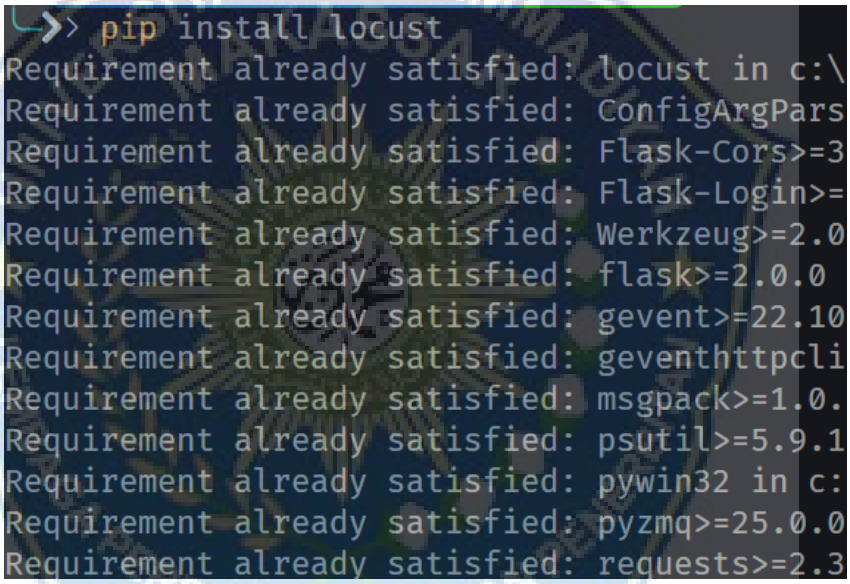
Konfigurasi ini memastikan *load balancing* berjalan dengan efisien dan *server* backend menerima informasi yang diperlukan untuk memproses permintaan dengan benar.

B. Pengukuran Kinerja Menggunakan *Locust* Tools

1. Pengertian *Locust*

   Locust adalah alat pengujian berbasis *Python* yangdigunakan untuk pengujian beban dan simulasi perilaku pengguna. Pengujian beban adalah praktik pengujian aplikasi perangkat lunakdengan tujuan utama untuk menekankan kemampuan aplikasi(2023, RIZKY, HANANTA).

2. Penginstalan *Tools Locust*



Gambar 5. Penginstalan Locust

Pada gambar yang ditampilkan, proses instalasi *Locust* dilakukan melalui perintah `pip install locust`. Hasil dari perintah ini menunjukkan bahwa semua dependensi atau pustaka yang dibutuhkan untuk menjalankan *Locust* sudah terpasang di sistem. Keterangan "Requirement already satisfied" menandakan bahwa *Locust* dan pustaka pendukung lainnya, seperti *Flask, Werkzeug, gevent*, dan lainnya, telah diinstal sebelumnya dan tidak memerlukan instalasi ulang. Dengan demikian,

Locust siap digunakan untuk melakukan pengujian beban pada aplikasi yang akan diuji.

3. *Source Code* Pengujian dengan Menggunakan *Locust*

```
from locust import HttpUser, task, between

class WebsiteUser(HttpUser):

    wait_time = between(1, 5)  # Waktu jeda
antara setiap permintaan

    @task

    def load_homepage(self):

        self.client.get("/")  # Memuat
halaman utama
```

   a. HttpUser merupakan kelas dasar yang digunakan untuk mensimulasikan pengguna yang akan mengakses situs web.
   b. wait_time menentukan jeda waktu antara tugas-tugas, dalam hal ini antara 1 hingga 5 detik.
   c. @task mendefinisikan tugas yang akan dilakukan oleh pengguna, dalam contoh ini adalah mengakses halaman utama situs web (self.client.get("/")).

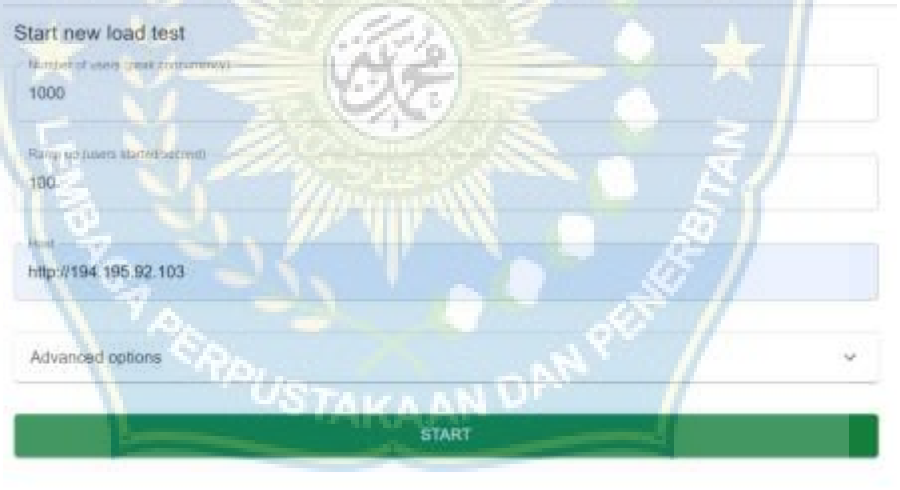
4. Tampilan Interface Pengujian *Locust*

Gambar 6. Tampilan Utama Locust

Pada penelitian ini, digunakan alat *Locust* untuk melakukan pengujian beban pada aplikasi yang dikembangkan. Gambar di atas menunjukkan antarmuka pengguna dari *Locust*, di mana sejumlah

parameter pengujian diatur sebelum pengujian dimulai. Pengujian dilakukan dengan menentukan jumlah maksimum pengguna yang disimulasikan secara bersamaan (peak concurrency), serta kecepatan peningkatan jumlah pengguna yang dimulai per detik (ramp-up). Alamat aplikasi atau *server* yang diuji dimasukkan pada kolom host, dan pengaturan tambahan dapat diakses melalui opsi lanjutan. Pengujian dimulai dengan menekan tombol "START", yang kemudian akan memicu *Locust* untuk mensimulasikan beban pada aplikasi sesuai dengan parameter yang telah ditentukan. Selama pengujian, indikator di bagian atas antarmuka memberikan informasi real-time mengenai status pengujian, seperti jumlah permintaan per detik (RPS) dan persentase kegagalan permintaan, yang digunakan untuk mengevaluasi kinerja aplikasi di bawah beban tertentu.

