Department of Digital Business
Journal of Artificial Intelligence and Digital Business (RIGGS)
Homepage: https://journal.ilmudata.co.id/index.php/RIGGS
Vol. 5 No. 2 (2026) pp: 25955-25964
P-ISSN: 2963-9298, e-ISSN: 2963-914X

Kajian Penerapan Load Balancing dan Backup Server untuk Mengurangi Downtime pada Sistem Informasi Akademik Perguruan Tinggi

Dian Maylani Silaban[1], Junita Saranina Br Barus[2], Charles Butar-Butar[3]
[1,2]Universitas Methodist Indonesia,
[3]Universitas Muhammadiyah Sumatera Utara
dianmaylani429@gmail.com[1], junitasaranina661@gmail.com[2], charlesbutar@umsu.ac.id[3]

Abstrak

Sistem Informasi Akademik (SIMAK) mendukung layanan penting perguruan tinggi, seperti pengisian Kartu Rencana Studi, pengelolaan nilai, akses Kartu Hasil Studi, penyusunan jadwal, dan administrasi akademik. Lonjakan akses secara bersamaan pada periode puncak serta ketergantungan pada satu server dapat menimbulkan kelebihan beban, penurunan kinerja, dan downtime. Penelitian ini mengkaji penerapan load balancing dan backup server sebagai strategi infrastruktur untuk meningkatkan ketersediaan, keandalan, dan kontinuitas layanan akademik. Metode yang digunakan adalah studi literatur deskriptif kualitatif melalui pengumpulan, seleksi, perbandingan, dan sintesis penelitian terkini mengenai HAProxy, arsitektur multi-server, replikasi basis data, failover otomatis, pengelolaan pencadangan, pemantauan, dan high availability berbasis Kubernetes. Hasil kajian menunjukkan bahwa load balancing mampu mendistribusikan permintaan pengguna ke beberapa server aplikasi, mencegah penumpukan sumber daya pada satu node, serta mengalihkan trafik ketika server backend mengalami gangguan. Backup server dan basis data tereplikasi melengkapi mekanisme tersebut dengan menjaga ketersediaan data dan mengambil alih layanan ketika server utama gagal. Pencadangan penuh dan inkremental yang terjadwal, health check, pemantauan kinerja, serta pengujian pemulihan berkala diperlukan agar redundansi berfungsi secara efektif. Sintesis penelitian merekomendasikan arsitektur tiga lapis yang mencakup load balancer, server aplikasi tersinkronisasi, dan basis data tereplikasi dengan prosedur backup serta failover otomatis. Integrasi ini dapat mengurangi single point of failure, meningkatkan skalabilitas saat aktivitas akademik memuncak, dan menjadi dasar pengembangan infrastruktur SIMAK perguruan tinggi yang tangguh.

Kata kunci: Sistem Informasi Akademik, Load Balancing, Backup Server, Downtime, High Availability

1. Pendahuluan

Perkembangan teknologi informasi telah mendorong perguruan tinggi untuk memanfaatkan sistem informasi dalam mendukung kegiatan akademik dan administrasi secara lebih efektif. Salah satu sistem yang memiliki peran penting adalah Sistem Informasi Akademik (SIMAK), yang digunakan untuk mengelola berbagai layanan akademik seperti pengisian Kartu Rencana Studi (KRS), pengelolaan nilai, jadwal perkuliahan, akses Kartu Hasil Studi (KHS) dan akses informasi akademik lainnya kepada mahasiswa maupun dosen. Keandalan sistem ini menjadi faktor penting dalam menjamin kelancaran proses akademik di lingkungan perguruan tinggi.

Universitas Methodist Indonesia sebagai salah satu perguruan tinggi yang memanfaatkan teknologi informasi dalam penyelenggaraan layanan akademik juga memerlukan sistem yang mampu memberikan layanan secara cepat, stabil, dan mudah diakses oleh pengguna. Seiring meningkatnya jumlah pengguna dan aktivitas akademik yang dilakukan secara daring, kebutuhan terhadap infrastruktur server yang memiliki tingkat ketersediaan tinggi (high availability) menjadi semakin penting. Pada periode tertentu, seperti pengisian Kartu Rencana Studi (KRS), penginputan nilai, akses Kartu Hasil Studi (KHS), terjadi peningkatan jumlah permintaan akses yang berpotensi menyebabkan penurunan kinerja server bahkan terjadinya downtime.

Menurut (Rahmatulloh & MSN, 2017) ketersediaan infrastruktur teknologi informasi yang kuat dan andal merupakan kebutuhan penting bagi institusi yang mengelola data dalam jumlah besar. Pada penelitian yang dilakukan pada Sistem Informasi Akademik Universitas Siliwangi ditemukan bahwa arsitektur server tunggal sering mengalami overload ketika banyak pengguna mengakses sistem secara bersamaan sehingga menyebabkan layanan menjadi tidak optimal. Penelitian tersebut menunjukkan bahwa penerapan load balancing pada arsitektur

multi-server mampu mendistribusikan permintaan pengguna secara merata sehingga server tidak mengalami kelebihan beban dan mampu melayani hingga 10.000 permintaan tanpa mengalami error.

Permasalahan serupa juga dijelaskan oleh (Wirawan et al., 2024) yang menyatakan bahwa meningkatnya aktivitas pengguna pada sistem informasi akademik, terutama pada saat proses pengisian Kartu Rencana Studi (KRS), menyebabkan server mengalami overload akibat tingginya permintaan data secara bersamaan. Untuk mengatasi kondisi tersebut, diperlukan infrastruktur yang mampu meningkatkan kemampuan server dalam melayani permintaan pengguna serta menjaga kontinuitas layanan ketika terjadi gangguan pada salah satu server. Penelitian tersebut menunjukkan bahwa penerapan load balancing menggunakan HAProxy dapat meningkatkan skalabilitas sistem, fault tolerance, dan mengurangi downtime layanan akademik.

Selain masalah beban akses yang tinggi, ketersediaan layanan juga dipengaruhi oleh kemampuan sistem dalam menghadapi kegagalan server. (Misbullah et al., 2020) menjelaskan bahwa sistem informasi akademik pada umumnya masih bergantung pada satu server basis data sehingga sangat rentan terhadap overload dan gangguan layanan. Oleh karena itu, penerapan mekanisme failover dan server cadangan (backup server) menjadi salah satu solusi yang dapat digunakan untuk menjaga ketersediaan layanan ketika terjadi kegagalan pada server utama. Dengan adanya backup server, proses layanan dapat tetap berjalan karena fungsi server utama dapat dialihkan secara otomatis ke server cadangan.

Perkembangan teknologi infrastruktur modern juga menunjukkan bahwa penerapan konsep high availability melalui clustering dan load balancing mampu meningkatkan keandalan sistem informasi akademik. Pada implementasi Kubernetes Cluster di Sistem Informasi Akademik Universitas Pendidikan Ganesha, arsitektur multi-server dirancang untuk menjaga ketersediaan layanan dan memastikan sistem tetap mampu menangani akses pengguna secara optimal meskipun terjadi gangguan pada salah satu node server.

