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: 21484-21490
P-ISSN: 2963-9298, e-ISSN: 2963-914X

Optimasi DNS Server Berbasis BIND9 pada Debian dengan Algoritma Least Frequently Used (LFU) Cache untuk Peningkatan Response Time

Ester Manalu, Virzinia A. Napitupulu, Lotar Mateus Sinaga
Teknik Informatika, Fakultas Ilmu Komputer, Universitas Katolik Santo Thomas Medan
estermanalu51@gmail.com, virzinianapitupulu5@gmail.com, lotarmateus88@gmail.com

Abstrak

Domain Name System (DNS) merupakan komponen penting dalam infrastruktur jaringan karena berfungsi menerjemahkan nama domain menjadi alamat Internet Protocol (IP). Kinerja DNS server berpengaruh langsung terhadap kecepatan akses layanan, sehingga pengelolaan cache perlu dioptimalkan untuk mengurangi keterlambatan proses resolusi. Penelitian ini bertujuan mengimplementasikan dan mengevaluasi algoritma Least Frequently Used (LFU) Cache pada DNS server berbasis BIND9 yang berjalan pada sistem operasi Debian 12. Penelitian menggunakan pendekatan kuantitatif eksperimental dengan metode performance benchmarking melalui perbandingan kondisi BIND9 standar dan kondisi setelah penerapan LFU Cache. Pengujian dilakukan pada tiga skenario beban, yaitu 20, 50, dan 100 klien simultan, dengan lima kali pengulangan pada setiap skenario. Response time diukur dalam milidetik menggunakan dnsperf, sedangkan mekanisme LFU mempertahankan record domain yang paling sering diakses dan mengeluarkan record dengan frekuensi penggunaan terendah ketika kapasitas cache terbatas. Hasil pengujian menunjukkan bahwa penerapan LFU Cache menurunkan rata-rata response time dari 51,8 ms menjadi 39,0 ms pada 20 klien, dari 42,4 ms menjadi 29,0 ms pada 50 klien, serta dari 127,0 ms menjadi 60,0 ms pada 100 klien. Penurunan tersebut setara dengan peningkatan efisiensi masing-masing sebesar 24,7%, 31,6%, dan 52,8%. Dengan demikian, LFU Cache efektif meningkatkan performa BIND9, terutama pada kondisi beban tinggi, tanpa memerlukan peningkatan perangkat keras server.

Kata kunci: DNS Server, BIND9, LFU Cache, Response Time, Optimasi Jaringan

1. Latar Belakang

Domain Name System (DNS) merupakan sistem penamaan terdistribusi yang memiliki peran penting dalam infrastruktur jaringan komputer dan internet. DNS berfungsi sebagai mekanisme penerjemahan nama domain yang mudah diingat oleh manusia menjadi alamat Internet Protocol (IP) yang dapat dipahami oleh perangkat jaringan. Proses tersebut memungkinkan pengguna mengakses berbagai layanan internet tanpa harus menghafalkan alamat IP dari setiap server tujuan. Hampir seluruh aktivitas komunikasi berbasis internet, seperti mengakses website, menggunakan layanan cloud, melakukan transaksi digital, hingga komunikasi berbasis aplikasi, diawali dengan proses resolusi DNS [1]. Oleh karena itu, kualitas layanan DNS menjadi salah satu faktor yang memengaruhi pengalaman pengguna dalam mengakses suatu layanan jaringan.

Dalam sistem jaringan modern, peningkatan jumlah pengguna dan layanan berbasis internet menyebabkan jumlah permintaan query DNS mengalami pertumbuhan yang signifikan. Setiap permintaan akses terhadap sebuah domain akan menghasilkan proses pencarian alamat IP melalui DNS resolver. Apabila proses resolusi DNS membutuhkan waktu yang lama, maka waktu tunggu pengguna terhadap suatu layanan akan meningkat dan menyebabkan bertambahnya latency jaringan secara keseluruhan [2]. Kondisi ini menjadi permasalahan terutama pada lingkungan jaringan dengan jumlah pengguna yang besar, seperti jaringan perusahaan, institusi pendidikan, maupun penyedia layanan internet yang menangani ribuan hingga jutaan permintaan DNS setiap harinya [3].

