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: 16413- 16420
P-ISSN: 2963-9298, e-ISSN: 2963-914X

Optimasi DNS Server Berbasis BIND9 pada Debian dengan Menggunakan Algoritma Least Recently Used Cache untuk Meningkatkan Response Time

Noperla AnjeliSari Lumbanbatu[1], Yadi Limanta Maha[2], Lonia Harefa[3], Lotar Mateus Sinaga[4]
[1,2,3,4] Teknik Informatika, Fakultas Ilmu Komputer, Universitas Katolik Santo Thomas Medan
[1]noverlaanjelii@gmail.com, [2]mahalimanta05@gmail.com, [3]harefaleonial@gmail.com, [4]lotarmateus88@gmail.com

Abstrak

Perkembangan teknologi jaringan yang semakin pesat menuntut sistem yang mampu memberikan layanan dengan cepat dan efisien, khususnya dalam proses resolusi Domain Name System (DNS). Salah satu permasalahan yang sering terjadi adalah tingginya response time akibat banyaknya permintaan query yang tidak tersimpan dalam cache, sehingga memperlambat akses pengguna dan meningkatkan beban kerja server dalam melakukan resolusi ulang ke DNS eksternal. Kondisi ini menjadi tantangan tersendiri bagi administrator jaringan dalam menjaga kualitas layanan, terutama pada sistem dengan volume permintaan yang tinggi. Oleh karena itu, diperlukan suatu metode untuk mengoptimalkan kinerja DNS server melalui pengelolaan cache yang lebih efektif. Penelitian ini bertujuan untuk menganalisis pengaruh penerapan algoritma Least Recently Used (LRU) terhadap performa DNS server berbasis BIND9 pada sistem operasi Debian 12. Metode yang digunakan adalah pendekatan eksperimen dengan membandingkan performa DNS server sebelum dan sesudah penerapan algoritma LRU pada lingkungan virtual menggunakan Oracle VirtualBox. Pengujian dilakukan menggunakan tools dnsperf dengan variasi jumlah client, yaitu 20, 50, dan 100, dan setiap skenario diulang lima kali untuk memperoleh nilai rata-rata yang lebih akurat. Parameter utama yang diukur adalah response time yang ditunjukkan melalui nilai average latency. Hasil penelitian menunjukkan bahwa penerapan algoritma LRU tidak selalu memberikan peningkatan performa pada semua kondisi. Pada beban rendah, performa tanpa LRU cenderung lebih baik, sedangkan pada beban tinggi (100 client), algoritma LRU mampu menurunkan response time secara signifikan. Dengan demikian, algoritma LRU lebih efektif digunakan pada kondisi dengan jumlah permintaan yang tinggi, sehingga dapat meningkatkan efisiensi kinerja DNS server pada sistem jaringan skala besar.

Kata Kunci: *DNS Server, BIND9, LRU, Cache, Response Time*

1. Latar Belakang

Peningkatan penggunaan layanan berbasis internet menuntut sistem jaringan yang mampu memberikan respon secara cepat dan efisien. Salah satu aspek penting dalam kinerja jaringan adalah proses resolusi Domain Name System (DNS) yang berpengaruh langsung terhadap kecepatan akses pengguna. DNS merupakan sistem yang digunakan untuk menerjemahkan nama domain menjadi alamat IP sehingga memudahkan komunikasi dalam jaringan [1]. Selain itu, struktur dan pengelolaan DNS juga menentukan bagaimana distribusi domain dilakukan secara global [2].

Beberapa penelitian menunjukkan bahwa performa DNS sangat dipengaruhi oleh mekanisme pengelolaan cache. Rendahnya cache hit rate menyebabkan meningkatnya latensi karena server harus melakukan query ulang ke resolver eksternal. Sebaliknya, pengelolaan cache yang optimal dapat meningkatkan efisiensi resolusi DNS dan menurunkan waktu respon secara signifikan [5]. Mekanisme cache yang baik juga berperan dalam meningkatkan performa sistem jaringan secara keseluruhan [4].

Permasalahan yang sering terjadi pada DNS server adalah tingginya response time akibat banyaknya permintaan query yang diproses secara berulang. Kondisi ini menyebabkan peningkatan beban server serta memperlambat proses resolusi domain, terutama ketika data tidak tersedia dalam cache lokal. Hal ini menunjukkan bahwa pengelolaan cache menjadi faktor penting dalam meningkatkan performa DNS server.