Berdasarkan berbagai penelitian tersebut, load balancing dan backup server merupakan dua teknologi yang memiliki peran penting dalam meningkatkan keandalan infrastruktur teknologi informasi. Oleh karena itu, artikel ini bertujuan untuk mengkaji penerapan load balancing dan backup server sebagai strategi untuk mengurangi downtime, meningkatkan ketersediaan layanan, serta mendukung keberlangsungan operasional Sistem Informasi Akademik pada perguruan tinggi, termasuk sebagai referensi pengembangan infrastruktur layanan akademik di Universitas Methodist Indonesia.

2. Kajian Pustaka

Sistem Informasi Akademik (SIMAK) merupakan salah satu sistem berbasis teknologi informasi yang memiliki peran penting dalam mendukung kegiatan akademik di perguruan tinggi. Sistem ini digunakan untuk mengelola berbagai aktivitas akademik, seperti pengisian Kartu Rencana Studi (KRS), pengelolaan nilai, penyusunan jadwal perkuliahan, akses Kartu hasil Studi (KHS) hingga penyampaian informasi akademik kepada mahasiswa dan dosen. Seiring dengan meningkatnya jumlah pengguna dan volume data yang dikelola, kebutuhan akan sistem yang mampu memberikan layanan secara cepat, stabil, dan mudah diakses menjadi semakin penting. Menurut (Misbullah et al., 2020), peningkatan jumlah data akademik yang terus terjadi setiap tahun menyebabkan sistem informasi akademik harus mampu mengelola dan menyediakan akses data secara efektif agar proses pelayanan akademik dapat berjalan dengan baik.

Dalam praktiknya, banyak sistem informasi akademik masih menggunakan arsitektur server tunggal yang berpotensi mengalami penurunan kinerja ketika menerima permintaan akses dalam jumlah besar secara bersamaan. Kondisi ini sering terjadi pada saat pelaksanaan registrasi mahasiswa, pengisian KRS, maupun pengumuman hasil studi. Tingginya jumlah akses yang melebihi kapasitas server dapat menyebabkan overload yang berujung pada lambatnya respons sistem bahkan menyebabkan layanan tidak dapat diakses untuk sementara waktu atau yang dikenal sebagai downtime. (Rahmatulloh & MSN, 2017) menjelaskan bahwa penggunaan server tunggal pada sistem informasi akademik berisiko mengalami kelebihan beban ketika banyak pengguna mengakses layanan secara bersamaan sehingga mengganggu ketersediaan layanan bagi pengguna.

Salah satu teknologi yang banyak digunakan untuk mengatasi permasalahan tersebut adalah load balancing. Load balancing merupakan teknik yang digunakan untuk mendistribusikan beban kerja atau permintaan pengguna ke beberapa server sehingga tidak terjadi penumpukan beban pada satu server tertentu. Dengan adanya distribusi beban yang merata, penggunaan sumber daya server menjadi lebih optimal dan risiko terjadinya overload dapat diminimalkan. (Rahmatulloh & MSN, 2017) menyatakan bahwa load balancing berfungsi untuk mendistribusikan

DOI: https://doi.org/10.31004/riggs.v5i2.11765

trafik ke beberapa server secara seimbang sehingga dapat meningkatkan performa sistem dan menghindari terjadinya kelebihan kapasitas pada salah satu server.

Penelitian yang dilakukan oleh (Wirawan et al., 2024) menunjukkan bahwa penerapan load balancing menggunakan HAProxy pada sistem informasi akademik mampu meningkatkan skalabilitas sistem dan toleransi terhadap kesalahan (fault tolerance). Selain itu, penerapan load balancing juga terbukti dapat mengurangi downtime serta meningkatkan pengalaman pengguna dalam mengakses layanan akademik. Dengan membagi permintaan pengguna ke beberapa server backend, beban kerja yang diterima masing-masing server menjadi lebih seimbang sehingga kinerja sistem dapat dipertahankan meskipun terjadi peningkatan jumlah pengguna.

Selain load balancing, keberadaan backup server juga memiliki peran penting dalam menjaga ketersediaan layanan sistem informasi akademik. Backup server merupakan server cadangan yang disiapkan untuk mengambil alih layanan ketika server utama mengalami gangguan atau kegagalan. Penerapan backup server biasanya dikombinasikan dengan mekanisme failover, yaitu proses pengalihan layanan secara otomatis dari server utama ke server cadangan ketika terjadi kegagalan sistem. Dengan adanya failover, layanan dapat tetap berjalan tanpa menyebabkan gangguan yang signifikan bagi pengguna. Menurut (Misbullah et al., 2020), penerapan cluster database dan mekanisme failover mampu meningkatkan keandalan sistem karena beban akses tidak lagi bergantung pada satu server saja, sehingga risiko terjadinya gangguan layanan dapat dikurangi.

Konsep load balancing dan backup server pada dasarnya merupakan bagian dari penerapan high availability, yaitu upaya untuk menjaga agar layanan teknologi informasi tetap tersedia dengan tingkat downtime yang rendah. High availability menjadi aspek yang sangat penting dalam pengelolaan sistem informasi akademik karena layanan tersebut digunakan oleh mahasiswa, dosen, dan tenaga kependidikan secara terus-menerus. Penelitian (Pradipta et al., 2026) menunjukkan bahwa penggunaan arsitektur multi-server dan clustering dapat meningkatkan ketersediaan layanan serta menjaga performa sistem ketika terjadi gangguan pada salah satu server. Pendekatan ini memungkinkan sistem tetap beroperasi dan melayani pengguna secara optimal meskipun terjadi kegagalan pada sebagian komponen infrastruktur.

Berdasarkan berbagai penelitian terdahulu tersebut, dapat dipahami bahwa load balancing dan backup server merupakan dua teknologi yang saling melengkapi dalam meningkatkan keandalan infrastruktur teknologi informasi. Load balancing berperan dalam mendistribusikan beban akses secara merata untuk mencegah overload, sedangkan backup server berfungsi menjaga kontinuitas layanan ketika terjadi kegagalan pada server utama. Oleh karena itu, penerapan kedua teknologi tersebut dapat menjadi strategi yang efektif untuk mengurangi downtime dan meningkatkan kualitas layanan Sistem Informasi Akademik pada perguruan tinggi, termasuk sebagai referensi pengembangan infrastruktur layanan akademik di Universitas Methodist Indonesia.

3. Metode Penelitian

Metode yang digunakan dalam penelitian ini adalah studi literatur (literature review) dengan pendekatan deskriptif kualitatif. Penelitian ini berfokus pada pengumpulan, penelaahan, dan penyintesisan data dari sejumlah penelitian terdahulu yang secara spesifik membahas implementasi arsitektur jaringan komputer, load balancing, backup server, failover, dan high availability pada sistem informasi. Tahapan pelaksanaan studi literatur ini dilakukan melalui langkah-langkah terstruktur sebagai berikut:

1. Identifikasi Masalah
   Tahap awal dilakukan dengan menganalisis fenomena tingginya beban akses pada arsitektur single server Sistem Informasi Akademik (SIMAK) selama periode puncak seperti pengisian KRS, yang kerap memicu terjadinya overload dan downtime layanan sebagaimana melatarbelakangi penelitian(Rahmatulloh & MSN, 2017) serta (Wirawan et al., 2024).