Salah satu implementasi DNS server yang banyak digunakan pada sistem operasi berbasis Linux adalah Berkeley Internet Name Domain versi 9 (BIND9). BIND9 merupakan perangkat lunak DNS server bersifat open-source yang dikembangkan untuk menyediakan layanan resolusi nama, pengelolaan zona domain, serta mekanisme caching DNS [4]. Pada sistem operasi Debian, BIND9 sering digunakan karena memiliki stabilitas, keamanan, dan dukungan konfigurasi yang fleksibel. Salah satu fitur utama BIND9 adalah DNS caching yang berfungsi

menyimpan hasil query sebelumnya sehingga permintaan yang sama dapat dilayani lebih cepat tanpa melakukan pencarian ulang ke server tujuan .

Meskipun mekanisme caching pada BIND9 mampu meningkatkan performa DNS, strategi pengelolaan cache yang digunakan masih memiliki keterbatasan. Penghapusan data cache (cache eviction) pada DNS umumnya lebih banyak dipengaruhi oleh parameter Time To Live (TTL), ukuran cache, dan waktu kedaluwarsa data dibandingkan dengan pola frekuensi akses suatu domain [5]. Akibatnya, domain yang sering digunakan oleh pengguna dapat mengalami penghapusan dari cache apabila masa penyimpanan berakhir, sementara domain yang jarang diakses masih dapat menempati ruang cache. Kondisi tersebut menyebabkan kemungkinan terjadinya cache miss yang berpengaruh terhadap peningkatan waktu respons DNS. [6],[7].

Salah satu pendekatan yang dapat digunakan untuk mengatasi permasalahan pengelolaan cache adalah algoritma Least Frequently Used (LFU). Algoritma LFU merupakan metode penggantian cache yang mempertahankan data berdasarkan jumlah frekuensi akses [8]. Dalam mekanisme LFU, objek atau data yang paling sering digunakan akan diprioritaskan untuk tetap berada dalam cache, sedangkan data dengan tingkat penggunaan rendah akan dikeluarkan terlebih dahulu ketika kapasitas cache penuh. Strategi ini sesuai dengan karakteristik pola akses internet yang umumnya mengikuti distribusi Zipf, yaitu sebagian kecil domain populer menerima sebagian besar permintaan pengguna dibandingkan domain lainnya [9],[10].

Penerapan algoritma LFU pada sistem caching DNS memiliki potensi untuk meningkatkan efisiensi penggunaan cache. Dengan mempertahankan record DNS yang sering diminta oleh pengguna, jumlah cache hit dapat meningkat dan jumlah permintaan yang harus diteruskan ke server eksternal dapat berkurang [11] . Penurunan jumlah query eksternal tersebut dapat mengurangi waktu pemrosesan DNS sehingga response time menjadi lebih rendah. Beberapa penelitian terkait optimasi caching menunjukkan bahwa strategi berbasis pola akses mampu memberikan peningkatan performa layanan jaringan dengan mengurangi latency dan meningkatkan efisiensi penggunaan sumber daya server [12].

Namun, implementasi optimasi cache berbasis frekuensi akses pada DNS server BIND9 masih menjadi topik yang perlu dikaji lebih lanjut. BIND9 secara bawaan belum menyediakan mekanisme LFU sebagai algoritma utama dalam pengelolaan cache, sehingga diperlukan pendekatan optimasi tambahan untuk melakukan evaluasi terhadap pengaruh algoritma tersebut. Pengujian secara langsung pada lingkungan Debian 12 [13] diperlukan untuk mengetahui seberapa besar peningkatan performa yang dapat diperoleh dibandingkan konfigurasi DNS standar.

Penelitian ini berfokus pada implementasi dan analisis optimasi DNS server berbasis BIND9 pada sistem operasi Debian 12 dengan menerapkan konsep algoritma Least Frequently Used (LFU) Cache. Pengujian dilakukan dengan membandingkan kinerja DNS server sebelum dan setelah penerapan optimasi berdasarkan parameter response time, jumlah query berhasil dilayani, serta tingkat efisiensi cache pada beberapa skenario jumlah klien. Variasi beban pengujian dilakukan menggunakan 20, 50, dan 100 klien untuk menggambarkan kondisi jaringan dengan tingkat permintaan yang berbeda.