Salah satu pendekatan yang dapat digunakan untuk meningkatkan efisiensi cache adalah algoritma Least Recently Used (LRU), yang bekerja dengan mempertahankan data yang sering diakses dan menghapus data yang jarang

digunakan. Algoritma LRU merupakan salah satu metode pengelolaan cache yang banyak digunakan dalam sistem komputer untuk meningkatkan efisiensi akses data [6]. Pendekatan ini terbukti mampu meningkatkan cache hit rate dan mempercepat proses akses data dalam berbagai sistem berbasis cache.

Dalam implementasinya, DNS server dapat dikonfigurasi menggunakan perangkat lunak BIND9 yang merupakan salah satu DNS server yang banyak digunakan karena stabil dan fleksibel dalam berbagai kebutuhan jaringan [3]. Selain itu, pengembangan DNS juga terus dilakukan untuk meningkatkan performa dan skalabilitas melalui berbagai ekstensi [7].

Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk mengoptimasi kinerja DNS server berbasis BIND9 pada sistem operasi Debian melalui penerapan algoritma Least Recently Used (LRU). Pengujian dilakukan menggunakan tools dnsperf untuk menganalisis pengaruh metode tersebut terhadap peningkatan response time DNS server.

Beberapa penelitian terdahulu telah mengembangkan berbagai varian algoritma cache untuk meningkatkan efisiensi sistem berbasis permintaan tinggi. Wang et al. mengusulkan LR-LRU, sebuah kebijakan penggantian cache yang mengombinasikan faktor waktu akses dan frekuensi penggunaan untuk sistem berorientasi PACS. Sementara itu, Feng et al. memperkenalkan pendekatan Least Popularly Used sebagai alternatif LRU pada jaringan berbasis informasi (Information-Centric Networking), dengan hasil yang menunjukkan peningkatan cache hit rate pada kondisi beban tertentu. Penelitian-penelitian tersebut menunjukkan bahwa efektivitas suatu algoritma cache sangat bergantung pada karakteristik beban dan pola akses data pada sistem yang diterapkan. Berbeda dengan penelitian-penelitian tersebut yang berfokus pada sistem PACS dan ICN, penelitian ini secara khusus menguji penerapan algoritma LRU pada konteks DNS server berbasis BIND9, dengan membandingkan performa secara langsung pada tiga variasi beban client (20, 50, dan 100) untuk melihat titik di mana algoritma tersebut mulai memberikan manfaat nyata.

2. Metode Penelitian

Penelitian ini menggunakan pendekatan eksperimen untuk menganalisis pengaruh penerapan algoritma Least Recently Used (LRU) terhadap kinerja DNS server berbasis BIND9. Pengujian dilakukan dengan membandingkan response time DNS server sebelum dan sesudah penerapan mekanisme pengelolaan cache. Tahapan penelitian meliputi perancangan lingkungan pengujian, konfigurasi DNS server, pelaksanaan pengujian, serta analisis hasil pengujian.

2.1. Lingkungan dan Topologi Pengujian

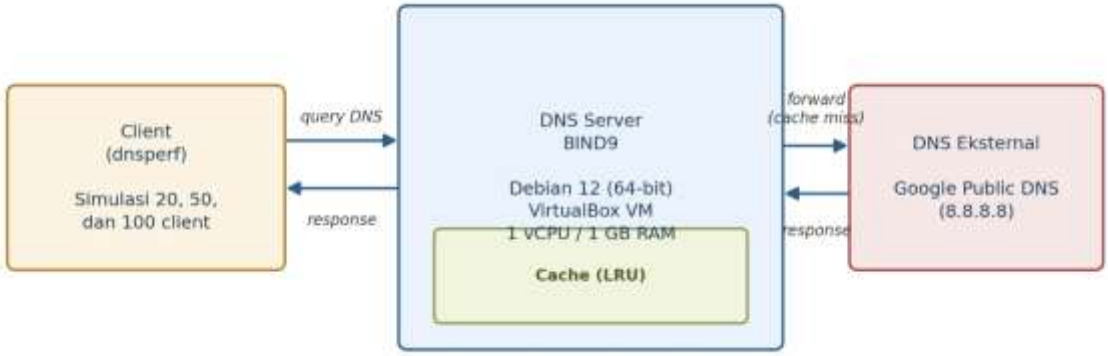
Lingkungan pengujian dibangun menggunakan Oracle VirtualBox dengan satu mesin virtual yang menjalankan sistem operasi Debian 12 (64-bit). DNS server dikonfigurasi menggunakan perangkat lunak BIND9. Sistem dijalankan dalam jaringan lokal dan terhubung ke DNS eksternal melalui mekanisme forwarders untuk menangani query yang tidak tersedia dalam cache lokal, yang umum digunakan dalam konfigurasi DNS untuk meningkatkan efisiensi resolusi [5].