2. Pengumpulan dan Seleksi Literatur
   Data dan materi pustaka dikumpulkan dari jurnal-jurnal ilmiah yang berfokus pada optimasi infrastruktur server. Literatur yang dikaji dipilih berdasarkan kesesuaian topik dan studi kasus pada pengelolaan sistem informasi, yang meliputi:

   - Penerapan load balancing menggunakan HAProxy pada multi-server akademik(Rahmatulloh & MSN, 2017).
   - Migrasi dan klasterisasi basis data untuk menangani failover(Misbullah et al., 2020).

DOI: https://doi.org/10.31004/riggs.v5i2.11765

- Penggunaan teknologi kontainer modern melalui Kubernetes Cluster untuk ketersediaan tinggi(Pradipta et al., 2026).
- Manajemen teknik pencadangan data (backup) untuk penyelamatan data dari kerusakan server(Rosano & Sudaradjat, 2020).
- Penerapan metode pembagian beban dan failover pada jaringan instansi sektoral(Syaputra & Assegaff, 2017).

3. Analisis dan Sintesis Data
   Literatur yang telah dikumpulkan kemudian diekstraksi secara kualitatif. Analisis dilakukan dengan membandingkan keandalan arsitektur server tunggal dengan arsitektur multi-server, serta menelaah efektivitas kombinasi teknologi load balancing dan backup server dalam mereduksi downtime.

4. Perumusan Rekomendasi
   Hasil akhir dari sintesis berbagai literatur tersebut digunakan untuk merumuskan sebuah rekomendasi strategi arsitektur infrastruktur yang andal dan efektif, yang diharapkan dapat diterapkan pada Sistem Informasi Akademik Universitas Methodist Indonesia.

4. Hasil dan Pembahasan

Berdasarkan penelaahan terhadap studi kasus di Universitas Siliwangi oleh (Rahmatulloh & MSN, 2017) serta di UIN Sunan Kalijaga oleh (Wirawan et al., 2024), ditemukan bahwa kendala utama kelumpuhan sistem akademik pada umumnya bersumber dari penggunaan arsitektur peladen tunggal (single server). Pada model ini, seluruh layanan penting mulai dari web server hingga database server dijalankan dalam satu mesin yang sama. Ketika memasuki periode sibuk seperti pengisian KRS atau pengumuman KHS, lonjakan permintaan dari ribuan mahasiswa secara bersamaan memicu kelebihan beban (overload) pada CPU dan RAM. Kondisi tersebut berujung pada lambatnya respons sistem (request timeout) hingga kegagalan total layanan (downtime). Selain rentan kelebihan beban, model peladen tunggal ini menciptakan titik rawan kegagalan (Single Point of Failure), di mana kerusakan pada salah satu komponen terkecil peladen akan langsung melumpuhkan seluruh operasional akademik institusi secara total.

Untuk mengeliminasi risiko tersebut dan memecah beban akses pengguna, penerapan teknologi load balancing menggunakan perangkat lunak HAProxy terbukti menjadi solusi yang sangat efektif. HAProxy berfungsi sebagai gerbang utama yang bertindak untuk mendistribusikan trafik masuk secara merata ke beberapa peladen aplikasi (backend web servers) dengan mengandalkan algoritma seperti Round Robin atau Least Connections. Sistem penyeimbang beban ini juga dilengkapi dengan fitur health check dinamis yang mampu memonitor status kesehatan setiap peladen backend secara berkala. Jika salah satu peladen backend mengalami kegagalan, HAProxy akan mengalihkan trafik secara instan ke peladen lain yang normal sehingga pengguna tidak merasakan adanya gangguan (zero-downtime). Selain pada tingkat aplikasi, pembagian beban trafik data juga dapat dioptimalkan pada tingkat jaringan menggunakan metode NTH pada router untuk mengantisipasi penyumbatan pita lebar (bandwidth choking) saat masa sibuk berlangsung(Syaputra & Assegaff, 2017).

Optimasi yang telah dilakukan di sisi layer aplikasi tersebut tentu harus diimbangi dengan keandalan pada layer penyimpanan basis data. Merujuk pada penelitian yang dilakukan oleh (Misbullah et al., 2020), migrasi dari basis data tunggal ke sistem basis data terklaster (database cluster) mampu meningkatkan kecepatan akses kueri dan keandalan sistem akademik. Arsitektur ini menggunakan konfigurasi peladen Master sebagai pemroses utama dan peladen Slave sebagai peladen cadangan (backup server). Apabila peladen Master mengalami kerusakan, mekanisme failover otomatis akan langsung memindahkan peran operasional penuh ke backup server dalam hitungan detik sehingga ketersediaan data KRS dan KHS mahasiswa tetap terjaga.

Keberadaan backup server ini wajib didukung oleh manajemen pencadangan data yang terstruktur demi mengamankan data dari risiko korup atau kehilangan. Mengacu pada strategi yang dipaparkan oleh (Rosano & Sudaradjat, 2020), kombinasi metode full backup yang dijadwalkan secara berkala pada akhir pekan saat trafik rendah, dan incremental backup untuk mencatat perubahan data harian, merupakan pilihan yang paling optimal karena sangat ekonomis dalam penggunaan ruang penyimpanan namun tetap menjamin aktualitas data saat proses pemulihan (disaster recovery). Di samping arsitektur konvensional, konsep ketersediaan tinggi saat ini juga dapat dikembangkan melalui teknologi kontainerisasi menggunakan Kubernetes Cluster(Pradipta et al., 2026). Kubernetes mengintegrasikan manajemen multi-server dengan keunggulan fitur auto-scaling untuk menggandakan

DOI: https://doi.org/10.31004/riggs.v5i2.11765

peladen cadangan secara otomatis saat trafik melonjak, serta fitur self-healing yang mampu memperbaiki kontainer peladen yang error secara mandiri tanpa intervensi manual dari administrator.

Hasil sintesis komprehensif dari seluruh literatur terdahulu tersebut pada akhirnya menghasilkan sebuah cetak biru rekomendasi arsitektur tiga lapis (Three-Tier Architecture) yang dapat diimplementasikan untuk mengoptimalkan SIMAK pada Universitas Methodist Indonesia. Pada lapisan pertama yaitu penyeimbang beban, institusi dapat menempatkan HAProxy sebagai penentu distribusi trafik sekaligus menerapkan metode NTH pada Mikrotik untuk menjamin redundansi jalur internet dari beberapa penyedia jasa internet. Kemudian pada lapisan kedua atau lapisan aplikasi, pemrosesan situs web SIMAK dipecah ke dalam dua atau tiga Virtual Web Server backend yang saling tersinkronisasi agar konten yang diakses mahasiswa tetap konsisten. Terakhir pada lapisan ketiga atau lapisan basis data, peladen database dipisahkan secara fisik dari peladen aplikasi dengan menerapkan sistem replikasi Master-Slave demi mengaktifkan fitur failover otomatis, yang diperkuat dengan kebijakan otomatisasi incremental backup harian serta full backup mingguan. Melalui integrasi berlapis antara load balancing dan penyediaan backup server ini, SIMAK Universitas Methodist Indonesia akan memiliki ketahanan sistem yang tinggi dan mampu mereduksi risiko downtime secara signifikan.

