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: 18432-18441
P-ISSN: 2963-9298, e-ISSN: 2963-914X

Optimasi Ketersediaan Web Server dengan Round Robin, Haproxy, dan Keepalived

Setiani Hulu, Michele Abelisa Manalu, Lotar Mateus Sinaga
Teknik Informatika, Fakultas Ilmu Komputer, Universitas Katolik Santo Thomas
setianyhulu@gmail.com, michelemanalu@gmail.com, lotarmateus88@gmail.com

Abstrak

Pesatnya perkembangan layanan digital menuntut ketersediaan infrastruktur web server yang andal dan responsif. Penggunaan server tunggal berpotensi menimbulkan masalah serius berupa beban kerja berlebih dan risiko Single Point of Failure. Untuk mengatasi tantangan tersebut, penelitian ini mengimplementasikan arsitektur load balancing menggunakan HAProxy dengan algoritma Round Robin serta mekanisme high availability berbasis Keepalived untuk meningkatkan ketersediaan layanan secara berkelanjutan. Metode eksperimental diterapkan melalui implementasi sistem secara langsung pada lingkungan virtual berbasis Debian, menggunakan topologi yang terdiri atas satu server master, satu server backup, dan satu klien penguji. Dalam konfigurasi ini, HAProxy berfungsi mendistribusikan trafik HTTP secara bergantian ke dua server backend untuk mencegah saturasi sumber daya, sedangkan Keepalived mengelola Virtual IP melalui protokol VRRP untuk mendukung failover secara otomatis. Hasil pengujian menunjukkan sistem mampu membagi permintaan simulasi 6.000 sample request menggunakan Apache Jmeter secara merata dengan rasio 50:50, sehingga mampu menekan error rate secara signifikan menjadi 2,35%. Saat server master mengalami gangguan buatan, mekanisme failover bekerja cepat dalam rentang waktu satu detik, diikuti proses failback yang lancar tanpa memutus sesi koneksi klien aktif. Berdasarkan pedoman standar TIPHON, kualitas jaringan pascapengujian dikategorikan "Sangat Bagus" dengan nilai packet loss 2% dan rata-rata latensi 47 ms. Kombinasi HAProxy dan Keepalived terbukti efektif menjaga kontinuitas layanan dan meningkatkan keandalan sistem.

Kata kunci: *Failover, HAProxy, High Availability*

1. Latar Belakang

Perkembangan teknologi informasi dan transformasi digital telah meningkatkan kebutuhan akan layanan web yang stabil, cepat, dan selalu tersedia di berbagai sektor, termasuk pendidikan, pemerintahan, industri, dan bisnis. Infrastruktur server web merupakan komponen penting untuk mendukung beragam layanan digital dan oleh karena itu harus mampu menyediakan layanan yang cepat, stabil, dan sangat tersedia bagi pengguna. Namun, pertumbuhan jumlah pengguna seringkali tidak diimbangi dengan kapasitas server yang memadai, sehingga mengakibatkan penurunan kinerja layanan, waktu respons yang lebih lama, dan bahkan kegagalan sistem karena beban kerja yang tidak seimbang [1]. Seiring bertambahnya jumlah pengguna dan peningkatan permintaan akses secara simultan, implementasi *web server* tunggal mulai menunjukkan keterbatasannya dalam mengelola volume pekerjaan yang berat dan berisiko menimbulkan kegagalan tunggal (*single point of failure*), sebuah situasi di mana seluruh layanan menjadi tidak terjangkau akibat gangguan pada server pusat [2]. Kondisi tersebut dapat menyebabkan penurunan kualitas layanan, meningkatnya waktu henti (*downtime*), serta gangguan terhadap aktivitas pengguna yang bergantung pada layanan web.

Untuk mengatasi masalah tersebut, *Load Balancing* diterapkan sebagai teknik pembagian beban lalu lintas jaringan guna menghindari kemacetan, menjaga kinerja, dan menyediakan jalur cadangan [3]. HAProxy dipilih karena efektivitasnya sebagai perangkat lunak *open-source* yang ringan dan andal dalam mendistribusikan trafik untuk mencegah penumpukan beban kerja pada satu titik [4]. Penelitian ini menggunakan algoritma Round Robin pada HAProxy untuk membagi setiap permintaan secara bergantian ke seluruh *server backend*, sehingga pemanfaatan sumber daya menjadi lebih seimbang dan risiko kelebihan beban dapat ditekan [5] [6]. Namun, penerapan load balancing saja tidak cukup jika server pembagi beban itu sendiri mengalami gangguan. Oleh karena itu, arsitektur ini diintegrasikan dengan Keepalived untuk menghasilkan High Availability [7]. Keepalived memanfaatkan *Virtual Router Redundancy Protocol* (VRRP) guna menyediakan *Virtual IP* (VIP) dan redundansi jaringan, sekaligus memantau kondisi server agar layanan tetap beroperasi tanpa interupsi [8][9].