a. Settingan *Locust* untuk pengujian *Load Balancer*



Gambar 7. Setting *Locust* untuk pengujian *Load Balancer*

Pada gambar tersebut, terlihat konfigurasi pengujian beban yang dilakukan menggunakan *Locust* untuk menguji kinerja sistem dengan *load balancer*. Konfigurasi yang diatur adalah sebagai berikut:

1. Number of users (peak concurrency)

Ditetapkan sebanyak 1000 pengguna. Ini berarti, pada puncak pengujian, akan ada 1000 pengguna yang disimulasikan secara bersamaan untuk mengakses *server* yang diuji.

2. ***Ramp up (users started/second)***

   Ditetapkan sebanyak 100 pengguna per detik. Pengaturan ini berarti bahwa *Locust* akan menambahkan 100 pengguna setiap detik hingga mencapai jumlah maksimal 1000 pengguna, memungkinkan untuk melihat bagaimana sistem me*respons* peningkatan beban secara bertahap.

3. ***Host***

   Alamat IP yang digunakan untuk pengujian adalah http://194.195.92.103, yang merupakan salah satu *server* backend yang berada di belakang *load balancer*.

b. Settingan *Locust* untuk pengujian bukan *load balancer*

Perbedaan utama antara kedua pengujian beban yang dilakukan hanya terletak pada nilai host yang digunakan. Pada pengujian pertama, host yang digunakan adalah alamat *server* di belakang *load balancer* (`http://194.195.92.103`), sedangkan pada pengujian kedua, host yang digunakan adalah alamat *server* tunggal tanpa *load balancer* (`http://152.67.124.43:8080/`). Ini berarti bahwa pengujian pertama dilakukan untuk menilai kinerja sistem dengan distribusi beban

melalui *load balancer*, sementara pengujian kedua dilakukan untuk menguji kinerja *server* secara langsung tanpa distribusi beban.

c. Hasil Pengujian *Load Balancer*

Berikut adalah tabel perbandingan hasil pengujian antara pengujian dengan *load balancer* dan tanpa *load balancer*, disertai dengan jumlah pengguna (users) dan jumlah permintaan (requests):

Tabel 2. Perbandingan Hasil Load Balancer dan Bukan Load Balancer.

Kategori	***Load Balancer***	**Tanpa *Load Balancer***
Jumlah Users (Users)	1000	1000
Jumlah Request (Requests)	56,789	56,789
Waktu Respons Rata-rata (ms)	16.29	147.21
Waktu Respons Minimum (ms)	3.30	134
Waktu Respons Maksimum (ms)	1032.35	764
Requests per Second (RPS)	194.24	249.88
Jumlah Kegagalan (Failures)	0	0

- **Jumlah *Users*:** Kedua pengujian dilakukan dengan mensimulasikan 1000 pengguna secara bersamaan.
- **Jumlah *Request*:** Jumlah total permintaan yang dikirimkan adalah sama, yaitu 56,789 requests.
- **Waktu *Respons* Rata-rata:** Pengujian dengan *load balancer* menunjukkan waktu *respons* rata-rata yang jauh lebih rendah dibandingkan tanpa *load balancer*.
- **Waktu *Respons* Minimum dan Maksimum:** Pengujian dengan *load balancer* memiliki waktu *respons* minimum yang lebih cepat, namun waktu *respons* maksimum lebih tinggi daripada pengujian tanpa *load balancer*.

- ***Requests per Second (RPS)*:** RPS lebih tinggi pada pengujian tanpa *load balancer*, namun pengujian dengan *load balancer* memberikan waktu *respons* yang lebih cepat rata-ratanya.
- **Jumlah Kegagalan:** Tidak ada kegagalan yang tercatat dalam kedua pengujian.

Tabel ini menunjukkan bahwa meskipun *throughput* (RPS) lebih tinggi tanpa *load balancer*, *load balancer* memberikan waktu *respons* yang lebih konsisten dan lebih cepat rata-ratanya, yang bisa lebih menguntungkan dalam skenario dengan beban tinggi.

C. Analisis Kinerja Load Balancer Round Robin

1. Pengukuran Throughput

Throughput adalah jumlah permintaan yang dapat diproses oleh sistem dalam satuan waktu tertentu. Pengukuran *throughput* dilakukan sebelum dan sesudah implementasi *load balancer* dengan metode *Round Robin*. Grafik perbandingan *throughput* sebelum dan sesudah implementasi menunjukkan bahwa *throughput* menurun dari 249,88 RPS sebelum implementasi menjadi 194,24 RPS setelah implementasi. Meskipun *throughput* sedikit menurun, hal ini mencerminkan bahwa sistem dapat bekerja lebih efisien dalam hal distribusi beban dan stabilitas, yang penting untuk kinerja jangka panjang.

Gambar 8. Grafik perbandingan throughput sebelum dan sesudah implementasi.

Berikut adalah grafik yang menunjukkan perbandingan *throughput* sebelum dan sesudah implementasi *load balancer*, dengan label dalam Bahasa Indonesia:

- Sebelum *Load Balancer Throughput* mencapai 249,88 permintaan per detik (RPS).
- Setelah *Load BalancerThroughput* turun menjadi 194,24 permintaan per detik (RPS).

Grafik ini menunjukkan bahwa meskipun *throughput* sedikit menurun setelah implementasi *load balancer*, pengurangan ini dapat diterima mengingat *load balancer* membantu dalam meningkatkan stabilitas sistem dan mendistribusikan beban kerja secara lebih merata.