Hasil kajian memperlihatkan bahwa persoalan downtime pada Sistem Informasi Akademik tidak hanya disebabkan oleh keterbatasan spesifikasi perangkat keras, tetapi juga oleh rancangan arsitektur yang menempatkan seluruh komponen layanan pada satu titik. Server tunggal harus menjalankan proses autentikasi, aplikasi web, penyimpanan berkas, pemrosesan kueri, dan pengelolaan sesi pengguna secara bersamaan. Pada kondisi normal, arsitektur tersebut mungkin masih mampu memberikan respons yang memadai. Akan tetapi, ketika ribuan mahasiswa membuka modul KRS atau KHS pada waktu yang hampir sama, permintaan yang masuk meningkat secara tidak linear karena satu aktivitas pengguna dapat menghasilkan beberapa permintaan HTTP dan kueri basis data. Beban yang terkonsentrasi menyebabkan antrean proses, peningkatan penggunaan CPU dan memori, serta waktu respons yang semakin panjang. Kondisi tersebut memperkuat argumentasi bahwa peningkatan kapasitas secara vertikal saja belum cukup, sebab kegagalan pada satu mesin tetap menyebabkan seluruh layanan berhenti. Arsitektur multi-server diperlukan untuk memisahkan fungsi, membagi beban, dan menghilangkan ketergantungan pada satu titik kegagalan.

Karakteristik beban SIMAK juga berbeda dengan aplikasi yang memiliki trafik relatif merata sepanjang hari. Aktivitas akademik bersifat musiman dan sangat dipengaruhi kalender institusi. Trafik dapat meningkat tajam pada pembukaan registrasi, pengisian KRS, perubahan kelas, pembayaran, penginputan nilai, dan pengumuman hasil studi. Oleh karena itu, perencanaan kapasitas harus menggunakan pola beban puncak, bukan hanya rata-rata penggunaan harian. Pengukuran jumlah permintaan per detik, pengguna aktif bersamaan, waktu respons persentil, rasio kegagalan, konsumsi CPU, penggunaan memori, dan waktu kueri menjadi dasar untuk menentukan jumlah server backend. Kajian mengenai pengujian sistem bertrafik tinggi menunjukkan bahwa load balancing akan memberikan manfaat optimal apabila kapasitas masing-masing node diketahui dan algoritma distribusi disesuaikan dengan karakteristik permintaan. Dengan demikian, institusi perlu melakukan load testing sebelum periode akademik utama agar batas kapasitas sistem dapat diketahui, konfigurasi dapat disesuaikan, dan penambahan sumber daya tidak dilakukan secara reaktif setelah layanan mengalami gangguan.

Pada lapisan aplikasi, HAProxy dapat ditempatkan sebagai reverse proxy sekaligus titik distribusi trafik. Setiap permintaan pengguna terlebih dahulu diterima oleh HAProxy, kemudian diteruskan ke server backend yang tersedia berdasarkan algoritma yang dipilih. Penelitian pada Sistem Informasi Akademik UIN Sunan Kalijaga menunjukkan bahwa integrasi HAProxy dan sinkronisasi berkas dapat meningkatkan skalabilitas, toleransi kesalahan, dan kesinambungan layanan akademik (Wirawan et al., 2024). Temuan tersebut relevan karena aplikasi akademik umumnya memiliki kode program, berkas konfigurasi, dan aset yang harus konsisten pada seluruh node. Apabila satu backend memiliki versi aplikasi yang berbeda, pengguna dapat menerima keluaran yang tidak seragam. Karena itu, implementasi load balancing perlu disertai mekanisme deployment terpusat, sinkronisasi file, atau penggunaan image aplikasi yang sama agar setiap server memproses permintaan dengan logika yang identik.

Pemilihan algoritma load balancing perlu mempertimbangkan kesetaraan kapasitas server dan variasi lama proses setiap permintaan. Round Robin sesuai untuk backend yang memiliki spesifikasi relatif sama karena permintaan dibagikan secara bergiliran. Weighted Round Robin lebih tepat ketika kapasitas CPU, memori, atau jaringan antarserver berbeda, sebab node yang lebih kuat dapat menerima proporsi trafik lebih besar. Penelitian Amalia dan Riskiono (2025) menunjukkan bahwa penerapan Weighted Round Robin dengan HAProxy dapat meningkatkan efisiensi distribusi permintaan dibandingkan kondisi tanpa load balancing. Sementara itu, Least Connection dan Weighted Least Connection lebih adaptif untuk transaksi yang durasinya tidak seragam karena

DOI: https://doi.org/10.31004/riggs.v5i2.11765

trafik baru diarahkan ke server dengan koneksi aktif paling sedikit. Analisis performansi HAProxy pada lingkungan web bertrafik tinggi juga menunjukkan pentingnya menyesuaikan algoritma dengan kapasitas aktual backend (Muhammad Algazali et al., 2025; Laksono & Riskiono, 2025). Bagi SIMAK, Weighted Least Connection dapat dipertimbangkan ketika proses seperti cetak KHS atau pengolahan laporan membutuhkan waktu lebih lama daripada akses halaman informasi biasa.

Load balancer tidak cukup hanya membagi permintaan, tetapi harus mampu mengetahui kesehatan setiap backend. Health check aktif dilakukan dengan mengirimkan permintaan berkala ke endpoint tertentu dan memeriksa apakah aplikasi, koneksi basis data, serta layanan pendukung masih berfungsi. Apabila pemeriksaan gagal beberapa kali, node harus dikeluarkan sementara dari kumpulan server agar pengguna tidak diarahkan ke layanan yang bermasalah. Setelah node pulih dan memenuhi ambang pemeriksaan, server dapat dimasukkan kembali secara otomatis. Mekanisme ini menghasilkan failover pada lapisan aplikasi dan mengurangi dampak kegagalan sebagian. Namun, endpoint health check tidak boleh hanya mengembalikan status web server; pemeriksaan ideal perlu memastikan bahwa aplikasi dapat membaca basis data, mengakses penyimpanan, dan menjalankan fungsi utama. Integrasi pemantauan dengan Grafana dan notifikasi juga mempermudah administrator mengidentifikasi perubahan response time, error rate, dan penggunaan sumber daya sebelum gangguan berkembang menjadi downtime (Pentanugraha et al., 2024).