Beberapa penelitian sebelumnya telah mengkaji efektivitas implementasi HAProxy. Waluyo dkk.(2024) membuktikan bahwa HAProxy dengan algoritma Round Robin mampu membagi *throughput* secara stabil pada pengujian hingga 3.000 pengguna secara bersamaan [7]. Dalam kajian komparatif, Laksono dan Riskiono menemukan bahwa algoritma *Least Connection* memberikan *response time* 6%-8% lebih cepat pada beban tidak merata, sementara Round Robin menghasilkan *throughput* 1%-3% lebih tinggi pada kondisi beban seimbang [10]. Selain itu, Hanafiah dan Wandri mengonfirmasi bahwa penerapan arsitektur *cluster load balance* secara signifikan meningkatkan performa dan pemerataan beban dibandingkan dengan penggunaan server tunggal [11].

Sementara itu, Syahputra dkk. membuktikan bahwa protokol VRRP memiliki performa redundansi yang baik dengan *delay* 0,16 ms, *packet loss* 1,01%, dan *downtime* 12,6 detik, sehingga dinilai memenuhi standar kualitas ITU-T untuk sistem *High Availlability* [12]. Selanjutnya, Hwishan dan Astour menunjukkan bahwa mekanisme *failover* HAProxy mampu mengisolasi server yang bermasalah dalam waktu 9 detik dan mengembalikannya ke sistem dalam 6 detik demi menjaga kontinuitas layanan [13].

Meskipun demikian, penelitian-penelitian terdahulu mayoritas masih berfokus pada implementasi load balancing dan high availability secara terpisah, serta diuji pada lingkungan sistem yang berbeda. Menjawab celah tersebut, penelitian ini mengintegrasikan perangkat lunak HAProxy menggunakan algoritma Round Robin dan Keepalived secara bersamaan pada arsitektur web server berbasis Debian. Sistem ini diimplementasikan menggunakan dua node server yang bertindak sebagai Master dan Backup, serta perangkat klien sebagai lingkungan pengujian. Selain mengevaluasi fungsionalitas distribusi permintaan dan mekanisme failover Virtual IP (VIP) secara otomatis, penelitian juga menguji batas ketahanan sistem (stress testing) di bawah gempuran lalu lintas masif sebanyak 6.000 samples request menggunakan Apache JMeter. Hasil penelitian ini diharapkan dapat memberikan alternatif konfigurasi manajemen web server yang sederhana, efisien, serta mampu menjaga ketersediaan layanan yang baik.

2. Metode Penelitian

Penelitian ini menggunakan metode eksperimental (Experimental Research) dengan pendekatan implementasi sistem untuk membangun dan mengevaluasi penyeimbangan beban menggunakan HAProxy dengan algoritma Round Robin pada server web berbasis Debian. Metode eksperimental dipilih karena memungkinkan proses implementasi, konfigurasi, dan pengujian secara langsung, sehingga memudahkan analisis kinerja sistem berdasarkan distribusi beban dan mekanisme failover yang diaktifkan ketika salah satu server mengalami kegagalan.

Tahapan penelitian meliputi identifikasi kebutuhan sistem, perancangan arsitektur jaringan, instalasi dan konfigurasi HAProxy, Keepalived, dan Apache2 pada masing-masing server, serta implementasi Virtual IP (VIP). Tahapan selanjutnya berfokus pada evaluasi sistem yang mencakup pengujian fungsional distribusi beban menggunakan metode Round Robin, pengujian keandalan mekanisme failover menggunakan protokol VRRP, dan pengujian performa titik jenuh (stress testing) menggunakan Apache JMeter. Rangkaian penelitian ini diakhiri dengan analisis hasil pengujian untuk mengevaluasi secara komprehensif performa sistem yang telah dibangun. Secara spesifik, pengujian sistem pada penelitian ini dibagi menjadi tiga skenario utama.

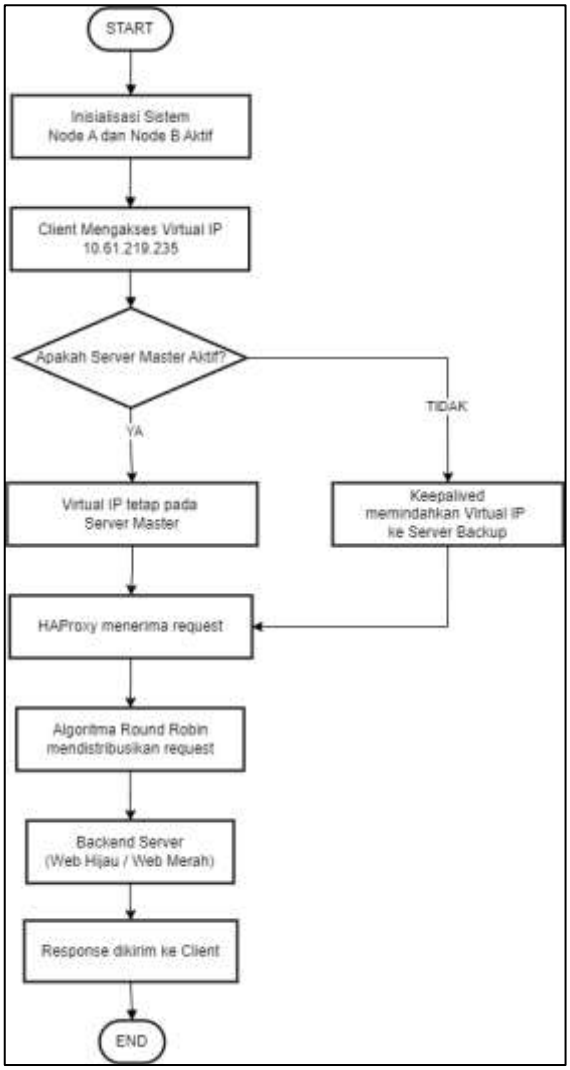
Pengujian fungsional distribusi beban (load balancing) dilakukan dengan menjalankan Node A (Server Master) dan Node B (Server Backup) serta memastikan seluruh layanan beroperasi. Selanjutnya, perangkat klien Windows dihubungkan ke jaringan yang sama untuk mengakses Virtual IP (VIP). Proses penyegaran (refresh) halaman web dilakukan secara berulang pada klien untuk mengamati perubahan antarmuka pada backend server. Apabila tampilan berubah secara bergantian antara latar belakang hijau pada Backend Server 1 dan merah pada Backend Server 2, mekanisme load balancing dinyatakan berjalan dengan baik.