Topologi pengujian menggunakan satu server DNS berbasis BIND9 yang menerima permintaan query dari client dan meneruskan permintaan tersebut ke DNS eksternal apabila data tidak ditemukan dalam cache lokal. Pendekatan ini banyak digunakan dalam implementasi DNS resolver untuk meningkatkan kecepatan akses dan efisiensi jaringan [6].

Selain digunakan untuk menggambarkan hubungan antar perangkat, topologi pengujian ini juga dirancang agar proses pengukuran dapat dilakukan secara terkontrol. Seluruh permintaan DNS berasal dari satu client yang menjalankan *tools* dnsperf, sehingga setiap skenario pengujian dilakukan pada lingkungan yang sama tanpa adanya perubahan konfigurasi jaringan. Dengan demikian, perbedaan nilai *response time* yang diperoleh lebih merepresentasikan pengaruh penerapan algoritma Least Recently Used (LRU) dibandingkan pengaruh faktor lain seperti perubahan topologi atau spesifikasi perangkat.

Penggunaan topologi sederhana juga bertujuan untuk memudahkan proses evaluasi mekanisme cache pada DNS server. Seluruh permintaan query diproses terlebih dahulu oleh server BIND9, kemudian dibandingkan apakah data dapat dilayani dari cache lokal atau harus diteruskan ke DNS eksternal melalui mekanisme *forwarders*. Alur komunikasi tersebut memungkinkan proses pengamatan terhadap kinerja DNS server dilakukan secara lebih sistematis, sehingga hasil pengujian dapat dianalisis berdasarkan kondisi sebelum dan sesudah penerapan algoritma LRU.

DOI: https://doi.org/10.31004/riggs.v5i2.11086
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)



Gambar 1. Topologi Pengujian DNS Server Berbasis BIND9

Seperti ditunjukkan pada Gambar 1, komunikasi diawali dari sisi client yang menggunakan tools dnsperf untuk mengirimkan query DNS dengan tiga skenario simulasi jumlah client, yaitu 20, 50, dan 100. Query tersebut diterima oleh DNS server berbasis BIND9 yang berjalan pada mesin virtual Debian 12, dan diperiksa terlebih dahulu terhadap data yang tersimpan pada cache dengan mekanisme Least Recently Used (LRU). Apabila terjadi cache hit, server akan langsung mengirimkan response ke client. Sebaliknya, apabila terjadi cache miss, permintaan akan diteruskan (forward) ke DNS eksternal, yaitu Google Public DNS (8.8.8.8), sebelum hasilnya dikembalikan ke client."

Lingkungan pengujian ini dirancang untuk mensimulasikan kondisi jaringan sederhana yang memungkinkan pengukuran performa DNS server secara terkontrol. Dengan konfigurasi ini, proses pengujian dapat dilakukan secara konsisten sehingga hasil yang diperoleh dapat menggambarkan pengaruh penerapan algoritma terhadap kinerja sistem.

Tabel 1. Spesifikasi Lingkungan Pengujian

Komponen	Spesifikasi
Hypervisor	Oracle VirtualBox
Sistem Operasi	Debian 12 (64-bit)
DNS Server	BIND9
Tools Uji	dnsperf
CPU	1 vCPU
RAM	1 GB
Forwarder DNS	8.8.8.8

Spesifikasi lingkungan pengujian yang digunakan dalam penelitian ini ditunjukkan pada Tabel 1, meliputi penggunaan 1 vCPU dan RAM sebesar 1 GB pada mesin virtual. Alokasi sumber daya ini dipilih untuk merepresentasikan kondisi server dengan spesifikasi terbatas, sehingga pengaruh penerapan algoritma LRU terhadap efisiensi pengelolaan cache dapat teramati lebih jelas dibandingkan pada server dengan sumber daya besar. Penggunaan satu unit vCPU juga bertujuan untuk mengontrol variabel pengujian agar perbedaan response time yang diperoleh benar-benar mencerminkan pengaruh algoritma cache, bukan disebabkan oleh perbedaan kapasitas komputasi.