Aspek sesi pengguna perlu diperhatikan karena mahasiswa biasanya melakukan beberapa langkah berurutan setelah login. Jika informasi sesi hanya disimpan pada memori lokal salah satu backend, permintaan berikutnya yang diarahkan ke node lain dapat menyebabkan pengguna keluar dari sistem atau kehilangan data sementara. Terdapat dua pendekatan yang dapat digunakan. Pertama, sticky session mengarahkan pengguna yang sama ke backend yang sama selama sesi berlangsung. Pendekatan ini mudah diterapkan, tetapi dapat menimbulkan distribusi beban yang kurang merata dan sesi tetap terganggu ketika server tujuan gagal. Kedua, penyimpanan sesi terpusat menggunakan Redis, database, atau layanan session store memungkinkan setiap backend membaca sesi yang sama. Pendekatan kedua lebih mendukung failover dan horizontal scaling, sehingga lebih sesuai untuk SIMAK yang membutuhkan kontinuitas transaksi. Pemisahan session state dari server aplikasi juga mempermudah penambahan dan pengurangan node tanpa mengubah pengalaman pengguna.

Pada lapisan basis data, konsep backup server harus dibedakan dengan pencadangan data. Server replika menyediakan salinan basis data yang terus diperbarui dan dapat mengambil alih operasi ketika server utama gagal, sedangkan backup merupakan salinan historis yang digunakan untuk memulihkan data akibat penghapusan, korupsi, serangan, atau kesalahan aplikasi. Replikasi saja tidak melindungi sistem dari kesalahan logis karena perubahan yang salah dapat ikut direplikasi ke node cadangan. Sebaliknya, backup tanpa replika mungkin mampu mengembalikan data, tetapi proses pemulihan memerlukan waktu dan menyebabkan layanan berhenti. Oleh karena itu, ketersediaan tinggi memerlukan kombinasi replikasi, failover, dan backup. Penerapan master-to-master replication berbasis Docker pada skenario disaster recovery menunjukkan bahwa replikasi dapat mempercepat pengalihan fungsi database ketika server utama mengalami gangguan (Suhatman et al., 2024).

Desain replikasi basis data harus disesuaikan dengan kebutuhan konsistensi transaksi akademik. Data KRS, nilai, pembayaran, dan status registrasi memiliki tingkat kritikal yang tinggi sehingga inkonsistensi dapat menimbulkan kerugian administratif. Replikasi sinkron menjaga agar transaksi dikonfirmasi setelah perubahan tersimpan pada lebih dari satu node, sehingga kehilangan data dapat ditekan, tetapi menambah latensi dan bergantung pada kualitas jaringan. Replikasi asinkron memiliki latensi lebih rendah karena server utama tidak menunggu replika, tetapi terdapat risiko sebagian transaksi terbaru belum tersalin ketika kegagalan terjadi. Institusi perlu menetapkan Recovery Point Objective (RPO), yaitu toleransi kehilangan data, dan Recovery Time Objective (RTO), yaitu batas waktu pemulihan layanan. Modul kritikal dapat menggunakan konfigurasi yang lebih ketat, sedangkan modul informatif dapat menggunakan replikasi asinkron untuk efisiensi. Mekanisme promosi replika menjadi server utama juga harus diuji agar tidak menimbulkan split brain atau dua node yang menerima penulisan secara bersamaan.

Strategi pencadangan yang efektif perlu mencakup full backup, incremental backup, retensi, enkripsi, dan salinan di lokasi berbeda. Full backup memberikan salinan lengkap dan mempermudah proses pemulihan, tetapi membutuhkan ruang penyimpanan serta waktu lebih besar. Incremental backup hanya menyimpan perubahan sejak pencadangan sebelumnya sehingga lebih efisien untuk dilakukan setiap hari atau beberapa kali sehari. Kombinasi full backup mingguan dan incremental backup harian dapat digunakan sebagai dasar, kemudian disesuaikan dengan volume perubahan data dan target RPO. Salinan backup sebaiknya tidak hanya berada pada storage yang terhubung langsung dengan server produksi karena kerusakan fisik, kesalahan konfigurasi, atau serangan

DOI: https://doi.org/10.31004/riggs.v5i2.11765

ransomware dapat memengaruhi keduanya. Prinsip pemisahan media dan lokasi, verifikasi checksum, serta enkripsi data perlu diterapkan. Hal yang paling penting adalah melakukan restore test secara berkala, sebab keberhasilan proses backup tidak otomatis menjamin bahwa data dapat dipulihkan dengan benar.

Ketersediaan layanan juga dipengaruhi oleh observabilitas. Administrator membutuhkan data real-time mengenai throughput, latensi, error rate, penggunaan CPU, memori, ruang penyimpanan, koneksi database, replication lag, dan status backup. Monitoring tanpa ambang peringatan hanya menghasilkan kumpulan grafik, sedangkan sistem peringatan yang baik harus menghubungkan indikator teknis dengan dampak layanan. Misalnya, peningkatan response time pada endpoint pengisian KRS, bertambahnya koneksi database, dan tingginya antrean permintaan dapat menjadi tanda awal bahwa kapasitas mendekati batas. Pemantauan terintegrasi memungkinkan tindakan proaktif seperti menambah backend, membatasi proses laporan berat, atau mengalihkan pekerjaan nonprioritas. Penelitian Pentanugraha et al. (2024) memperlihatkan manfaat integrasi HAProxy, Grafana, dan notifikasi dalam mengevaluasi performa server. Untuk lingkungan perguruan tinggi, dashboard sebaiknya dipisahkan antara metrik operasional harian dan indikator layanan kritis yang perlu segera ditangani.

Pengujian performa perlu dilakukan dengan skenario yang merepresentasikan perilaku nyata pengguna. Uji sederhana yang hanya membuka halaman utama belum cukup untuk menilai kesiapan SIMAK. Skenario harus mencakup login, pencarian mata kuliah, pemilihan kelas, penyimpanan KRS, pembatalan mata kuliah, akses KHS, dan unduh laporan. Pengujian dilakukan secara bertahap melalui baseline test, load test, stress test, spike test, endurance test, dan failover test. Baseline mengukur kinerja normal, load test menilai target pengguna bersamaan, stress test menemukan batas kapasitas, spike test mensimulasikan lonjakan tiba-tiba, sedangkan endurance test mendeteksi kebocoran memori atau penurunan kinerja dalam waktu panjang. Failover test dilakukan dengan mematikan backend, load balancer, atau database utama secara terkontrol untuk mengukur waktu pengalihan. Hasil pengujian menjadi dasar penetapan Service Level Indicator dan Service Level Objective, bukan hanya asumsi bahwa sistem multi-server selalu tersedia.

Walaupun HAProxy meningkatkan ketersediaan backend, penempatan hanya satu load balancer masih membentuk single point of failure baru. Karena itu, lapisan load balancer idealnya terdiri atas dua node dengan mekanisme virtual IP atau protokol redundansi yang dapat mengalihkan alamat layanan ketika node aktif gagal. Konfigurasi active-passive lebih sederhana dan sesuai untuk institusi dengan sumber daya terbatas, sedangkan active-active dapat meningkatkan kapasitas tetapi membutuhkan pengaturan yang lebih kompleks. Selain redundansi perangkat, konfigurasi HAProxy, sertifikat TLS, daftar backend, dan aturan keamanan harus disinkronkan. Jalur internet juga perlu dipertimbangkan karena server yang sehat tetap tidak dapat diakses apabila koneksi utama terputus. Redundansi penyedia internet, monitoring konektivitas, dan kebijakan routing yang jelas memperkuat ketersediaan end-to-end, bukan hanya ketersediaan server di pusat data.