Berdasarkan uraian tersebut, penelitian ini bertujuan untuk mengetahui pengaruh penerapan algoritma LFU Cache terhadap performa DNS server BIND9 dalam meningkatkan kecepatan resolusi domain. Rumusan masalah pada penelitian ini adalah apakah penerapan algoritma Least Frequently Used (LFU) Cache mampu menurunkan response time DNS server BIND9 secara signifikan dibandingkan dengan konfigurasi cache standar pada Debian 12. Hasil penelitian diharapkan dapat menjadi referensi dalam pengembangan optimasi layanan DNS yang lebih efisien, terutama pada lingkungan jaringan yang memiliki pola akses domain berulang dengan jumlah pengguna yang tinggi.

2. Metode Penelitian

Penelitian ini menerapkan pendekatan kuantitatif eksperimental dengan metode performance benchmarking untuk mengevaluasi pengaruh penerapan algoritma Least Frequently Used (LFU) Cache pada layanan DNS berbasis BIND9. Pendekatan eksperimental dipilih karena penelitian ini melakukan pengujian secara langsung terhadap sistem DNS dengan memberikan perlakuan berupa perubahan mekanisme pengelolaan cache, kemudian membandingkan hasil kinerja sebelum dan sesudah penerapan metode tersebut.

DOI: https://doi.org/10.31004/riggs.v5i2.11459

Pada penelitian ini terdapat dua jenis variabel yang diamati. Variabel independen dalam penelitian adalah penerapan algoritma LFU Cache serta jumlah permintaan klien yang berjalan secara bersamaan (concurrent client). Sementara itu, variabel dependen yang digunakan sebagai indikator keberhasilan sistem adalah nilai response time DNS yang diukur dalam satuan milidetik (ms). Nilai response time digunakan sebagai parameter utama karena menunjukkan seberapa cepat server DNS memberikan jawaban terhadap permintaan resolusi nama domain.

Pengujian dilakukan dengan membandingkan dua kondisi sistem, yaitu kondisi baseline dan kondisi eksperimen. Kondisi baseline merupakan pengujian terhadap server DNS BIND9 dengan mekanisme caching standar tanpa optimasi LFU. Sedangkan kondisi eksperimen merupakan pengujian setelah dilakukan penerapan mekanisme LFU Cache pada pengelolaan data cache DNS. Setiap skenario pengujian dilakukan sebanyak lima kali pengulangan untuk memperoleh hasil yang lebih stabil serta mengurangi kemungkinan adanya pengaruh faktor acak selama proses pengujian.

2.1. Lingkungan Pengujian

Lingkungan pengujian dibangun menggunakan sistem operasi Debian sebagai platform server DNS. Pada server tersebut dilakukan instalasi dan konfigurasi BIND9 (Berkeley Internet Name Domain versi 9 yang berfungsi sebagai recursive DNS resolver. Pemilihan BIND9 didasarkan pada kemampuannya dalam melakukan proses resolusi domain, menyimpan hasil query pada cache, serta menyediakan parameter konfigurasi yang dapat disesuaikan untuk kebutuhan optimasi performa.

Proses pembangkitan trafik dilakukan menggunakan perangkat lunak DNS benchmarking yang mendukung pengiriman query dalam jumlah besar dan waktu bersamaan [14]. Tools tersebut digunakan untuk mensimulasikan kondisi penggunaan nyata, dimana banyak pengguna melakukan permintaan resolusi domain dalam waktu yang sama. Setiap permintaan yang dikirimkan akan dicatat waktu responnya sehingga dapat dilakukan perbandingan performa antara sistem sebelum dan sesudah penerapan LFU Cache.