2.2. Konfigurasi DNS Server

Konfigurasi DNS server dilakukan dengan mengatur file *named.conf.options* pada perangkat lunak BIND9. Pada konfigurasi ini ditambahkan parameter *forwarders* yang digunakan untuk meneruskan permintaan query ke DNS eksternal, yaitu 8.8.8.8 (Google Public DNS). Dengan adanya parameter ini, apabila DNS server lokal tidak memiliki data dalam cache, maka permintaan akan langsung diteruskan ke DNS eksternal sehingga proses resolusi domain dapat dilakukan dengan lebih cepat.

Selain itu, dilakukan pengaturan parameter cache seperti *max-cache-size*, *max-cache-ttl*, *min-ncache-ttl*, *min-cache-ttl*, dan *prefetch* untuk mengoptimalkan kinerja DNS server. Parameter *max-cache-size* digunakan untuk membatasi penggunaan memori cache, sedangkan *max-cache-ttl* dan *min-cache-ttl* mengatur lama penyimpanan data dalam cache. Parameter *prefetch* digunakan untuk mempercepat pengambilan data yang sering diakses sebelum masa cache berakhir.

DOI: https://doi.org/10.31004/riggs.v5i2.11086
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

Konfigurasi ini bertujuan untuk meningkatkan efisiensi DNS server dalam menangani permintaan query dengan memanfaatkan cache secara optimal serta mengurangi beban kerja server dalam melakukan resolusi ulang ke DNS eksternal, di tunjukkan pada Gambar 2.

```
 GNU nano 7.2              /etc/bind/named.conf.options
options {
        directory "/var/cache/bind";
        max-cache-size 256m;
        max-cache-ttl 300;
        min-ncache-ttl 60;
        min-cache-ttl 10;
        prefetch 5 20;

        recursion yes;
        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        forwarders {
        8.8.8.8;
```

Gambar 2. Konfigurasi forwarders pada file named.conf.options

Selain konfigurasi DNS server, dibuat file queryfile.txt yang berisi daftar domain yang digunakan sebagai data pengujian, seperti google.com, youtube.com, facebook.com, instagram.com, twitter.com, netflix.com, dan yahoo.com. File ini berfungsi sebagai input dalam proses pengujian performa DNS server menggunakan dnsperf. Daftar domain yang digunakan dalam pengujian ditunjukkan pada Gambar 3.

Pemilihan domain-domain tersebut didasarkan pada tingkat popularitas dan volume trafik yang tinggi pada penggunaan internet sehari-hari, sehingga dapat merepresentasikan pola permintaan DNS yang umum terjadi pada jaringan nyata. Dengan menggunakan domain-domain populer sebagai data uji, hasil pengujian diharapkan dapat mencerminkan kondisi beban kerja DNS server pada skenario penggunaan yang realistis, baik pada organisasi kecil maupun jaringan dengan jumlah pengguna yang lebih besar.

```
 GNU nano 7.2                    /root/queryfile.txt *
google.com A
youtube.com A
facebook.com A
instagram.com A
twitter.com A
netflix.com A
yahoo.com A
```

Gambar 3. Daftar domain pada file queryfile.txt

Setelah konfigurasi selesai dilakukan, layanan BIND9 dijalankan kembali untuk memastikan seluruh pengaturan yang telah dilakukan dapat diterapkan dengan baik sebelum dilakukan proses pengujian.

2.3. Parameter dan Skenario Pengujian

Pengujian dilakukan untuk mengukur performa DNS server dengan menggunakan tools dnsperf. Parameter utama yang diukur dalam penelitian ini adalah response time DNS server dalam menangani permintaan query dari client. Response time digunakan sebagai indikator untuk mengetahui kecepatan server dalam melakukan proses resolusi domain.

Pengujian dilakukan dengan menggunakan file queryfile.txt sebagai input yang berisi daftar domain yang telah disiapkan sebelumnya. Proses pengujian dilakukan dengan mengirimkan sejumlah permintaan query ke DNS server dan mencatat hasil performa yang dihasilkan.

DOI: https://doi.org/10.31004/riggs.v5i2.11086
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

Untuk memperoleh hasil yang lebih akurat, setiap skenario pengujian dilakukan sebanyak lima kali percobaan, kemudian diambil nilai rata-rata sebagai hasil akhir. Variasi jumlah client yang digunakan dalam pengujian adalah 20, 50, dan 100 untuk melihat pengaruh beban terhadap performa DNS server.