Teknologi kontainer dan Kubernetes memberikan pilihan pengelolaan yang lebih dinamis untuk aplikasi SIMAK. Kubernetes dapat menjalankan beberapa replika aplikasi, melakukan service discovery, mendistribusikan trafik, mengganti pod yang gagal, dan menambah replika berdasarkan metrik. Penelitian Nguyen et al. (2022) menunjukkan bahwa mekanisme load balancing berbasis Kubernetes dapat ditingkatkan dengan mempertimbangkan kondisi sumber daya setiap node. Sementara itu, Mondal et al. (2023) mengembangkan pendekatan prediksi beban dan autoscaling yang dapat disesuaikan untuk mengatasi fluktuasi trafik. Bagi perguruan tinggi, Kubernetes bermanfaat ketika aplikasi telah dikemas secara konsisten dan tim teknis memiliki kemampuan mengelola cluster. Namun, kompleksitas operasional, kebutuhan monitoring, pengelolaan storage stateful, pembaruan cluster, dan keamanan tidak boleh diabaikan. Implementasi sebaiknya didasarkan pada kebutuhan nyata, bukan sekadar mengikuti tren teknologi.

Autoscaling reaktif umumnya menambah replika setelah penggunaan CPU atau memori melewati ambang tertentu. Pada lonjakan KRS yang sangat cepat, penambahan pod mungkin terlambat karena proses penjadwalan, penarikan image, dan inisialisasi aplikasi memerlukan waktu. Penelitian Augustyn et al. (2024) menunjukkan pentingnya penyetelan parameter Horizontal Pod Autoscaler agar kebutuhan kinerja dapat dipenuhi tanpa pemborosan sumber daya. Yuan dan Liao (2024) juga memperlihatkan bahwa prediksi deret waktu dapat digunakan untuk menambah kapasitas sebelum beban mencapai puncak. Dalam konteks SIMAK, kalender akademik menyediakan informasi yang kuat untuk scaling terjadwal. Institusi dapat meningkatkan jumlah replika sebelum pembukaan KRS, kemudian menurunkannya setelah periode sibuk. Pendekatan terjadwal dapat dikombinasikan dengan autoscaling berbasis metrik agar sistem tetap responsif terhadap variasi yang tidak diperkirakan.

DOI: https://doi.org/10.31004/riggs.v5i2.11765

Ketahanan sistem perlu dibangun melalui pendekatan defence in depth. Load balancing dan backup server tidak akan efektif apabila akses administratif tidak dilindungi, credential disimpan pada kode aplikasi, atau port basis data terbuka ke jaringan publik. Segmentasi jaringan harus memisahkan lapisan load balancer, aplikasi, database, monitoring, dan backup. Akses antarsegmen dibatasi berdasarkan kebutuhan minimum. Komunikasi sensitif menggunakan enkripsi, akun administrator menerapkan autentikasi berlapis, dan seluruh perubahan konfigurasi dicatat. Backup juga harus dilindungi dari modifikasi tidak sah dan diuji integritasnya. Risiko kegagalan cloud dan perangkat lunak dapat berasal dari sumber yang beragam, sehingga prediksi serta deteksi dini menjadi bagian penting dari reliability engineering (Tengku Asmawi et al., 2022). Dengan demikian, high availability tidak hanya berarti server tetap menyala, tetapi juga memastikan bahwa layanan yang tersedia tetap benar, aman, dan menjaga integritas data akademik.

Fault tolerance perlu diuji melalui simulasi kegagalan yang terkendali. Pengujian dapat dimulai dengan menghentikan satu backend ketika trafik berlangsung, kemudian mengamati apakah HAProxy mengeluarkan node tersebut, berapa banyak permintaan yang gagal, dan berapa lama sistem kembali stabil. Tahap berikutnya adalah memutus koneksi database utama, mempromosikan replika, dan memverifikasi konsistensi transaksi. Penelitian mengenai scalability resilience menunjukkan bahwa injeksi gangguan pada tingkat aplikasi dapat digunakan untuk menilai kemampuan layanan cloud bertahan ketika beban dan kegagalan terjadi bersamaan (Ahmad & Andras, 2022). Pendekatan fault-tolerant aware load balancing juga menekankan bahwa distribusi trafik perlu memperhitungkan kegagalan jaringan, bukan hanya kondisi beban (Alawadi & Molnar, 2022). Bagi SIMAK, pengujian semacam ini sebaiknya dilakukan di lingkungan staging agar tim memperoleh prosedur penanganan tanpa mengganggu data dan layanan produksi.

Arsitektur tiga lapis yang direkomendasikan dapat dijabarkan sebagai berikut. Lapisan pertama terdiri atas dua node load balancer yang menyediakan virtual IP, terminasi TLS, health check, rate limiting, dan distribusi trafik. Lapisan kedua terdiri atas sedikitnya dua server aplikasi dengan versi kode yang sama, konfigurasi terkelola, session store terpusat, dan akses terbatas ke basis data. Lapisan ketiga terdiri atas database utama dan replika dengan mekanisme failover, penyimpanan transaksi, pencadangan terjadwal, serta salinan off-site. Sistem monitoring mengumpulkan metrik dari seluruh lapisan dan mengirim peringatan kepada tim teknis. Dengan susunan ini, kegagalan satu backend tidak menghentikan layanan, kegagalan database utama dapat dialihkan ke replika, dan data masih dapat dipulihkan ketika terjadi kesalahan logis. Desain tersebut juga memungkinkan peningkatan kapasitas secara horizontal tanpa mengganti seluruh infrastruktur.

Penerapan pada Universitas Methodist Indonesia dapat dilakukan secara bertahap untuk mengurangi risiko migrasi. Tahap pertama adalah melakukan inventarisasi aplikasi, dependensi, volume data, pola trafik, serta titik kegagalan saat ini. Tahap kedua membangun lingkungan uji dengan satu HAProxy dan dua backend, kemudian memindahkan session state serta mengotomatisasi deployment. Tahap ketiga menerapkan replikasi database dan menjalankan pengujian failover. Tahap keempat menambahkan monitoring, sentralisasi log, kebijakan backup, dan restore test. Tahap kelima membangun redundansi load balancer serta jaringan. Setelah sistem stabil, institusi dapat mempertimbangkan kontainerisasi dan autoscaling. Model bertahap membuat biaya dapat dikendalikan dan memberikan kesempatan kepada tim untuk membangun kompetensi operasional. Setiap tahap harus memiliki kriteria penerimaan, dokumentasi konfigurasi, rencana rollback, serta evaluasi keamanan.