Pengujian fungsional high availability dilakukan dengan memastikan kedua node aktif dan VIP berada pada Node A. Klien menjalankan perintah continuous ping menuju VIP untuk memantau konektivitas secara real-time. Kegagalan sistem kemudian disimulasikan (fault injection) dengan menghentikan layanan Keepalived pada Node A secara paksa menggunakan perintah systemctl stop keepalived. Pemantauan dilakukan terhadap proses failover atau perpindahan VIP secara otomatis ke Node B dengan memastikan klien tetap terhubung melalui jumlah paket Request Timed Out yang minimal. Setelah layanan Node A diaktifkan kembali, mekanisme failback diamati untuk memastikan VIP kembali ke Node A sesuai prioritas konfigurasi.

DOI: https://doi.org/10.31004/riggs.v5i2.11414

Pengujian performa dan analisis titik jenuh (stress testing) dilakukan dengan mempersiapkan instrumen Apache JMeter berparameter 3.000 concurrent users dan target 6.000 samples request. Simulasi lalu lintas jaringan masif diarahkan ke VIP untuk membandingkan metrik performa berupa error rate dan latensi antara arsitektur server tunggal dan sistem cluster (HA). Selama pengujian, utilitas sumber daya perangkat keras dipantau menggunakan perangkat lunak htop pada backend server, diiringi analisis stabilitas distribusi sesi melalui dasbor statistik HAProxy pada kondisi beban puncak.

Alur kerja implementasi *load balancing* dan *high availability* pada *web server* divisualisasikan menggunakan *flowchart* [14] untuk menggambarkan tahapan operasional sistem. Alur dimulai ketika klien mengakses layanan melalui VIP, kemudian permintaan diterima oleh HAProxy dan didistribusikan ke *backend server* menggunakan algoritma *Round Robin*. Selanjutnya, Keepalived akan melakukan mekanisme *failover* secara otomatis apabila *Server Master* mengalami gangguan, sehingga layanan tetap dapat diakses oleh klien. Alur kerja sistem yang diimplementasikan dapat dilihat pada Gambar 1.



Gambar 1. Flowchart Sistem

Untuk memperoleh gambaran yang lebih objektif mengenai kinerja sistem, data hasil pemantauan selama proses failover dan failback dianalisis menggunakan parameter pengukuran *Quality of Service* (QoS). Berdasarkan pedoman evaluasi kuantitatif dari standar TIPHON [15], analisis keandalan jaringan dalam penelitian ini difokuskan secara spesifik pada pengukuran metrik *packet loss* [16]. dan *delay* [17]. Adapun kategori degradasi kualitas jaringan untuk kedua metrik tersebut dapat dirujuk pada Tabel 1 dan Tabel 2 di bawah ini.

DOI: https://doi.org/10.31004/riggs.v5i2.11414

Tabel 1. Standar Indeks Packet Loss

Indeks	Persentase (%)	Kategori Degradasi
4	0-2%	Sangat baik
3	3-14%	Baik
2	15-24%	Sedang
1	>25%	Buruk

Tabel 2. Standar Indeks Delay

Indeks	Besaran *Delay*	Kategori Degradasi
4	<150 ms	Sangat baik
3	150 ms – 300 ms	Baik
2	300 ms – 450 ms	Sedang
1	>450 ms	Buruk

Untuk menguji batas kemampuan infrastruktur *cluster server* secara objektif, instrumen pengujian beban dikonfigurasi menggunakan parameter pada Apache JMeter seperti yang tertera pada Tabel 3 berikut:

Tabel 3. Parameter Pengujian Beban (*Stress Testing*)

Parameter Pengujian	Nilai / Pengaturan	Keterangan
Alat Pengujian (*Tools*)	Apache JMeter	Alat simulasi beban lalu lintas (*traffic*)
Jumlah Pengguna (*Threads*)	3.000 *Concurrent Users*	Jumlah pengguna simulasi yang mengakses server secara simultan
Waktu Akses Serentak (*Ramp-up*)	1 Detik	Waktu yang dibutuhkan untuk mengirimkan trafik dalam waktu bersamaan
Jumlah Perulangan (*Loop Count*)	2 Kali	Setiap pengguna melakukan pemuatan halaman web sebanyak 2 kali
Total Sampel Beban (*Samples*)	6.000 *Samples Request*	Hasil perkalian antara jumlah *Threads* dan *Loop Count* (3.000 x 2)
Metrik Evaluasi Utama	*Error rate & Latency*	Indikator utama untuk mengukur tingkat kegagalan dan waktu respons pemrosesan

3. Hasil dan Diskusi

3.1. Implementasi Konfigurasi HAProxy dan Keepalived

Tahap awal dalam penelitian ini adalah melakukan implementasi parameter konfigurasi pada layanan HAProxy dan Keepalived yang berjalan di sistem operasi server. Konfigurasi tersebut menjadi dasar utama dalam mengatur mekanisme distribusi lalu lintas jaringan HTTP.