Untuk mengetahui pengaruh penerapan algoritma Least Recently Used (LRU), pengujian dilakukan dengan membandingkan performa DNS server sebelum dan sesudah penerapan mekanisme pengelolaan cache.

Nilai response time yang dilaporkan dalam penelitian ini diperoleh dari output dnsperf berupa average latency, yaitu rata-rata waktu yang dibutuhkan server untuk merespons seluruh query yang berhasil diproses dalam satu sesi pengujian. Nilai ini dihitung dengan membagi total waktu respons seluruh query terhadap jumlah query yang berhasil dijawab oleh server. Semakin rendah nilai average latency yang dihasilkan, semakin cepat dan efisien kinerja DNS server dalam menangani permintaan query dari client.

3. Hasil dan Diskusi

Pengujian dilakukan untuk mengevaluasi performa DNS server berbasis BIND9 dengan menggunakan tools dnsperf. Parameter utama yang diamati dalam penelitian ini adalah response time yang ditunjukkan melalui nilai average latency dalam menangani permintaan query dari client.

Pengujian dilakukan dengan variasi jumlah client untuk mengetahui pengaruh beban terhadap kinerja DNS server sebelum dan sesudah penerapan algoritma Least Recently Used (LRU). Hasil pengujian disajikan pada Tabel.

Tabel 2. Pengujian dengan Algoritma LRU

Client	Uji 1	Uji 2	Uji 3	Uji 4	Uji 5	hasil
20	169ms	95ms	85ms	62ms	63ms	94.8ms
50	91ms	51ms	49ms	48ms	45ms	56.8ms
100	82ms	72ms	85ms	74ms	85ms	79.6ms

Tabel 3. Pengujian Tanpa Algoritma LRU

Client	Uji 1	Uji 2	Uji 3	Uji 4	Uji 5	hasil
20	65ms	44ms	48ms	55ms	47ms	51.8ms
50	46ms	54ms	46ms	52ms	14ms	42.2ms
100	146ms	146ms	125ms	107ms	111ms	127.0.ms

Tabel 4. Perbandingan dengan LRU dan Tanpa LRU

Client	Tanpa LRU	Dengan LRU
20	50.8ms	94.8ms
50	42.4ms	56.8ms
100	127.ms	79.6ms

Pengujian dilakukan sebanyak lima kali pada setiap variasi jumlah *client*, yaitu 20, 50, dan 100 *client*, untuk memperoleh nilai rata-rata *response time* yang lebih representatif. Pengujian pertama dilakukan dengan menerapkan algoritma Least Recently Used (LRU) pada DNS server berbasis BIND9. Hasil pengujian pada Tabel 2 menunjukkan bahwa nilai rata-rata *response time* yang diperoleh sebesar 94,8 ms pada 20 *client*, 56,8 ms pada 50 *client*, dan 79,6 ms pada 100 *client*. Hasil tersebut menunjukkan bahwa performa DNS server mengalami perubahan pada setiap variasi jumlah *client*, dimana algoritma LRU mampu mempertahankan performa yang lebih stabil ketika jumlah permintaan semakin meningkat.

Sebagai pembanding, dilakukan pengujian tanpa menerapkan algoritma LRU dengan skenario pengujian yang sama. Berdasarkan hasil pada Tabel 3, diperoleh nilai rata-rata *response time* sebesar 51,8 ms pada 20 *client*, 42,2 ms pada 50 *client*, dan 127,0 ms pada 100 *client*. Hasil tersebut menunjukkan bahwa DNS server tanpa algoritma LRU memiliki performa yang lebih baik pada jumlah *client* yang rendah. Namun, ketika jumlah *client* meningkat menjadi 100, nilai *response time* mengalami peningkatan yang cukup signifikan sehingga menunjukkan penurunan performa akibat meningkatnya beban permintaan.