Dari sisi biaya, implementasi multi-server memang membutuhkan tambahan perangkat, penyimpanan, lisensi pendukung, dan waktu pengelolaan. Namun, analisis tidak boleh hanya membandingkan biaya pembelian server dengan kondisi saat ini. Downtime pada periode KRS dapat menimbulkan antrean layanan, keluhan pengguna, pekerjaan ulang, keterlambatan akademik, dan penurunan kepercayaan terhadap institusi. Beban kerja tim teknologi informasi juga meningkat ketika pemulihan dilakukan secara manual tanpa prosedur. Penggunaan virtualisasi memungkinkan beberapa node dijalankan pada perangkat fisik yang tersedia, sedangkan perangkat lunak open-source seperti HAProxy, Prometheus, Grafana, dan Kubernetes dapat mengurangi biaya lisensi. Meskipun demikian, penghematan perangkat lunak harus diimbangi dengan investasi pada sumber daya manusia, dokumentasi, serta pemeliharaan. Arsitektur yang terlalu kompleks tanpa kemampuan operasional justru dapat meningkatkan risiko.

Penelitian load balancing di lingkungan cloud menunjukkan bahwa tujuan distribusi beban tidak hanya mengejar throughput, tetapi juga menjaga kualitas layanan, penggunaan sumber daya, dan toleransi kesalahan. Batista et al. (2022) menunjukkan pentingnya pengambilan keputusan penempatan beban pada lingkungan fog dan cloud, sedangkan Li et al. (2022) menekankan penjadwalan sumber daya yang mempertimbangkan Quality of Service. Tasneem dan Jabbar (2022) juga menempatkan skalabilitas dan fault tolerance sebagai indikator utama

DOI: https://doi.org/10.31004/riggs.v5i2.11765

evaluasi load balancing. Implikasinya bagi SIMAK adalah pengukuran keberhasilan tidak cukup menggunakan rata-rata response time. Institusi perlu menilai persentil ke-95 atau ke-99, tingkat keberhasilan transaksi, jumlah sesi yang terputus, waktu failover, replication lag, RPO, RTO, dan ketersediaan bulanan. Indikator tersebut memberikan gambaran yang lebih akurat mengenai pengalaman pengguna pada kondisi puncak dan kegagalan.

Berdasarkan sintesis tersebut, load balancing dan backup server merupakan komponen yang saling melengkapi, tetapi keduanya harus ditempatkan dalam kerangka tata kelola layanan. Institusi memerlukan penanggung jawab sistem, prosedur perubahan, jadwal pemeliharaan, daftar kontak eskalasi, runbook insiden, serta mekanisme evaluasi pascakejadian. Setiap perubahan pada aplikasi, basis data, atau konfigurasi load balancer harus diuji dan dapat dikembalikan apabila menimbulkan masalah. Dokumentasi arsitektur perlu diperbarui agar tidak hanya dipahami oleh satu administrator. Pelatihan dan simulasi insiden membantu tim melakukan failover serta pemulihan dengan cepat. Dengan tata kelola tersebut, investasi pada server cadangan tidak menjadi perangkat pasif yang tidak pernah diuji, melainkan bagian aktif dari strategi kontinuitas layanan akademik.

Kajian ini juga menunjukkan bahwa tidak terdapat satu konfigurasi yang selalu paling baik untuk seluruh perguruan tinggi. Jumlah pengguna, karakter aplikasi, kemampuan jaringan, anggaran, dan kompetensi tim menentukan pilihan teknologi. Institusi dengan skala menengah dapat memulai dari dua backend, satu session store, replikasi database, dan load balancer redundan. Institusi dengan trafik sangat besar dapat mengembangkan cluster Kubernetes, autoscaling, distributed storage, dan observabilitas yang lebih lengkap. Namun, prinsip dasarnya tetap sama, yaitu menghilangkan single point of failure, mendistribusikan beban, menjaga konsistensi data, menyediakan salinan pemulihan, memonitor seluruh komponen, dan menguji prosedur kegagalan. Penerapan yang bertahap dan berbasis metrik akan lebih efektif dibandingkan modernisasi sekaligus tanpa pemahaman terhadap pola beban SIMAK.

Keterbatasan kajian terletak pada penggunaan metode studi literatur sehingga rekomendasi belum diuji secara langsung pada infrastruktur Universitas Methodist Indonesia. Perbedaan versi aplikasi, struktur database, kapasitas jaringan, dan pola akses dapat menghasilkan kinerja yang berbeda dari penelitian terdahulu. Oleh karena itu, tahap lanjutan perlu melakukan eksperimen menggunakan replika lingkungan SIMAK, menguji beberapa algoritma HAProxy, membandingkan arsitektur single server dan multi-server, serta mengukur response time, throughput, error rate, failover time, dan konsistensi data. Penelitian lanjutan juga dapat mengevaluasi biaya implementasi, kebutuhan sumber daya manusia, dan efektivitas autoscaling terjadwal dibandingkan autoscaling reaktif. Hasil eksperimen tersebut akan memperkuat rekomendasi arsitektur dan menghasilkan konfigurasi yang lebih spesifik bagi kebutuhan institusi.

5. Kesimpulan

Berdasarkan hasil analisis dan sintesis komprehensif terhadap berbagai literatur ilmiah yang dikaji, dapat disimplukan bahwa penerapan teknologi load balancing dan penyediaan backup server merupakan strategi infrastruktur yang sangat efektif untuk meminimalkan risiko downtime serta meningkatkan ketersediaan layanan (high availability) pada Sistem Informasi Akademik (SIMAK). Penggunaan arsitektur multi-server terbukti mampu mengeliminasi kelemahan titik rawan kegagalan (Single Point of Failure) yang selama ini menjadi kendala utama pada arsitektur server tunggal (single server). Implementasi load balancing menggunakan HAProxy berhasil mendistribusikan trafik pengguna secara merata pada layer aplikasi sehingga mencegah terjadinya overload saat periode puncak aktivitas akademik seperti pengisian KRS dan akses KHS. Sementara itu, ketersediaan backup server yang dikonfigurasi dalam sistem basis data terklaster (database cluster) mampu menjamin kontinuitas operasional melalui mekanisme failover otomatis yang memindahkan peran peladen utama yang terganggu ke peladen cadangan secara real-time. Keamanan data akademik tersebut diperkuat oleh manajemen pencadangan terstruktur lewat kombinasi metode full backup dan incremental backup untuk mengantisipasi skenario pemulihan bencana secara cepat. Rangkaian solusi teknologi yang disintesis dari berbagai penelitian terdahulu ini menghasilkan sebuah cetak biru arsitektur tiga lapis (Three-Tier Architecture) yang sangat relevan untuk direkomendasikan pada Universitas Methodist Indonesia. Melalui integrasi berlapis yang mencakup penyeimbang beban trafik di sisi gerbang masuk, pembagian virtual web server di sisi aplikasi, hingga replikasi master-slave dan otomatisasi pencadangan terjadwal di sisi basis data, Universitas Methodist Indonesia dapat membangun infrastruktur SIMAK yang memiliki ketahanan sistem yang tinggi (service resilience). Implementasi strategi ini diharapkan dapat menjadi acuan teknis yang andal bagi institusi dalam memodernisasi perangkat operasionalnya, sehingga pelayanan administrasi dan akademik kepada dosen maupun mahasiswa dapat terselenggara secara cepat, stabil, aman, dan berkelanjutan.