DOI: https://doi.org/10.31004/riggs.v5i2.11414

Selain itu, pengaturan ini juga menentukan cara server menjaga dan mengelola ketersediaan layanan (availability) agar tetap beroperasi dengan baik. Langkah ini sangat esensial untuk memastikan kesiapan sistem sebelum menghadapi lonjakan permintaan maupun simulasi kegagalan perangkat keras keras (hardware failure).

3.1.1 Konfigurasi Load Balancing pada HAProxy

Implementasi mekanisme load balancing dilakukan melalui konfigurasi berkas utama HAProxy yang berada pada direktori /etc/haproxy/haproxy.cfg. Konfigurasi tersebut terdiri atas dua bagian utama, yaitu frontend yang berfungsi menerima permintaan dari klien dan backend yang bertugas memproses serta meneruskan permintaan tersebut ke server tujuan.

Pada bagian backend, metode distribusi beban ditentukan menggunakan parameter balance roundrobin. Metode ini memungkinkan HAProxy mendistribusikan permintaan klien secara bergiliran ke setiap server yang tersedia sehingga beban kerja dapat tersebar secara lebih merata tanpa membebani satu titik secara berlebihan.

Selanjutnya, Node A dan Node B dikonfigurasi sebagai server backend dengan alamat IP masing-masing. Konfigurasi ini memungkinkan HAProxy mengarahkan lalu lintas data ke server yang sesuai dengan mekanisme load balancing yang telah ditetapkan, sebagaimana diilustrasikan pada Gambar 2.

```
frontend http_front
        bind *:80
        default_backend http_back

backend http_back
        balance roundrobin
        server node-a 10.61.219.169:8080 check
        server node-b 10.61.219.165:8080 check
```

Gambar 2. Konfigurasi parameter Round Robin

3.1.2 Konfigurasi High Availability pada Keepalived

Untuk meningkatkan ketersediaan layanan dan menghindari terjadinya Single Point of Failure (SPOF), layanan Keepalived dikonfigurasi pada kedua node dengan memanfaatkan protokol Virtual Router Redundancy Protocol (VRRP). Konfigurasi dilakukan melalui berkas /etc/keepalived/keepalived.conf, yang digunakan untuk mengatur mekanisme perpindahan layanan secara otomatis (failover) antara server utama dan server cadangan.

Pada Node A, parameter state ditetapkan sebagai MASTER dengan nilai priority 100. Prioritas yang lebih tinggi ini menjadikan Node A sebagai server utama yang bertanggung jawab untuk memiliki dan mengumumkan Virtual IP (VIP) 10.61.219.235 ke jaringan selama server berada dalam kondisi normal, sebagaimana ditunjukkan pada Gambar 3.

```
vrrp_instance VI_1 {
        state MASTER
        interface enp0s3
        virtual_router_id 51
        priority 100
        advert_int 1
        authentication {
                auth_type PASS
                auth_pass 12345
        }
        virtual_ipaddress {
                10.61.219.235/24
                }
        }
```

Gambar 3. Konfigurasi Layanan Keepalived pada Server Master (Node A)

Sementara itu, Node B dikonfigurasi dengan parameter state BACKUP dan nilai priority 90 seperti yang terlihat pada Gambar 4. Dengan prioritas yang lebih rendah, Node B berfungsi sebagai server cadangan yang terus memantau kondisi Node A. Apabila Node A mengalami gangguan atau tidak dapat diakses, Node B akan secara

DOI: https://doi.org/10.31004/riggs.v5i2.11414

otomatis mengambil alih kepemilikan VIP sehingga layanan tetap dapat diakses tanpa memerlukan intervensi manual.

```
vrrp_instance VI-1{
        state BACKUP
        interface enp0s3
        virtual_router_id 51
        priority 90
        advert_int 1
        authentication {
                auth_type PASS
                auth_pass 12345
        }
        virtual_ipaddress {
                10.61.219.235/24
        }
}
```

Gambar 4. Konfigurasi Layanan Keepalived pada Server Backup (Node B)

Implementasi konfigurasi HAProxy dan Keepalived telah berhasil dilakukan tanpa ditemukan pesan kesalahan (error). Kondisi ini menunjukkan bahwa mekanisme load balancing dan high availability telah berjalan dengan baik sebagai layanan di latar belakang (background service). Dengan demikian, sistem dinyatakan siap untuk memasuki tahap pengujian yang bertujuan memverifikasi fungsi distribusi beban dan mekanisme failover dari sisi klien.