Perbandingan hasil pengujian pada Tabel 4 menunjukkan bahwa penerapan algoritma LRU tidak selalu menghasilkan performa yang lebih baik pada setiap kondisi. Pada pengujian dengan 20 dan 50 client, DNS server tanpa algoritma LRU menghasilkan nilai *response time* yang lebih rendah dibandingkan dengan penggunaan algoritma LRU. Sebaliknya, pada pengujian dengan 100 client, algoritma LRU mampu menurunkan nilai *response time* dari 127,0 ms menjadi 79,6 ms. Hasil ini menunjukkan bahwa mekanisme pengelolaan cache menggunakan

DOI: https://doi.org/10.31004/riggs.v5i2.11086
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

algoritma LRU mulai bekerja secara efektif ketika DNS server menerima jumlah permintaan yang lebih besar, sehingga proses resolusi domain menjadi lebih cepat dibandingkan konfigurasi tanpa algoritma LRU.

Selain disajikan dalam bentuk tabel, hasil pengujian juga ditampilkan melalui keluaran (*output*) tools dnsperf sebagaimana ditunjukkan pada Gambar 4. Output tersebut menampilkan informasi statistik pengujian, seperti jumlah *queries sent*, *queries completed*, *queries lost*, *queries per second*, serta nilai *average latency*. Pada penelitian ini, parameter *average latency* digunakan sebagai indikator utama dalam mengukur *response time* DNS server karena menggambarkan rata-rata waktu yang dibutuhkan server untuk memberikan respons terhadap setiap permintaan query dari *client*. Nilai *average latency* tersebut selanjutnya digunakan sebagai dasar dalam analisis performa DNS server sebelum dan sesudah penerapan algoritma Least Recently Used (LRU).

Selain dalam bentuk tabel, hasil pengujian juga ditunjukkan melalui output tools dnsperf pada Gambar 3.

```
[Timeout] Query timed out: msg id 28531
[Timeout] Query timed out: msg id 28532
[Timeout] Query timed out: msg id 28533
[Status] Testing complete (time limit)

Statistics:

  Queries sent:         33121
  Queries completed:    33052 (99.79%)
  Queries lost:         69 (0.21%)

  Response codes:       NOERROR 33052 (100.00%)
  Average packet size:  request 29, response 79
  Run time (s):         60.181492
  Queries per second:   549.285394

  Average Latency (s):  0.169985 (min 0.021679, max 2.437696)
  Latency StdDev (s):   0.241691

root@yadi:~# dnsperf -s 192.168.10.2 -d /root/queryfile.txt -c 20 -l 60
DNS Performance Testing Tool
Version 2.10.0

[Status] Command line: dnsperf -s 192.168.10.2 -d /root/queryfile.txt -c 20 -l 6
```

Gambar 4. Hasil pengujian DNS server menggunakan dnsperf

Berdasarkan Gambar 4, *tools* dnsperf berhasil menampilkan hasil pengujian berupa informasi statistik yang digunakan untuk mengevaluasi performa DNS server. Statistik tersebut meliputi jumlah *queries sent*, *queries completed*, *queries lost*, *queries per second*, serta nilai *average latency*. Dari hasil pengujian terlihat bahwa sebagian besar permintaan query berhasil diproses oleh DNS server dengan tingkat keberhasilan yang tinggi, sedangkan jumlah *queries lost* relatif kecil sehingga tidak memberikan pengaruh yang signifikan terhadap hasil pengujian.

Pada penelitian ini, parameter utama yang digunakan sebagai acuan analisis adalah nilai average latency, karena parameter tersebut menunjukkan rata-rata waktu yang dibutuhkan DNS server dalam memberikan respons terhadap setiap permintaan query dari *client*. Nilai *average latency* yang diperoleh dari *dnsperf* selanjutnya digunakan sebagai dasar dalam menghitung *response time* pada setiap skenario pengujian, baik sebelum maupun sesudah penerapan algoritma Least Recently Used (LRU). Berdasarkan hasil tersebut, analisis lebih lanjut mengenai pengaruh algoritma LRU terhadap performa DNS server dijelaskan pada subbab berikut.

3.1. Analisis Response Time

Berdasarkan hasil pengujian pada Tabel 4, terlihat bahwa performa DNS server menunjukkan hasil yang berbeda pada setiap variasi jumlah client. Pada jumlah client 20, response time tanpa penerapan algoritma LRU tercatat sebesar 50.8 ms, sedangkan dengan penerapan LRU justru meningkat menjadi 94.8 ms, atau naik sekitar 87%. Pola serupa juga terjadi pada jumlah client 50, di mana response time tanpa LRU sebesar 42.4 ms meningkat menjadi 56.8 ms setelah penerapan LRU, atau naik sekitar 34%. Kondisi ini menunjukkan bahwa pada beban ringan hingga menengah, mekanisme cache LRU belum memberikan dampak positif, bahkan cenderung menambah waktu proses akibat overhead pengelolaan cache.