2.2. Implementasi LFU Cache pada BIND9

Implementasi LFU Cache dilakukan dengan mengoptimasi parameter konfigurasi named.conf dan menambahkan mekanisme tracking frekuensi akses per domain. Parameter utama yang dikonfigurasi meliputi max-cache-size (disesuaikan kapasitas memori server), max-cache-ttl (mengontrol masa berlaku cache), dan min-cache-ttl (memastikan domain populer tetap tersimpan). Algoritma mencatat counter frekuensi setiap domain; saat cache penuh, domain dengan frekuensi terendah dikeluarkan terlebih dahulu [15], [16].

2.3. Prosedur Pengujian

Pengujian dilakukan dalam tiga skenario berdasarkan jumlah klien simultan: 20 klien, 50 klien, dan 100 klien. Setiap skenario diulang sebanyak lima kali (Uji 1 hingga Uji 5) untuk memastikan konsistensi hasil. Sebelum setiap skenario pengujian, cache DNS dikosongkan untuk memastikan kondisi awal yang seragam. Permintaan DNS yang diuji mencakup 500 domain unik yang dipilih secara acak dari daftar Alexa Top 1 Juta domain.

Metrik utama yang diukur adalah response time dalam milidetik (ms), yang didefinisikan sebagai waktu antara pengiriman permintaan DNS oleh klien hingga penerimaan respons dari server. Pengukuran dilakukan menggunakan tool dnsperf yang berjalan di sisi klien dengan resolusi waktu mikrodetik [17].

Tabel 1. Perangkat Lunak dan Perangkat Keras Pendukung

Komponen	Spesifikasi
Hypervisor	Solaris 2.X
Sistem Operasi Server	Debian 12 (64-bit)
DNS Server	BIND9
CPU	1 vCPU
RAM Server	1 GB
Penyimpanan	20 GB Virtual Disk
Forwader DNS	8.8.8.8
Jumlah Beban Client	20, 50, dan 100 client

2.4. Konfigurasi DNS Server

Konfigurasi DNS Server dilakukan dengan menggunakan perangkat lunak BIND9 pada sistem operasi Debian 12. Pada tahap ini dilakukan pengaturan file konfigurasi utama BIND9 yaitu named.conf.options yang digunakan untuk mengatur layanan DNS, mekanisme query, serta pengelolaan cache pada server. Konfigurasi awal dilakukan

DOI: https://doi.org/10.31004/riggs.v5i2.11459

dengan mengaktifkan fitur recursion agar DNS Server dapat melakukan pencarian domain secara bertingkat apabila data yang diminta belum tersedia pada cache lokal.

Pada konfigurasi DNS Server ditambahkan parameter forwarders yang berfungsi untuk meneruskan permintaan query ke DNS eksternal ketika data domain tidak ditemukan pada cache lokal. Pada penelitian ini digunakan DNS eksternal Google Public DNS dengan alamat 8.8.8.8 sebagai forwarder. Penggunaan forwarder bertujuan untuk mempercepat proses resolusi domain serta mengurangi beban pencarian langsung dari DNS Server lokal ke server tujuan.

Selain pengaturan forwarder, dilakukan konfigurasi mekanisme caching pada BIND9 dengan menambahkan beberapa parameter cache seperti max-cache-size, max-cache-ttl, min-cache-ttl, dan prefetch [16]. Parameter max-cache-size digunakan untuk menentukan batas penggunaan memori yang dialokasikan untuk penyimpanan cache DNS. Parameter max-cache-ttl dan min-cache-ttl digunakan untuk mengatur waktu penyimpanan record DNS di dalam cache, sedangkan parameter prefetch digunakan untuk melakukan pembaruan data sebelum masa cache berakhir terhadap domain yang sering diakses.

Konfigurasi tersebut bertujuan untuk meningkatkan efisiensi proses resolusi DNS dengan memanfaatkan penyimpanan cache secara maksimal. Dengan adanya cache, permintaan terhadap domain yang sama tidak perlu melakukan proses pencarian ulang ke DNS eksternal sehingga dapat mengurangi waktu respon (response time) dan meningkatkan performa layanan DNS Server. Konfigurasi forwarders dan cache pada file named.conf.options ditunjukkan pada Gambar 1.