3.2. Pengujian Fungsional

Pengujian fungsional dilakukan untuk memastikan bahwa seluruh komponen sistem bekerja secara terintegrasi sesuai dengan rancangan. Fokus utama dari pengujian ini mencakup evaluasi terhadap proses pembagian trafik jaringan dan mekanisme redundansi layanan.

Melalui pengujian fungsional ini, kinerja sistem pada kondisi operasional normal maupun saat terjadi gangguan simulasi dapat dipantau secara langsung dari perspektif klien atau pengguna akhir.

3.2.1. Uji Distribusi Beban

Pengujian ini bertujuan memastikan efektivitas algoritma *Round Robin* dalam membagi permintaan klien. Pengujian dilakukan dengan mengakses alamat VIP 10.61.219.235 melalui peramban *web* klien. Pada akses pertama, permintaan berhasil diteruskan ke Node A, yang ditunjukkan dengan halaman *web* berlatar hijau bertuliskan "SERVER MASTER" pada Gambar 5.

Gambar 5. Tampilan Antarmuka Web Saat Lalu Lintas Diarahkan ke Server Master

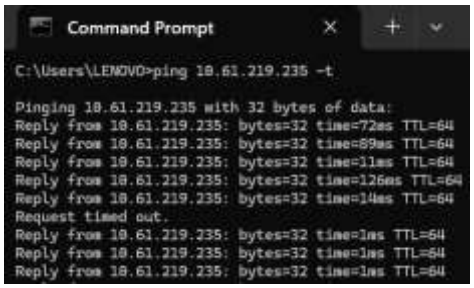
Selanjutnya, melalui proses penyegaran (*refresh*) halaman peramban secara berkala, sistem secara konsisten mengarahkan lalu lintas ke Node B yang ditandai dengan tampilan berlatar merah bertuliskan "SERVER BACKUP" pada Gambar 6. Pergantian tampilan yang dinamis tersebut menunjukkan bahwa HAProxy berhasil membagi beban trafik secara bergantian ke setiap server *backend* sesuai dengan aturan algoritma *Round Robin*.

DOI: https://doi.org/10.31004/riggs.v5i2.11414

Tampilan Antarmuka Web Saat Lalu Lintas Diarahkan ke Server Backup

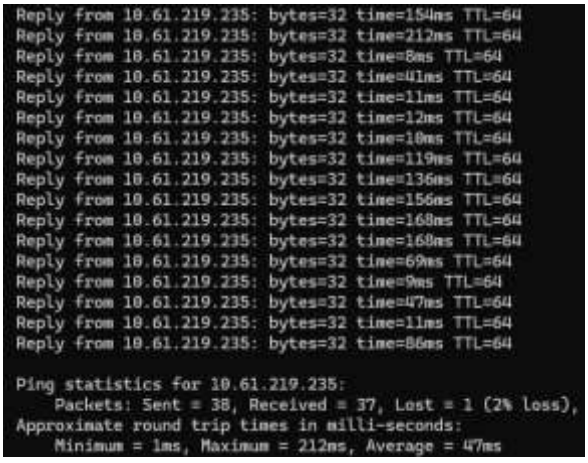
3.2.2. Uji Keandalan

Pengujian ini dilakukan untuk menilai respons sistem terhadap kondisi kegagalan menggunakan metode continuous ping ke alamat VIP. Ketika layanan Keepalived pada Node A dihentikan secara paksa, Virtual IP secara otomatis berpindah ke Node B. Hasil pemantauan pada Gambar 7 menunjukkan bahwa proses perpindahan tersebut hanya menimbulkan gangguan koneksi dalam waktu yang sangat singkat, ditandai dengan munculnya satu pesan Request Timed Out (RTO) sebagai indikasi downtime sekitar 1 detik.



Gambar 7. Log Pengujian Failover Menunjukkan RTO 1 Detik

Begitu pula pada proses pemulihan (failback), sistem mampu mengembalikan kepemilikan VIP ke Node A secara otomatis tanpa memutus sesi komunikasi pengguna. Statistik akhir jaringan pada Gambar 8 menunjukkan kualitas yang sangat baik dengan tingkat packet loss hanya sebesar 2% dan rata-rata latensi 47 ms. Mengacu pada standar indeks Quality of Service (QoS) TIPHON, kedua perolehan nilai tersebut masuk ke dalam kategori "Sangat Bagus". Hasil ini mengonfirmasi bahwa protokol VRRP berhasil menjaga kontinuitas layanan secara optimal.



Gambar 8. Statistik Jaringan Pasca-Pengujian menunjukkan Packet Loss 2%

DOI: https://doi.org/10.31004/riggs.v5i2.11414

3.3. Pengujian Performa dan Titik Jenuh

Pengujian performa dilakukan secara terukur untuk membandingkan kapasitas komputasi arsitektur server tunggal dengan sistem High Availability (HA) yang menerapkan load balancing. Evaluasi perbandingan ini penting untuk mengkuantifikasi efisiensi yang dihasilkan dari penerapan klasterisasi server.

Selain itu, skenario pengujian ini juga dirancang untuk mengetahui batas toleransi maksimal (threshold) dari infrastruktur yang dibangun ketika dipaksa menangani lonjakan lalu lintas data yang ekstrem dalam satu waktu.