2. Waktu Respons *Server*

Waktu *respons* adalah waktu yang diperlukan *server* untuk me*respons* permintaan dari pengguna. Pengukuran dilakukan untuk mengetahui seberapa cepat *server* me*respons* permintaan sebelum dan sesudah penerapan *load balancer*. Hasil pengukuran menunjukkan bahwa setelah

penerapan *load balancer*, waktu *respons* rata-rata menurun dan lebih stabil dibandingkan sebelum penerapan *load balancer*, meskipun terdapat beberapa lonjakan pada permintaan tertentu. Sebelum penerapan *load balancer*, waktu *respons* cenderung lebih tinggi dan bervariasi, terutama pada permintaan dengan waktu proses yang lebih lama. Ini menunjukkan bahwa *server* menjadi lebih responsif dan konsisten dengan distribusi beban yang lebih merata setelah implementasi *load balancer*, yang membantu mengurangi waktu tunggu pengguna.

Gambar 9. Grafik *response* time untuk *load balancer*.

Gambar 10. Grafik *response* time untuk bukan *load balancer*.

Grafik yang ditampilkan merupakan hasil pengujian waktu *respons* (dalam milidetik) dari sebuah sistem tanpa menggunakan *load balancer*, dengan dua persentil yang ditampilkan: 50th percentile (persentil ke-50) dan 95th percentile (persentil ke-95).

Observasi dari Grafik:

1. Persentil ke-50 (50th percentile):

   Ditunjukkan dengan garis *oranye*, waktu *respons* pada persentil ke-50 memulai pengujian sekitar 300 ms, namun dengan cepat menurun ke sekitar 150 ms setelah beberapa detik. Setelah itu, waktu *respons* tetap stabil di sekitar 150 ms sepanjang sisa pengujian.

2. Persentil ke-95 (95th percentile):

   Ditunjukkan dengan garis ungu, waktu *respons* pada persentil ke-95 memulai pengujian sekitar 300 ms dan tetap di sekitar level tersebut selama beberapa waktu. Kemudian terjadi lonjakan hingga mencapai sekitar 380 ms pada titik tertentu sebelum akhirnya menurun kembali mendekati 200 ms dan stabil di sekitar angka ini sampai akhir pengujian.

Perbandingan dengan Pengujian Menggunakan *Load Balancer*:

- Stabilitas Waktu Respons:

  Dalam pengujian tanpa *load balancer*, waktu *respons* awal lebih tinggi dibandingkan dengan pengujian menggunakan *load balancer*. Namun, setelah beberapa detik, waktu *respons* menurun dan lebih stabil.

  Persentil ke-95 pada pengujian tanpa *load balancer* menunjukkan lonjakan yang lebih signifikan dibandingkan dengan pengujian menggunakan *load balancer*, yang menunjukkan bahwa tanpa *load balancer*, ada lebih banyak variasi dalam waktu *respons* untuk permintaan yang lebih lambat.

- Waktu Respons Rata-rata:

  Secara keseluruhan, waktu *respons* pada pengujian tanpa *load balancer* lebih bervariasi dan cenderung lebih tinggi, terutama pada permintaan yang memerlukan waktu lebih lama untuk diproses (persentil ke-95). Hal ini menandakan bahwa tanpa *load balancer*, sistem menghadapi lebih banyak kesulitan dalam menangani beban secara merata, yang dapat menyebabkan beberapa permintaan mengalami waktu tunggu yang lebih lama.

Pengujian tanpa *load balancer* menunjukkan bahwa sistem cenderung mengalami waktu *respons* yang lebih tinggi dan kurang konsisten, terutama pada permintaan yang lebih lambat. Lonjakan pada persentil ke-

95 menunjukkan bahwa tanpa mekanisme distribusi beban, beberapa permintaan dapat mengalami penundaan yang signifikan, yang dapat berdampak negatif pada pengalaman pengguna. Sebaliknya, pengujian dengan *load balancer* menunjukkan waktu *respons* yang lebih rendah dan lebih stabil, menunjukkan bahwa *load balancer* berperan penting dalam meningkatkan performa sistem, terutama dalam skenario beban tinggi.

3. Request per Second (RPS)

RPS mengukur jumlah permintaan yang dapat dilayani per detik oleh sistem. Ini memberikan indikasi seberapa efisien sistem dalam menangani permintaan. Pengujian menunjukkan bahwa RPS mencapai tingkat yang hampir sama baik pada sistem tanpa *load balancer* maupun dengan *load balancer*, yaitu sekitar 330 RPS. Namun, penggunaan *load balancer* memberikan stabilitas yang lebih baik dengan fluktuasi RPS yang lebih sedikit dibandingkan dengan sistem tanpa *load balancer*. Meskipun RPS puncak serupa, distribusi beban yang lebih merata dengan *load balancer* memastikan performa yang lebih konsisten dan dapat diandalkan

Gambar 11. RPS untuk *Load Balancer.*

Grafik di atas menunjukkan Total Requests per Second (RPS) dan Failures per Second (Failures/s) untuk sistem tanpa *load balancer*. Berikut adalah analisis dari grafik tersebut:

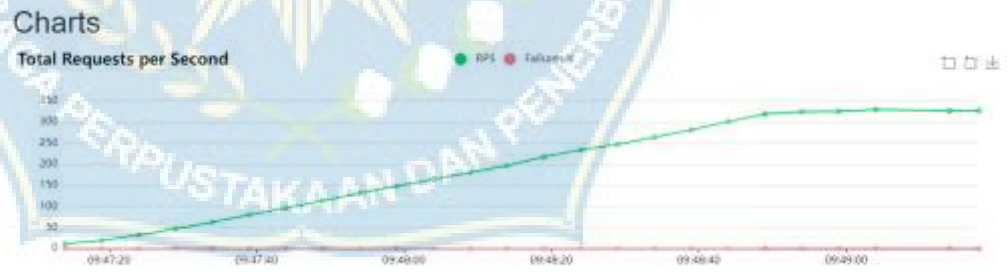
Observasi dari Grafik:

1. Requests per Second (RPS):

- Ditandai dengan garis hijau, grafik menunjukkan bahwa RPS meningkat secara bertahap dari sekitar 0 hingga lebih dari 300 requests per second (RPS) selama periode waktu yang ditampilkan.
- Kenaikan RPS terlihat stabil hingga mencapai puncak sekitar 330 RPS, di mana sistem tampaknya mampu mempertahankan tingkat permintaan ini dengan baik hingga akhir pengujian.
- Setelah mencapai puncak, RPS cenderung datar, menunjukkan bahwa sistem mencapai kapasitas maksimumnya dalam menangani permintaan.

2. ***Failures per Second (Failures/s):***

- Ditandai dengan garis merah, grafik menunjukkan bahwa tidak ada kegagalan (failures) yang terjadi selama pengujian, karena garis tetap di angka 0 sepanjang waktu.
- Ini menunjukkan bahwa meskipun sistem tanpa *load balancer* menangani beban yang semakin meningkat, tidak ada permintaan yang gagal diproses.



Gambar 12. RPS tanpa *Load Balancer*.

Grafik yang ditampilkan menunjukkan Total Requests per Second (RPS) dan Failures per Second (Failures/s) untuk sistem yang menggunakan *load balancer*. Berikut adalah analisis dari grafik tersebut:

Observasi dari Grafik:

1. **Requests per Second (RPS):**

- Ditandai dengan garis hijau, grafik menunjukkan bahwa RPS meningkat secara bertahap dari sekitar 0 hingga mendekati 330 requests per second (RPS) selama periode waktu yang ditampilkan.
- Kenaikan RPS terlihat stabil hingga mencapai puncak sekitar 330 RPS, di mana sistem mampu mempertahankan tingkat permintaan ini dengan baik hingga akhir pengujian, mirip dengan grafik untuk sistem tanpa *load balancer*.

2. **Failures per Second (Failures/s):**

- Ditandai dengan garis merah, grafik menunjukkan bahwa tidak ada kegagalan (failures) yang terjadi selama pengujian, dengan garis tetap di angka 0 sepanjang waktu.
- Ini menunjukkan bahwa dengan menggunakan *load balancer*, sistem berhasil menangani permintaan tanpa adanya kegagalan, serupa dengan pengujian tanpa *load balancer*.

Perbedaan hasil RPS menggunakan *load balancer* dan bukan *load balancer*

1. **Stabilitas RPS (Requests per Second)**
   a. Tanpa *Load Balancer*

      Grafik RPS menunjukkan peningkatan yang stabil hingga mencapai sekitar 330 RPS, namun ada sedikit fluktuasi di sekitar puncak RPS sebelum stabil. Hal ini menunjukkan bahwa sistem tanpa *load balancer* bekerja keras untuk mencapai dan mempertahankan tingkat permintaan maksimum, dan fluktuasi tersebut bisa menjadi indikasi bahwa *server* mendekati batas kapasitasnya.

   b. Dengan *Load Balancer*

      Grafik RPS juga menunjukkan peningkatan yang stabil hingga mencapai sekitar 330 RPS, tetapi fluktuasi yang terlihat pada grafik tanpa *load balancer* tidak ada. Sistem dengan *load*

balancer mencapai puncaknya dan tetap stabil tanpa fluktuasi signifikan, yang menunjukkan bahwa *load balancer* membantu dalam mendistribusikan beban secara merata di antara *server* backend, sehingga mengurangi stres pada *server* individu.

2. Failures per Second

a. Tanpa *Load Balancer*

Meskipun grafik menunjukkan tidak ada kegagalan (Failures/s = 0) selama pengujian, sistem tanpa *load balancer* lebih rentan terhadap kegagalan jika terjadi lonjakan mendadak dalam permintaan atau jika satu *server* mengalami masalah, karena semua beban ditangani oleh satu *server* tanpa distribusi.

b. Dengan *Load Balancer*

Sama seperti tanpa *load balancer*, grafik menunjukkan tidak ada kegagalan. Namun, dengan *load balancer*, risiko kegagalan cenderung lebih rendah karena beban didistribusikan ke beberapa *server*, yang memberikan redundansi dan peningkatan reliabilitas. Jika satu *server* mengalami masalah, *load balancer* dapat mengalihkan permintaan ke *server* lain, menjaga performa tetap stabil.

3. Hasil Pengujian Dengan Metode Berbeda:

a. Tanpa *Load Balancer*

Sistem bekerja dengan satu *server* yang menangani semua permintaan, yang bisa menjadi titik kegagalan tunggal. Jika beban mencapai batas *server*, ada risiko waktu *respons* yang meningkat atau bahkan kegagalan dalam memproses permintaan.

b. Dengan *Load Balancer*

Load balancer mendistribusikan beban secara merata ke beberapa *server*, yang tidak hanya meningkatkan performa dengan menjaga stabilitas waktu *respons* tetapi juga meningkatkan reliabilitas sistem secara keseluruhan. Sistem lebih tangguh

terhadap beban puncak dan memiliki kemampuan untuk menghindari kegagalan dengan mengalihkan beban ke *server* lain.

Penelitian kami menunjukkan bahwa penggunaan load balancer secara signifikan meningkatkan stabilitas dan keandalan sistem, terutama saat menghadapi beban kerja yang tinggi. Sistem dengan load balancer mampu menjaga performa yang konsisten bahkan di bawah tekanan, dengan fluktuasi minimal dalam jumlah permintaan yang diproses per detik. Meskipun kedua sistem (dengan dan tanpa load balancer) dapat mencapai tingkat pemrosesan permintaan yang sama tanpa kegagalan, sistem dengan load balancer menunjukkan keandalan yang lebih besar karena kemampuannya dalam mendistribusikan beban secara cerdas dan menangani potensi kegagalan server. Load balancer memberikan lapisan perlindungan tambahan, membuat sistem lebih tahan terhadap lonjakan lalu lintas atau masalah perangkat keras yang tidak terduga.