Gambar 1. Konfigurasi forwarders dan cache pada file named.conf.options

Selain konfigurasi utama DNS Server, dilakukan pembuatan file daftar query yang digunakan sebagai data pengujian performa. File tersebut diberi nama queryfile.txt yang berisi kumpulan domain yang akan digunakan dalam proses pengujian menggunakan tools dnsperf. Domain yang digunakan terdiri dari beberapa website populer seperti google.com, youtube.com, facebook.com, instagram.com, twitter.com, netflix.com, dan yahoo.com.

Penggunaan domain populer dalam pengujian bertujuan untuk mensimulasikan pola akses pengguna yang sering melakukan permintaan terhadap domain tertentu. Data query tersebut digunakan untuk menghasilkan beban permintaan DNS sehingga dapat dilakukan pengukuran terhadap kemampuan caching BIND9 sebelum dan sesudah penerapan algoritma Least Frequently Used (LFU) Cache. Daftar domain pada file queryfile.txt ditunjukkan pada Gambar 2.

DOI: https://doi.org/10.31004/riggs.v5i2.11459

Gambar 2. Daftar domain pada file queryfile.txt

Setelah seluruh konfigurasi DNS Server selesai dilakukan, layanan BIND9 dijalankan ulang menggunakan perintah restart service untuk menerapkan perubahan konfigurasi. Selanjutnya dilakukan verifikasi menggunakan tools dig dan nslookup untuk memastikan DNS Server dapat melakukan proses resolusi domain dengan baik sebelum masuk ke tahap pengujian performa dan analisis response time.

Tahap selanjutnya adalah penerapan optimasi cache menggunakan algoritma Least Frequently Used (LFU) Cache. Mekanisme LFU digunakan untuk mempertahankan record DNS yang memiliki frekuensi akses tinggi agar tetap berada di dalam cache. Dengan mempertahankan domain yang sering diminta oleh client, diharapkan jumlah cache hit meningkat sehingga response time DNS Server dapat mengalami penurunan dibandingkan konfigurasi cache standar.

3. Hasil dan Diskusi

Pengujian dilakukan secara sistematis untuk membandingkan performa DNS server BIND9 dalam dua kondisi: dengan implementasi algoritma LFU Cache dan tanpa implementasi LFU Cache. Hasil pengukuran response time disajikan dalam bentuk tabel untuk memudahkan analisis perbandingan.

3.1. Hasil Pengujian dengan Algoritma LFU

Tabel 2 menyajikan hasil pengujian response time DNS server BIND9 yang telah dioptimasi dengan algoritma LFU Cache untuk ketiga skenario jumlah klien. Kolom Hasil merupakan rata-rata dari lima kali pengujian.

Tabel 2. Pengujian dengan Algoritma LFU

Client	Uji 1	Uji 2	Uji 3	Uji 4	Uji 5	Hasil
20	48 ms	42 ms	39 ms	35 ms	31 ms	39.0 ms
50	36 ms	31 ms	28 ms	26 ms	24 ms	29.0 ms
100	71 ms	65 ms	59 ms	54 ms	51 ms	60.0 ms

Dari Tabel 2 terlihat bahwa response time cenderung menurun seiring bertambahnya jumlah pengujian dalam setiap skenario. Hal ini disebabkan oleh efek pemanasan cache (cache warm-up) 24], di mana entri-entri yang sering diakses mulai terakumulasi dalam cache LFU sehingga hit rate meningkat dari waktu ke waktu. Pada skenario 100 klien, terjadi penurunan yang paling signifikan dari 71 ms pada Uji 1 menjadi 51 ms pada Uji 5.

3.2. Hasil Pengujian Tanpa Algoritma LFU

Tabel 3 menyajikan hasil pengujian response time DNS server BIND9 dengan konfigurasi standar tanpa implementasi algoritma LFU Cache. Pengujian dilakukan pada kondisi dan infrastruktur yang identik dengan pengujian sebelumnya.