3.3.1. Komparasi Performa

Pengujian komparatif ini dilakukan untuk membandingkan secara langsung metrik kinerja antara arsitektur server tunggal tanpa mekanisme distribusi beban dan sistem HA yang menerapkan algoritma Round Robin. Hasil simulasi pembebanan pada server tunggal, sebagaimana ditunjukkan pada Gambar 9, memperlihatkan bahwa sistem mengalami tingkat penolakan koneksi yang cukup tinggi.

Gambar 9. Hasil Pengujian beban pada Server Tunggal

Sebagai pembanding, Gambar 10 menampilkan hasil simulasi pembebanan yang sama pada sistem HA. Hasilnya menunjukkan peningkatan kinerja yang signifikan ditandai dengan penurunan jumlah kegagalan (*error*) yang sangat drastis saat sistem menangani 6.000 sampel permintaan.

Gambar 10. Hasil Pengujian beban pada Sistem High Availability (Error Rate 2,35%)

Untuk memberikan gambaran perbandingan yang lebih komprehensif, rekapitulasi metrik kinerja dari kedua skenario pengujian disajikan pada Tabel 4. Berdasarkan Tabel 4, implementasi *load balancing* pada sistem HA terbukti berhasil meningkatkan keandalan server. Penurunan *error rate* dari 7,03% menjadi 2,35% menunjukkan bahwa algoritma *Round Robin* dapat mendistribusikan permintaan klien secara efektif.

Tabel 4. Perbandingan Performa Sistem

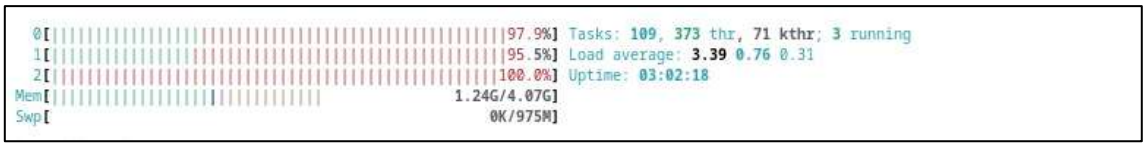
Metrik Kinerja	Server Tunggal (Tanpa HA)	Sistem HA (*Round Robin*)
Tingkat Kegagalan (*Error Rate*)	7,03%	2,35%
Rata-rata Waktu Respons (*Average Latency*)	2.161 ms	4.606 ms
Stabilitas Distribusi Beban	Rendah (Trafik menumpuk)	Tinggi (Trafik Merata)

Berbeda dengan server tunggal yang mengalami hambatan pemrosesan dan tingginya penolakan koneksi, sistem HA mampu mendistribusikan antrean ke beberapa server cadangan. Kemampuan ini secara fundamental menjelaskan alasan mengapa jumlah kegagalan pemrosesan dapat ditekan sedemikian rupa meskipun beban lalu lintas yang dikirimkan berukuran sangat masif.

3.3.2 Analisis Titik Jenuh (Stress Testing)

Selain membandingkan performa sistem secara kuantitatif, pengujian ini juga difokuskan untuk menentukan batas kapasitas perangkat keras (threshold) dari infrastruktur server. Analisis dilakukan pada kondisi beban puncak guna mengamati respons beban sumber daya server saat dihujani trafik dalam jumlah masif. Kondisi pemanfaatan sumber daya perangkat keras ketika sistem mencapai titik jenuh ditunjukkan melalui hasil pemantauan utilitas sistem pada Gambar 11.

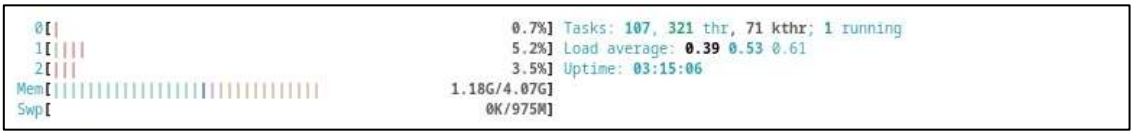
DOI: https://doi.org/10.31004/riggs.v5i2.11414



Gambar 11. Pemantauan Utilitas Sistem melalui htop Menunjukkan Saturasi CPU yang tinggi

Hasil pemantauan utilitas sistem menggunakan perintah htop pada skenario server tunggal menunjukkan bahwa penggunaan prosesor (CPU) mencapai kondisi jenuh dengan tingkat utilisasi menyentuh 100%. Beban komputasi ekstrem ini didorong oleh menumpuknya antrean proses pada web server Apache2, yang secara berantai memicu perlambatan kinerja (bottleneck) dan melesetnya tingkat error rate ke angka 7,03%.