4. Perbandingan Jumlah Request pada Load Balancer

Kategori	Load Balancer 1000	Load Balancer 2000	Load Balancer 3000
Jumlah Users (Users)	1000	2000	3000
Jumlah Request (Requests)	56,789	18,777	18,801
Waktu Respons Rata-rata (ms)	16.29	24.77	43.74
Waktu Respons Minimum (ms)	3.30	3.00	3.00
Waktu Respons Maksimum (ms)	1032.35	279.00	443.00

Kategori	Load Balancer 1000	Load Balancer 2000	Load Balancer 3000
Requests per Second (RPS)	194.24	654.10	880.80
Jumlah Kegagalan (Failures)	0	0	0

Saat menguji coba jumlah pengguna dari 1000 ke 2000 lalu ke 3000, jumlah total permintaan yang diproses oleh system load balancer masih stabil di sekitar 18.800 untuk 2000 dan 3000 pengguna, sementara pada 1000 pengguna, jumlah permintaan jauh lebih tinggi, yaitu 56.789. Ini bisa menunjukkan bahwa ketika pengguna bertambah, beban sistem semakin berat, sehingga jumlah permintaan per pengguna berkurang karena kapasitas yang terbatas. Waktu respons rata-rata juga naik secara bertahap, dari 16,29 ms dengan 1000 pengguna menjadi 24,77 ms dengan 2000 pengguna, dan akhirnya menjadi 43,74 ms dengan 3000 pengguna. Artinya bahwa sistem membutuhkan lebih banyak waktu untuk memproses setiap permintaan seiring dengan meningkatnya beban.

Oleh karena itu, waktu respons minimum tetap stabil di sekitar 3 ms untuk semua pengujian, yang berarti bahwa dalam kondisi optimal, sistem masih bisa merespons dengan cepat meski pengguna bertambah. Namun, ada perubagan dalam waktu respons maksimum, dengan puncak tertinggi tercatat saat pengujian 1000 pengguna di 1032,35 ms. Saat pengguna meningkat menjadi 2000, waktu respons maksimum turun menjadi 279 ms, tetapi naik lagi menjadi 443 ms ketika mencapai 3000 pengguna. Artinya ini menunjukkan bahwa ada beberapa permintaan yang mungkin mengalami keterlambatan yang signifikan di bawah beban berat.

Requests per Second (RPS) terus meningkat seiring bertambahnya pengguna, dari 194,24 dengan 1000 pengguna menjadi 654,1 dengan 2000 pengguna, dan mencapai 880,8 dengan 3000 pengguna, yang menandakan bahwa sistem mampu menangani lebih banyak permintaan per detik dengan baik.Dan yang terpenting bahwa, tidak ada kegagalan yang tercatat selama semua pengujian ini, menunjukkan bahwa sistem tetap stabil dan tidak mengalami error meskipun beban meningkat.

BAB V
PENUTUP

A. Kesimpulan

Penelitian ini berhasil menunjukkan bahwa penerapan *load balancing* dengan metode *round robin* pada sistem pendaftaran beasiswa di Universitas Muhammadiyah Makassar memberikan peningkatan kinerja yang signifikan dalam beberapa aspek utama. Pengujian menunjukkan bahwa:

1. Waktu Respons

   Dengan menggunakan *load balancer*, waktu *respons* rata-rata menjadi lebih konsisten dan stabil dibandingkan dengan sistem tanpa *load balancer*. Sistem tanpa *load balancer* menunjukkan waktu *respons* yang lebih bervariasi, terutama pada permintaan yang lebih lambat, yang menunjukkan beban yang tidak merata pada *server*.

2. Requests per Second (RPS)

   Sistem dengan *load balancer* mencapai tingkat RPS yang serupa dengan sistem tanpa *load balancer*, yaitu sekitar 330 RPS. Namun, *load balancer* membantu dalam mempertahankan tingkat RPS ini dengan lebih stabil, mengurangi fluktuasi yang mungkin terjadi pada sistem tanpa *load balancer*.

3. Stabilitas dan Reliabilitas

   Penggunaan *load balancer* secara signifikan meningkatkan stabilitas dan reliabilitas sistem. Dengan mendistribusikan beban secara merata ke beberapa *server*, risiko kelebihan beban pada satu *server* berkurang, yang mengurangi kemungkinan kegagalan sistem dan memastikan layanan tetap tersedia selama periode beban tinggi.

4. Handling Lonjakan Beban

   Sistem dengan *load balancer* lebih efektif dalam menangani lonjakan beban, dengan sedikit atau tanpa penurunan kinerja yang terlihat dalam waktu *respons* atau RPS, bahkan ketika beban meningkat secara

signifikan. Ini menunjukkan bahwa *load balancer* berperan penting dalam menjaga kinerja sistem selama periode puncak pendaftaran.

B. Saran

Berdasarkan ruang lingkup penelitian ini, beberapa saran untuk pengembangan lebih lanjut adalah sebagai berikut:

1. Optimalisasi Konfigurasi *Load Balancer*:

   Mengingat bahwa penelitian ini fokus pada penerapan metode *round robin*, disarankan untuk terus mengoptimalkan konfigurasi *load balancer*, terutama dalam hal penyesuaian beban kerja yang dinamis. Pemantauan berkelanjutan dan penyesuaian konfigurasi berdasarkan kebutuhan aktual dapat meningkatkan efisiensi sistem lebih lanjut.