DOI: https://doi.org/10.31004/riggs.v5i2.11459

Tabel 3. Pengujian Tanpa Algoritma LFU

Client	Uji 1	Uji 2	Uji 3	Uji 4	Uji 5	Hasil
20	65 ms	44 ms	48 ms	55 ms	47 ms	51.8 ms
50	46 ms	54 ms	46 ms	52 ms	14 ms	42.4 ms
100	146 ms	146 ms	125 ms	107 ms	111 ms	127.0 ms

Tanpa implementasi LFU, response time menunjukkan variabilitas yang lebih tinggi antar pengujian, khususnya pada skenario 20 klien di mana nilai berkisar antara 44 ms hingga 65 ms. Hal ini mengindikasikan bahwa mekanisme caching default BIND9 kurang konsisten dalam mempertahankan entri yang sering diakses. Pada skenario 100 klien, rata-rata response time mencapai 127.0 ms, yang menunjukkan penurunan performa signifikan ketika beban meningkat.

3.3. Perbandingan Hasil

Tabel 4 merangkum perbandingan rata-rata response time antara kondisi tanpa LFU dan dengan LFU untuk ketiga skenario jumlah klien.

Tabel 4. Perbandingan LFU dan Tanpa LFU

Client	Tanpa LFU	Dengan LFU
20	51.8 ms	39.0 ms
50	42.4 ms	29.0 ms
100	127.0 ms	60.0 ms

Berdasarkan Tabel 4, implementasi algoritma LFU Cache memberikan peningkatan performa yang konsisten pada semua skenario. Untuk skenario 20 klien, response time berkurang dari 51.8 ms menjadi 39.0 ms, setara dengan peningkatan efisiensi sebesar 24.7%. Pada skenario 50 klien, penurunan terjadi dari 42.4 ms menjadi 29.0 ms (31.6%). Peningkatan paling signifikan terjadi pada skenario 100 klien, di mana response time turun dari 127.0 ms menjadi 60.0 ms, merepresentasikan peningkatan sebesar 52.8%.
Pola peningkatan yang semakin besar seiring meningkatnya jumlah klien menunjukkan bahwa algoritma LFU semakin efektif pada kondisi beban tinggi. Ini dapat dijelaskan dengan fakta bahwa pada beban tinggi, entri cache yang sering diakses memiliki frekuensi yang jauh lebih tinggi dibandingkan entri yang jarang diakses, sehingga algoritma LFU dapat membuat keputusan penggantian yang lebih optimal [18], [19].

Hasil ini sejalan dengan penelitian Meghanathan [20] yang menunjukkan keunggulan LFU pada workload dengan distribusi akses yang tidak merata (skewed distribution). Dalam konteks DNS, distribusi akses domain memang sangat tidak merata, di mana sebagian kecil domain populer mendominasi sebagian besar query. Kondisi ini sangat menguntungkan algoritma LFU yang dirancang untuk mempertahankan entri dengan frekuensi tinggi [21],[22].

4. Kesimpulan

Penelitian ini berhasil mengimplementasikan dan mengevaluasi algoritma Least Frequently Used (LFU) Cache pada DNS server berbasis BIND9 yang berjalan di sistem operasi Debian. Hasil pengujian secara konsisten menunjukkan bahwa implementasi LFU Cache meningkatkan response time DNS secara signifikan dibandingkan konfigurasi standar tanpa LFU. Peningkatan response time yang dicapai adalah 24.7% untuk 20 klien simultan, 31.6% untuk 50 klien, dan 52.8% untuk 100 klien. Temuan ini mengkonfirmasi bahwa algoritma LFU sangat efektif dalam skenario beban tinggi, di mana kemampuannya mempertahankan entri domain yang paling sering diakses menghasilkan cache hit rate yang lebih tinggi dan mengurangi kebutuhan query ke server upstream. Implikasi praktis dari penelitian ini adalah bahwa administrator jaringan dapat meningkatkan kualitas layanan DNS secara signifikan hanya dengan mengoptimalkan strategi caching, tanpa memerlukan peningkatan hardware. Untuk penelitian lebih lanjut, disarankan untuk menguji kombinasi LFU dengan algoritma caching lain seperti Adaptive Replacement Cache (ARC), serta mengevaluasi dampak pada jaringan dengan pola akses domain yang berbeda-beda.