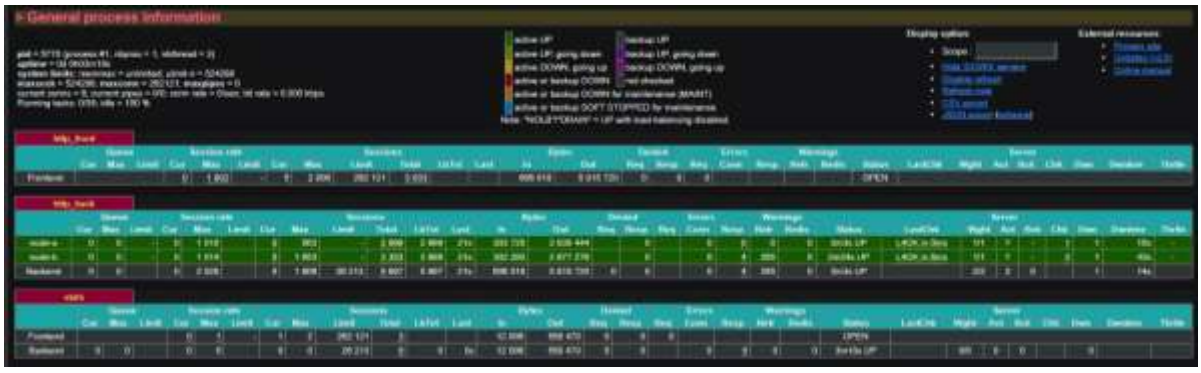
Sebagai pembanding, saat simulasi beban yang sama sebanyak 6.000 sampel diterapkan pada sistem HA, serapan sumber daya memperlihatkan tingkat efisiensi yang sangat kontras. Gambar 12 memvisualisasikan kondisi utilitas pada Node A (Server Master) ketika perangkat ini sedang sibuk mengatur pendistribusian jaringan melalui HAProxy.



Gambar 12. Pemantauan Utilitas htop pada Node A Saat Simulasi HA

Berdasarkan pemantauan pada Gambar 12, utilisasi CPU pada mesin utama bertahan pada tingkat yang sangat stabil, yakni di bawah 10%. Hal ini menjadi landasan diskursus bahwa HAProxy mampu mengeksekusi perutean sesi secara sangat ringan (efisien) tanpa membebani komputasi memori maupun prosesor pada perangkat. Server tidak mengalami saturasi karena siklus pemrosesan dialihkan secara seragam ke seluruh infrastruktur backend.

Efisiensi penggunaan sumber daya ini sangat berkorelasi positif dengan akurasi sistem dalam memecah jumlah sesi (sessions) secara simetris. Stabilitas pemerataan beban lalu lintas dapat diverifikasi melalui data statistik waktu nyata dari dasbor pemantauan HAProxy pada Gambar 13.



Gambar 13. Dasbor Statistik HAProxy Menunjukkan Distribusi Sesi yang Seimbang

Mengacu pada metrik *Load Balancing Total* (LbTot) di dasbor tersebut, algoritma Round Robin terbukti presisi dalam membagi total 6.000 permintaan klien. Secara spesifik, node-a berhasil memproses 2.999 sesi, sedangkan node-b menyelesaikan 3.001 sesi. Terciptanya rasio distribusi beban yang nyaris sempurna (50:50) ini merupakan justifikasi logis atas ketahanan server, di mana tidak ada satupun node yang diwajibkan memikul beban lebih dari batas toleransinya, sehingga kelumpuhan layanan berhasil dihindari dan error rate berhasil ditekan ke angka 2,35%.

DOI: https://doi.org/10.31004/riggs.v5i2.11414

4. Kesimpulan

Penelitian ini menunjukkan bahwa integrasi HAProxy dan Keepalived berhasil mengoptimalkan ketersediaan layanan web serta mengeliminasi Single Point of Failure (SPOF) pada infrastruktur peladen. Penerapan algoritma Round Robin terbukti efektif mencegah penumpukan beban kerja tunggal melalui pemerataan sesi secara seimbang dengan rasio distribusi mencapai 50:50, di mana Node A memproses 2.999 sesi dan Node B menyelesaikan 2.998 sesi. Mekanisme redundansi berbasis protokol VRRP juga terbukti sangat andal dalam mendeteksi gangguan, dengan waktu jeda perpindahan layanan (failover) otomatis yang sangat minim yaitu hanya sebesar 1 detik. Evaluasi kuantitatif menggunakan standar QoS TIPHON menempatkan performa jaringan klaster ini pada kategori tingkat tertinggi atau "Sangat Bagus", yang ditunjukkan oleh tingkat packet loss sebesar 2% dan rata-rata latensi sebesar 47 ms. Lebih lanjut, pada pengujian beban ekstrem (stress testing) dengan total 6.000 samples request, arsitektur high availability (HA) terbukti tangguh mempertahankan kontinuitas layanan dan mampu menekan tingkat kegagalan (error rate) secara signifikan menjadi 2,35%, berbanding terbalik dengan arsitektur server tunggal yang menyentuh angka kegagalan hingga 7,03% akibat saturasi CPU. Implikasi dari penelitian ini memberikan kontribusi nyata berupa cetak biru arsitektur manajemen server sumber terbuka (open-source) yang terstruktur, efisien, dan siap diimplementasikan pada infrastruktur penyedia layanan digital skala riil. Meskipun demikian, penelitian ini memiliki keterbatasan karena eksperimen validasi baru dilakukan pada lingkungan virtual berskala laboratorium dengan spesifikasi perangkat yang homogen. Oleh karena itu, pengembangan pada penelitian selanjutnya disarankan untuk mengevaluasi performa sistem pada topologi jaringan yang lebih heterogen, meningkatkan volume beban lalu lintas di atas 3.000 concurrent users secara simultan melalui metode multiklien terdistribusi, serta mengintegrasikan lapisan keamanan tambahan seperti Web Application Firewall (WAF) untuk memproteksi klaster server dari potensi gangguan siber.