2. Pengujian pada Kondisi Beban yang Lebih Beragam

   Penelitian lebih lanjut bisa dilakukan dengan melakukan pengujian pada kondisi beban yang lebih bervariasi, seperti penggunaan metode *load balancing* lainnya (misalnya Least Connections atau IP Hash) untuk melihat apakah ada metode lain yang mungkin lebih sesuai dengan pola akses yang spesifik.

DAFTAR PUSTAKA

Adityarini, E., Nur Ayuni, S., & Aminatus Sa'diah, R. (2021). ANALISIS SENTIMEN TERHADAP ULASAN PRODUK PADA SISTEM PENJUALAN TOKO PUTRA ELEKTRONIK. *Journal of Islamic Business Management Studies (JIBMS)*, *2*(2), 84–98. https://doi.org/10.51875/jibms.v2i2.184

Af'idah, D. I., Dairoh, D., Handayani, S. F., & Pratiwi, R. W. (2021). Pengaruh parameter word2vec terhadap performa deep learning pada klasifikasi sentimen. *Jurnal Informatika: Jurnal Pengembangan IT*, *6*(3), 156-161.

Agung, B. A. I. G. N. (2023). Implementasi Deep Learning untuk ImageClasification menggunakan Arsitektur Convolutional Neural Network (CNN) pada Citra Sampah Hotel (Studi Kasus: Hotel http://eprints.unram.ac.id/id/eprint/41624%0Ahttp://eprints.unram.ac.id/4162

Akib, E. (2020). Pariwisata Dalam Tinjauan Pendidikan: Studi Menuju Era Revolusi Industri. *PUSAKA (Journal of Tourism, Hospitality, Travel and Business Event)*, *2*(1), 1–7. https://doi.org/10.33649/pusaka.v2i1.40

Amalia, P. R. (2021). Analisis Sentimen Berdasarkan Aspek Pada Ulasan Restoran Berbahasa Indonesia Menggunakan Kombinasi Convolutional Neural Network (CNN) dan Contextualized Word Embedding (Doctoral dissertation, Universitas Gadjah Mada).

Arsi, P., & Waluyo, R. (2021). Analisis Sentimen Wacana Pemindahan Ibu Kota Indonesia Menggunakan Algoritma Support Vector Machine (SVM). *Jurnal Teknologi Informasi dan Ilmu Komputer*, *8*(1), 147. https://doi.org/10.25126/jtiik.0813944

Bahits, A., Komarudin, M. F., & Afriani, R. I. (2020). STRATEGI PENGEMBANGAN TEMPAT WISATA RELIGI UNTUK MENINGKATKAN PEREKONOMIAN MASYARAKAT DI GUNUNG SANTRI DESA BOJONEGARA KECAMATAN BOJONEGARA KABUPATEN SERANG BANTEN. *Jurnal Manajemen STIE Muhammadiyah Palopo*, *6*(2), 55. https://doi.org/10.35906/jm001.v6i2.593

Dinata, R. K., Hasdyna, N., & Azizah, N. (2020). *Analisis K-Means Clustering pada Data Sepeda Motor*. *5*(1).

Diponegoro, Sri Suning Kusumawardani, & Indriana Hidayah. (2021). Tinjauan Pustaka Sistematis: Implementasi Metode Deep Learning pada Prediksi Kinerja Murid. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, *10*(2), 131–138. https://doi.org/10.22146/jnteti.v10i2.1417

HELDIANSYAH, M. F. (2022). DETEKSI EMOSI PADA TWEET DENGAN MENGGABUNGKAN CONTEXTUALIZED WORD EMBEDDING DAN CONVOLUTIONAL NEURAL NETWORK (CNN) (Doctoral dissertation, Universitas Gadjah Mada).

Hermanto, D. T., Setyanto, A., & Luthfi, E. T. (2021). *Algoritma LSTM-CNN untuk Sentimen Klasifikasi dengan Word2vec pada Media Online*. *8*(1).

Jihad, M. A. A., Adiwijaya, A., & Astut, W. (2021). Analisis sentimen terhadap ulasan film menggunakan algoritma random forest. eProceedings of Engineering, 8(5).

Khesya, N. (2021). *MENGENAL FLOWCHART DAN PSEUDOCODE DALAM ALGORITMA DAN PEMROGRAMAN*.

Khomsah, S. (2021). Sentiment Analysis On YouTube Comments Using Word2Vec and Random Forest. *Telematika*, *18*(1), 61. https://doi.org/10.31315/telematika.v18i1.4493

Kristiawan, K., & Widjaja, A. (2021). Perbandingan Algoritma Machine Learning dalam Menilai Sebuah Lokasi Toko Ritel. *Jurnal Teknik Informatika dan Sistem Informasi*, *7*(1). https://doi.org/10.28932/jutisi.v7i1.3182

Manalu, D. A., & Gunadi, G. (2022). IMPLEMENTASI METODE DATA MINING K-MEANS CLUSTERING TERHADAP DATA PEMBAYARAN TRANSAKSI MENGGUNAKAN BAHASA PEMROGRAMAN PYTHON PADA CV DIGITAL DIMENSI. *Infotech: Journal of Technology Information*, *8*(1), 43–54. https://doi.org/10.37365/jti.v8i1.131

Manalu, R., & Fikri, A. (2021). *INNOVATIVE: Volume 1 Nomor 2 Tahun 2021 Research & Learning in Primary Education*.

Naquitasia. (2021). *ANALISIS SENTIMEN BERBASIS ASPEK PADA WISATA HALAL DENGAN DEEP LEARNING*.

Ningsih, S. R., Hartama, D., Wanto, A., & Parlina, I. (2019). *Penerapan Sistem Pendukung Keputusan Pada Pemilihan Objek Wisata di Simalungun*.

Nurdin, A., Aji, B. A. S., Bustamin, A., & Abidin, Z. (2020). Perbandingan Kinerja Word Embedding Word2Vec, Glove, Dan Fasttext Pada Klasifikasi Teks. *Jurnal Tekno Kompak*, *14*(2), 74-79.