DOI: https://doi.org/10.31004/riggs.v5i2.11765

Referensi

1. Misbullah, A., Nazaruddin, Rasudin, & Zulfan. (2020). Analisa Proses Migrasi Mysql Non-Cluster Ke Cluster Dalam Menangani Fail-Over Sistem Akademik Universitas Syiah Kuala. *Cyberspace: Jurnal Pendidikan Teknologi Informasi*, *4*(1), 40. https://doi.org/10.22373/cj.v4i1.7246
2. Pradipta, K. S., Arna, G., Saskara, J., Gede, B., & Yudistira, K. (2026). *Implementasi Kubernetes Cluster untuk High Availability dan Load Balancing SIAK Undiksha*. *4*(4), 12831–12840.
3. Rahmatulloh, A., & MSN, F. (2017). Implementasi Load Balancing Web Server menggunakan Haproxy dan Sinkronisasi File pada Sistem Informasi Akademik Universitas Siliwangi. *Jurnal Nasional Teknologi Dan Sistem Informasi*, *3*(2), 241–248. https://doi.org/10.25077/teknosi.v3i2.2017.241-248
4. Rosano, A., & Sudaradjat, D. (2020). *Manajemen Backup Data untuk Penyelamatan Data Nasabah pada Sistem Informasi Perbankan (Studi Kasus : PT Bank XYZ)*. *4*(2), 210–217.
5. Syaputra, A. W., & Assegaff, S. (2017). *ANALISIS DAN IMPLEMENTASI LOAD BALANCING DENGAN METODE NTH PADA JARINGAN DINAS PENDIDIKAN PROVINSI JAMBI*. *2*(4).
6. Wirawan, A., Gatra, R., Hidayat, H., & Prasetyawan, D. (2024). *Implementasi Load Balancing dengan HAProxy di Sistem Informasi Akademik UIN Sunan Kalijaga*. *9*(1), 39–49.
7. Pentanugraha, E., Saragih, A. S., & Christian, E. (2024). Analisis Kinerja Load Balancing Webserver Menggunakan HAProxy Terintegrasi dengan Grafana sebagai Monitoring dan Notifikasi Telegram. Journal of Information Technology and Computer Science, 4(1), 68-80. https://doi.org/10.47111/jointecoms.v4i1.13191
8. Suhatman, R., Rida, M. A. F., Surya, I., & Tasya, R. (2024). Penerapan Disaster Recovery Plan pada Database MySQL Menggunakan Metode Master to Master Replication Berbasis Docker. Jurnal Komputer Terapan, 10(1), 78-85. https://doi.org/10.35143/jkt.v10i1.6226
9. Amalia, Z., & Riskiono, S. D. (2025). Optimization of Web Server Load Balancing Using HAProxy with the Weighted Round Robin Algorithm. Jurnal Teknologi Informasi dan Pendidikan, 18(1), 823-834. https://doi.org/10.24036/jtip.v18i1.965
10. Muhammad Algazali, Surimi, L., Aksara, L. O. M. B., Nangi, J., Saputra, R. A., & Wibowo, A. H. (2025). Analisis Performansi Load Balancing Weighted Least Connection dengan HAProxy. Jurnal Informatika Polinema, 11(3). https://doi.org/10.33795/jip.v11i3.7198
11. Rifat, T. M. F., & Oktiawati, U. Y. (2025). Analisis Kinerja Algoritma Weighted Round Robin dan Weighted Least Connection pada Sistem Load Balancing Web Server dengan HAProxy Terintegrasi Cacti Monitoring. Journal of Internet and Software Engineering, 6(1). https://doi.org/10.22146/jise.v6i1.15494
12. Laksono, P. A., & Riskiono, S. D. (2025). Analisis Perbandingan Algoritma Round Robin dan Least Connection dalam HAProxy untuk Load Balancer Server Web. Dinamik, 30(2). https://doi.org/10.35315/dinamik.v30i2.10154
13. Nguyen, Q.-M., Phan, L.-A., & Kim, T. (2022). Load-Balancing of Kubernetes-Based Edge Computing Infrastructure Using Resource Adaptive Proxy. Sensors, 22(8), 2869. https://doi.org/10.3390/s22082869
14. Mondal, S. K., Wu, X., Kabir, H. M. D., Dai, H.-N., Ni, K., Yuan, H., & Wang, T. (2023). Toward Optimal Load Prediction and Customizable Autoscaling Scheme for Kubernetes. Mathematics, 11(12), 2675. https://doi.org/10.3390/math11122675
15. Augustyn, D. R., Wycislik, L., & Sojka, M. (2024). Tuning a Kubernetes Horizontal Pod Autoscaler for Meeting Performance and Load Demands in Cloud Deployments. Applied Sciences, 14(2), 646. https://doi.org/10.3390/app14020646
16. Yuan, H., & Liao, S. (2024). A Time Series-Based Approach to Elastic Kubernetes Scaling. Electronics, 13(2), 285. https://doi.org/10.3390/electronics13020285
17. Ahmad, A. A.-S., & Andras, P. (2022). Scalability Resilience Framework Using Application-Level Fault Injection for Cloud-Based Software Services. Journal of Cloud Computing, 11, 1. https://doi.org/10.1186/s13677-021-00277-z
18. Tengku Asmawi, T. N., Ismail, A., & Shen, J. (2022). Cloud Failure Prediction Based on Traditional Machine Learning and Deep Learning. Journal of Cloud Computing, 11, 47. https://doi.org/10.1186/s13677-022-00327-0
19. Alawadi, A. H., & Molnar, S. (2022). Oddlab: Fault-Tolerant Aware Load-Balancing Framework for Data Center Networks. Annals of Telecommunications, 77, 641-662. https://doi.org/10.1007/s12243-021-00898-0
20. Batista, E., Figueiredo, G., & Prazeres, C. (2022). Load Balancing between Fog and Cloud in Fog of Things Based Platforms through Software-Defined Networking. Journal of King Saud University - Computer and Information Sciences, 34, 7111-7125. https://doi.org/10.1016/j.jksuci.2021.10.003
21. Tasneem, R., & Jabbar, M. A. (2022). An Insight into Load Balancing in Cloud Computing. In Proceeding of the 2021 International Conference on Wireless Communications, Networking and Applications (pp. 1125-1140). Springer. https://doi.org/10.1007/978-981-19-2456-9_113
22. Li, J., Zhang, R., & Zheng, Y. (2022). QoS-Aware and Multi-Objective Virtual Machine Dynamic Scheduling for Big Data Centers in Clouds. Soft Computing, 26, 10239-10252. https://doi.org/10.1007/s00500-022-07327-x
23. Pradhan, A., & Bisoy, S. K. (2022). A Novel Load Balancing Technique for Cloud Computing Platform Based on PSO. Journal of King Saud University - Computer and Information Sciences, 34, 3988-3995. https://doi.org/10.1016/j.jksuci.2020.10.016

DOI: https://doi.org/10.31004/riggs.v5i2.11765