DOI: https://doi.org/10.31004/riggs.v5i2.11459

Referensi

[1] C. Mou, "DNS is the Internet Pivotal Basics and Fundamental," *Int. J. Adv. Network, Monit. Control.*, vol. 7, no. 2, pp. 11–23, 2022, doi: 10.2478/ijanmc-2022-0012.
[2] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "DNS Performance and the Effectiveness of Caching," 1997.
[3] K. F and K. Y, "DNS Tra ffi c Analysis — CDN and the World IPv6 Launch," vol. 21, no. 3, pp. 517–526, 2013, doi: 10.2197/ipsjjip.21.517.
[4] K. Scarfone, P. Mell, and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS) Recommendations of the National Institute of Standards and Technology".
[5] Y. KANNO, "An introduction to information security evaluation," *J. Inf. Process. Manag.*, vol. 48, no. 6, pp. 320–332, 2005, doi: 10.1241/johokanri.48.320.
[6] K. Fujiwara, A. Sato, and K. Yoshida, "Cache Effect of Shared DNS Resolver," no. 6, pp. 1170–1179, 2019, doi: 10.1587/transcom.2018EBP3184.
[7] G. C. M. Moura, J. Heidemann, and W. Hardaker, "Cache Me If You Can : Effects of DNS Time-to-Live," no. 1.
[8] M. I. Zulfa, A. E. Permanasari, A. Fadli, and W. Ali, "Performance comparison of cache replacement algorithms on various internet traffic," vol. 15, no. 1, pp. 1–7, 2023.
[9] Q. H. Dang, "Secure Hash Standard," *FIBS 180-4 Publ.*, vol. 4, no. August, p. 36, 2015.
[10] F. T. Industri and U. I. Indonesia, "MELALUI PENGGUNAAN FIREWALL DENGAN," 2023.
[11] S. Hao and H. Wang, "Exploring Domain Name-Based Features on the Effectiveness of DNS Caching," vol. 47, no. 1, 2017.
[12] N. Keren, G. Einziger, G. Scalosub, and G. Einziger, "Latency-Aware Caching with Delayed Hits : From Bursty Traffic to Pipeline Architectures," 2026.
[13] G. Lencse, A. Pivoda, and K. Shima, "Performance evaluation of DNS servers to build a benchmarking system of DNS64 implementations," *Telecommun. Syst.*, vol. 77, no. 4, pp. 643–653, 2021, doi: 10.1007/s11235-021-00780-3.
[14] G. Lencse, "Benchmarking Authoritative DNS Servers," vol. XX, 2020, doi: 10.1109/ACCESS.2020.3009141.
[15] L. A. Saputra, F. M. Akbar, F. Cahyaningtias, and M. Puspa, "Ancaman Keamanan Pada Sistem Informasi Manajemen Perusahaan," vol. 1, no. 2, pp. 58–66, 2023.
[16] I. S. Consortium, "BIND 9 Administrator Reference".
[17] V. Date, "DNS Performance dnsperf Tool Manual," 2012.
[18] B. Ullrich and J. Wang, "Optical properties of PbS quantum dots deposited on glass employing a supercritical CO2 fluid process," *Appl. Sci.*, vol. 9, no. 21, pp. 1–7, 2019, doi: 10.3390/app9214567.
[19] P. K. Shah, "An O (1) algorithm for implementing the LFU cache eviction scheme," no. 1, pp. 1–8, 2010.
[20] F. María, V. Pardo, and M. G. Esteban, "Classification of species of Stipa with awns having plumose distal segments," vol. 13, pp. 155–176.
[21] C. Florin, R. Moreau-gobard, J. Williams, and E. Nationale, "Automatic heart peripheral vessels segmentation based on a normal MIP ray casting technique".
[22] N. A. Saputra, R. H. Irawan, and U. Mahdiyah, "Hybrid Ensemble Learning Sistem Keamanan Jaringan untuk Meningkatkan Performa Deteksi Anomali," vol. 8, no. 2, pp. 361–369, 2025.

DOI: https://doi.org/10.31004/riggs.v5i2.11459