Pelham, I. (2023). Erd2. Secretory Pathway, 5, 135–135. https://doi.org/10.1093/oso/9780198599425.003.0085

Peryanto, A., Yudhana, A., & Umar, R. (2020). Rancang Bangun Klasifikasi Citra Dengan Teknologi Deep Learning Berbasis Metode Convolutional Neural Network. *Format : Jurnal Ilmiah Teknik Informatika*, *8*(2), 138. https://doi.org/10.22441/format.2019.v8.i2.007

Rachman, F. P., & Santoso, H. (2021). Perbandingan Model Deep Learning untuk Klasifikasi Sentiment Analysis dengan Teknik Natural Languange Processing. *Jurnal Teknologi dan Manajemen Informatika*, *7*(2), 113–121. https://doi.org/10.26905/jtmi.v7i2.6506

R.H. Zer, P. P. P. A. N. W. F. I., Hayadi, B. H., & Damanik, A. R. (2022). PENDEKATAN MACHINE LEARNING MENGGUNAKAN ALGORITMA C4.5 BERBASIS PSO DALAM ANALISA PEMAHAMAN PEMROGRAMAN WEBSITE. *Jurnal Informatika dan Teknik Elektro Terapan*, *10*(3). https://doi.org/10.23960/jitet.v10i3.2700

Samsir, S., Ambiyar, A., Verawardina, U., Edi, F., & Watrianthos, R. (2021). Analisis Sentimen Pembelajaran Daring Pada Twitter di Masa Pandemi COVID-19 Menggunakan Metode Naïve Bayes. *JURNAL MEDIA INFORMATIKA BUDIDARMA*, *5*(1), 157. https://doi.org/10.30865/mib.v5i1.2580

Silitonga, Y. R. (2019). *SISTEM PENDETEKSI BERITA HOAX DI MEDIA SOSIAL DENGAN TEKNIK DATA MINING SCIKIT LEARN*. *4*.

Tilasefana, R. A., & Putra, R. E. (2023). *Penerapan Metode Deep Learning Menggunakan Algoritma CNN Dengan Arsitektur VGG NET Untuk Pengenalan Cuaca*. *05*.

Cahyadi, M. L., Setiawan, H., & Mair, Z. R. (2023). *Analisis PerbandinganKinerja Web Server Nginx dan Litespeed Menggunakan Httperf Dengan Sistem Operasi Debian*. Jurnal Pengembangan Sistem Informasi dan Informatika, 4(2).

Hananta, R. (2023). *Implementasi Load Balancer Service Menggunakan Metode Autoscaling Berbasis Orchestration System*. Tugas Akhir. Program Studi S1 - Teknik Informatika, Fakultas Teknologi Informasi dan Komunikasi, Universitas Semarang.

MAJELIS PENDIDIKAN TINGGI PIMPINAN PUSAT MUHAMMADIYAH
UNIVERSITAS MUHAMMADIYAH MAKASSAR
UPT PERPUSTAKAAN DAN PENERBITAN
Alamat kantor: Jl.Sultan Alauddin NO.259 Makassar 90221 Tlp.0411(866972,881593, Fax.(0411) 865588

بسم الله الرحمن الرحيم

SURAT KETERANGAN BEBAS PLAGIAT

UPT Perpustakaan dan Penerbitan Universitas Muhammadiyah Makassar, Menerangkan bahwa mahasiswa yang tersebut namanya di bawah ini:

Nama : Ismi Salsabila
Nim : 105841101720
Program Studi : Teknik Informatika

Dengan nilai:

No	Bab	Nilai	Ambang Batas
1	Bab 1	8 %	10 %
2	Bab 2	14 %	25 %
3	Bab 3	5 %	10 %
4	Bab 4	0 %	10 %
5	Bab 5	3 %	5 %

Dinyatakan telah lulus cek plagiat yang diadakan oleh UPT- Perpustakaan dan Penerbitan Universitas Muhammadiyah Makassar Menggunakan Aplikasi Turnitin.

Demikian surat keterangan ini diberikan kepada yang bersangkutan untuk dipergunakan sepercanya.

Makassar, 30 Agustus 2024
Mengetahui,

Kepala UPT- Perpustakaan dan Penerbitan,

Nursinah, S.Ag., M.I.P
NBM. 963 591

Jl. Sultan Alauddin no 259 makassar 90222
Telepon (0411)866972,881 593,fax (0411)865 588
Website: www.library.unismuh.ac.id
E-mail : perpustakaan@unismuh.ac.id


BAB I ismi salsabila
105841101720
by Tahap Tutup
Submission date: 30-Aug-2024 10:15AM (UTC+0700)
Submission ID: 2440939963
File name: BAB_I_TURNITIN.docx (25.94K)
Word count: 961
Character count: 6458

BAB I ismi salsabila 105841101720

ORIGINALITY REPORT

8%	8%	0%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	anzdoc.com Internet Source	2%
2	digilib.its.ac.id Internet Source	2%
3	tuporupotupo.blogspot.com Internet Source	2%
4	core.ac.uk Internet Source	2%