Referensi

[1] Z. Amalia and S. D. Riskiono, “Optimization of Web Server Load Balancing Using HAProxy with the Weighted Round Robin Algorithm,” vol. 18, no. 1, pp. 823–834, 2025.
[2] D. P. Yuwono, A. I. Agung, I. G. Putu, and A. Buditjahjanto, “Peningkatan Skalabilitas Server Menggunakan HAProxy dengan Algoritma Load Balancing Round-Robin,” vol. 9, no. 1, pp. 65–75, 2025.
[3] J. Ramadani, “Implementasi Sistem Load Balancing dengan Metode Equal Cost Multi Path (ECMP) Interkoneksi Jaringan pada Kantor Dinas Koperasi Palopo,” *Peripher. J. Ilmu Komput.*, vol. 1, no. 2, 2025, [Online]. Available: https://e-journal.my.id/Peripheral/article/view/6584%0Ahttps://e-journal.my.id/Peripheral/article/download/6584/4089
[4] A. Arifin, G. P. Wardana, S. Arifin, M. F. Ridho, and H. K. Prabu, “Penerapan Cloud Load Balancing Dengan menggunakan HAProxy Dalam Meningkatkan Server Availability Pada Studi Kasus Learning Management System (LMS) Universitas XYZ,” pp. 1009–1021, 2022.
[5] C. Rawls and M. A. Salehi, “Load Balancer Tuning : Comparative Analysis of HAProxy Load Balancing Methods,” 2022.
[6] D. Siregar, A. Ariangga, Sarudin, H. Harahap, and R. Liza, “Load Balancing untuk Lalu Lintas Tinggi pada Lingkungan Cloud,” vol. 9, no. 2, pp. 38–45, 2024.
[7] M. A. Waluyo, F. Antony, and C. Setiawan, “IMPLEMENTASI LOAD BALANCING WEB,” pp. 1–7, 2024.
[8] Usanto, L. Nurlaela, and Purwono, “PENERAPAN METODE VIRTUAL ROUTER REDUNDANCY PROTOCOL (VRRP) PADA YAYASAN MASJID AL IKHLAS,” *J. ELEKTRO Inform. SWADHARMA*, vol. 02, 2022.
[9] A. Cassen, “Keepalived User Guide Release 1.4.3,” 2021. [Online]. Available: https://keepalived.readthedocs.io/
[10] P. A. Laksono and S. D. Riskiono, “ANALISIS PERBANDINGAN ALGORITMA ROUND ROBIN DAN LEAST,” vol. xx, pp. 351–360, 2025.
[11] A. Hanafiah and R. Wandri, “Implementasi Load Balancing Dengan Algoritma Penjadwalan Weighted Round Robin Dalam Mengatasi Beban Webserver,” vol. 5, no. 2, pp. 226–234, 2021.
[12] R. Syahputra *et al.*, “Analisis dan Implementasi Perbandingan Protokol VRRP dan HSRP pada Jaringan Topologi Star,” vol. 3, no. 1, 2024.
[13] S. Hwishan and S. Astour, “Enhancing Email System High Availability and Load Balancing via HAProxy : A Comparative Study on Microsoft Exchange,” vol. 110482, pp. 1–5, 2025.
[14] K. I. Listyoningrum, D. Y. Fenida, and N. Hamidi, “Inovasi Berkelanjutan dalam Bisnis: Manfaatkan Flowchart untuk Mengoptimalkan Nilai Limbah Perusahaan,” *J. Inf. Pengabdi. Masy.*, vol. 1, no. 4, pp. 100–112, 2023, doi: 10.47861/jipm-nalanda.v1i4.552.
[15] I. P. A. Wiadnyana, G. S. Santyadiputra, B. G. K. Yudistira, P. Z. E. S. Nugraha, and J. Shiddiq, “IMPLEMENTASI JARINGAN RELAY MULTI-HOP BERBASIS ESP-NOW UNTUK AKUISISI METRIK QUALITY OF SERVICE (QoS) PADA EKOSISTEM SMART GARDEN,” *JITET (Jurnal Inform. dan Tek. Elektro Ter.*, vol. 14, no. 2, 2026, doi: https://doi.org/10.23960/jitet.v14i2.9500.
[16] A. S. Muhammad Ryan Kamil, Fahmi Arzalega, Rosalinda, “Analisis Kualitas Layanan Jaringan Internet Wifi PT . XYZ dengan Metode QoS (Quality of Service),” vol. 5, pp. 77–88, 2023.
[17] R. Nuryani, M. Rahmatuloh, and W. Resdiana, “APLIKASI DASHBOARD KINERJA WIRELESS LOCAL AREA NETWORK (WLAN) MENGGUNAKAN METODE QUALITY OF SERVICE (QOS) WIRELESS LOCAL AREA NETWORK (WLAN) PERFORMANCE DASHBOARD APPLICATION USING QUALITY OF SERVICE (QOS) METHOD),” vol. 3, no. 3, pp. 146–155, 2024.

DOI: https://doi.org/10.31004/riggs.v5i2.11414