Exclude quotes

Exclude bibliography

Exclude matches

LULUS

turnitin

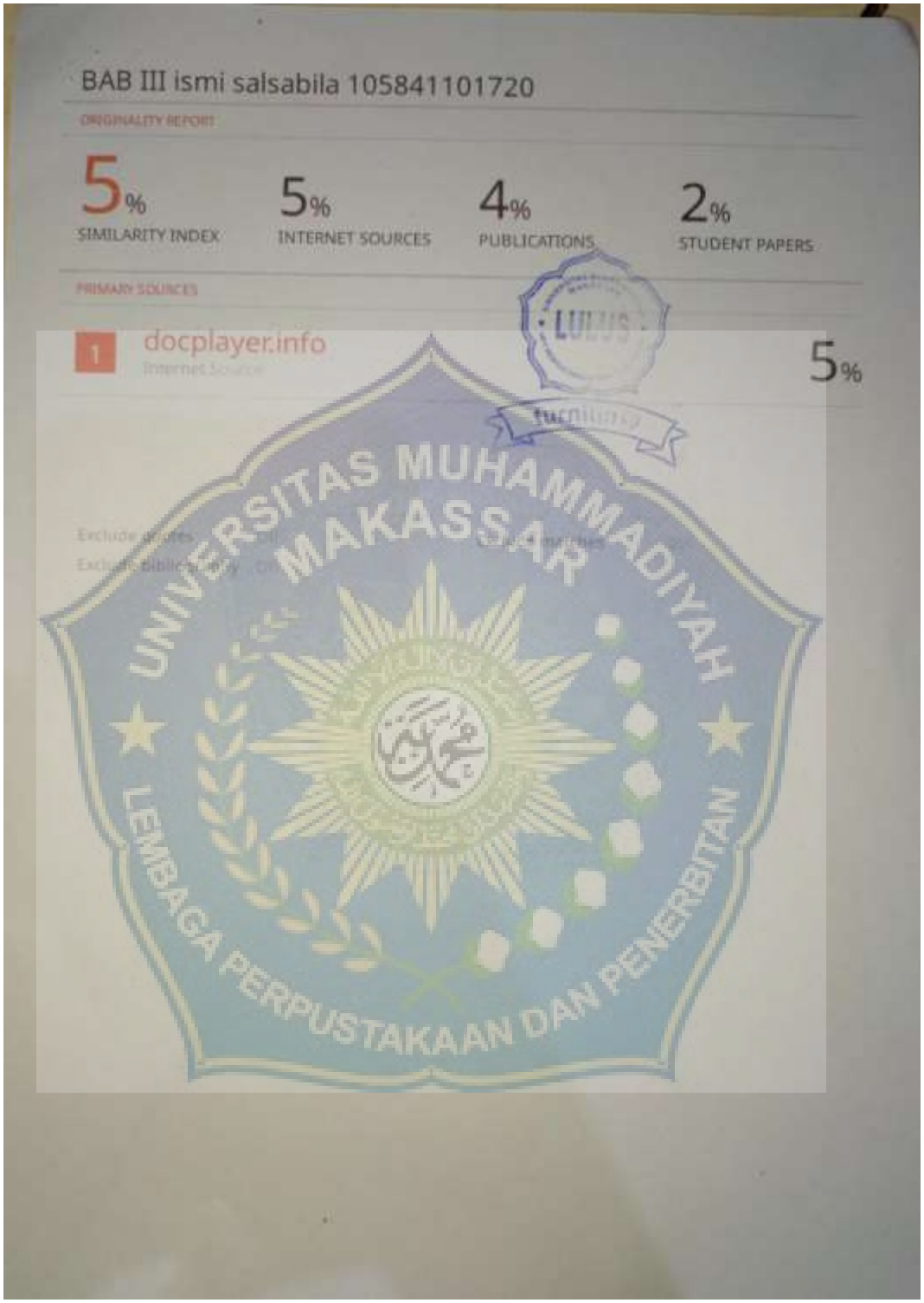
UNIVERSITAS MUHAMMADIYAH MAKASSAR

LEMBAGA PERPUSTAKAAN DAN PENERBITAN


BAB II ismi salsabila
105841101720
by Tahap Tutup
Submission date: 30-Aug-2024 10:15AM (UTC+0700)
Submission ID: 2440940494
File name: BAB_II_TURNITIN.docx (43.19K)
Word count: 997
Character count: 6805


BAB II ismi salsabila 105841101720
ORIGINALITY REPORT
14% SIMILARITY INDEX
14% INTERNET SOURCES
0% PUBLICATIONS
4% STUDENT PAPERS
PRIMARY SOURCES
1 ejournal.poltektegal.ac.id Internet Source 5%
2 jurnal.unsil.ac.id Internet Source 4%
3 garuda.kemdikbud.go.id 4%
2%
LULUS
turnitin
UNIVERSITAS MUHAMMADIYAH MAKASSAR
LEMBAGA PERPUSTAKAAN DAN PENERBITAN


BAB III ismi salsabila
105841101720
by Tahap Tutup
Submission date: 30-Aug-2024 10:16AM (UTC+0700)
Submission ID: 2440941019
File name: BAB_III_9.docx (86.89K)
Word count: 699
Character count: 4693



BAB III ismi salsabila 105841101720

ORIGINALITY REPORT

5%	5%	4%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	docplayer.info Internet Source	5%


BAB IV ismi salsabila
105841101720
by Tahap Tutup
Submission date: 30-Aug-2024 10:17AM (UTC+0700)
Submission ID: 2440941539
File name: BAB_IV_TURNITIN.docx (454.52K)
Word count: 2556
Character count: 16768


BAB IV ismi salsabila 105841101720
ORIGINALITY REPORT
0%
SIMILARITY INDEX
0%
INTERNET SOURCES
0%
PUBLICATIONS
0%
STUDENT PAPERS
PRIMARY SOURCES
LULUS
turnitin
Exclude quotes
Exclude bibliography
UNIVERSITAS MUHAMMADIYAH MAKASSAR
LEMBAGA PERPUSTAKAAN DAN PENERBITAN


BAB V ismi salsabila
105841101720
by Tahap Tutup
Submission date: 30-Aug-2024 10:18AM (UTC+0700)
Submission ID: 2440942270
File name: BAB_V_TURNITIN.docx (15.75K)
Word count: 327
Character count: 2153

BAB V ismi salsabila 105841101720

ORIGINALITY REPORT

3%	3%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	docplayer.info Internet Source	3%

LULUS

turnitin

Exclude quotes

Exclude bibliography

Exclude matches

UNIVERSITAS MUHAMMADIYAH MAKASSAR

LEMBAGA PERPUSTAKAAN DAN PENERBITAN