Namun, pada jumlah client yang lebih tinggi yaitu 100, nilai response time mengalami penurunan yang cukup signifikan setelah penerapan algoritma LRU, dari 127.0 ms menjadi 79.6 ms, atau turun sekitar 37%. Hal ini menunjukkan bahwa algoritma LRU mulai bekerja secara optimal ketika jumlah permintaan meningkat, karena

DOI: https://doi.org/10.31004/riggs.v5i2.11086
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

semakin banyak client yang mengakses domain yang sama secara berulang, sehingga data yang tersimpan dalam cache semakin sering dimanfaatkan sebelum digantikan oleh data baru.

3.2. Pembahasan Pengaruh LRU terhadap Performa DNS

Hasil pengujian menunjukkan bahwa algoritma Least Recently Used (LRU) tidak selalu memberikan peningkatan performa pada semua kondisi. Pada beban rendah, penggunaan LRU cenderung belum optimal karena data yang diakses belum sering berulang, sehingga mekanisme cache tidak memberikan keuntungan yang signifikan dan bahkan menambah overhead pada sistem.

Sebaliknya, pada kondisi beban tinggi, algoritma LRU mampu meningkatkan performa dengan menyimpan data yang sering diakses di dalam cache. Hal ini menyebabkan proses resolusi domain menjadi lebih cepat karena tidak perlu melakukan query ulang ke DNS eksternal.

Peningkatan response time pada beban rendah ini dapat dijelaskan dari sisi mekanisme kerja algoritma LRU itu sendiri. Setiap kali terjadi permintaan query, sistem tetap harus melakukan pemeriksaan dan pembaruan struktur data cache untuk menentukan entri mana yang paling baru diakses, meskipun tingkat pengulangan permintaan (cache hit rate) masih rendah. Proses pemeriksaan dan pembaruan inilah yang menimbulkan overhead komputasi tambahan, sehingga pada kondisi beban ringan manfaat cache belum sebanding dengan biaya pemrosesan yang dikeluarkan.

Dengan demikian, dapat disimpulkan bahwa penerapan algoritma LRU lebih efektif digunakan pada kondisi dengan jumlah permintaan yang tinggi, dan kurang optimal pada kondisi beban rendah. Hal ini menunjukkan bahwa efektivitas algoritma LRU sangat bergantung pada kondisi beban sistem.

Secara keseluruhan, hasil penelitian ini menunjukkan bahwa mekanisme pengelolaan cache memiliki peran penting dalam meningkatkan efisiensi dan kinerja DNS server, terutama dalam menghadapi jumlah permintaan query yang besar.

Temuan ini sejalan dengan hasil penelitian Feng et al. yang menunjukkan bahwa efektivitas algoritma penggantian cache sangat bergantung pada pola akses dan tingkat pengulangan permintaan pada sistem yang diuji. Pada sistem dengan pola akses yang belum stabil atau frekuensi pengulangan rendah, algoritma berbasis LRU cenderung kurang optimal dibandingkan pendekatan tanpa cache. Hal ini menegaskan pentingnya mempertimbangkan karakteristik beban kerja sebelum menerapkan mekanisme cache tertentu pada sistem produksi.

4. Kesimpulan

Berdasarkan hasil pengujian dan analisis yang telah dilakukan terhadap DNS server berbasis BIND9, dapat disimpulkan bahwa penerapan algoritma Least Recently Used (LRU) memberikan pengaruh terhadap performa response time, namun tidak selalu menghasilkan peningkatan performa pada semua kondisi. Pada kondisi jumlah client yang rendah, yaitu 20 dan 50, performa DNS server tanpa penerapan algoritma LRU menunjukkan hasil yang lebih baik dibandingkan dengan menggunakan LRU. Hal ini disebabkan karena frekuensi permintaan yang masih rendah sehingga mekanisme cache belum bekerja secara optimal. Sebaliknya, pada kondisi jumlah client yang lebih tinggi yaitu 100, penerapan algoritma LRU mampu meningkatkan performa DNS server dengan menurunkan nilai response time secara signifikan. Hal ini menunjukkan bahwa algoritma LRU lebih efektif dalam mengelola cache ketika jumlah permintaan meningkat dan data yang sama sering diakses. Dengan demikian, dapat disimpulkan bahwa algoritma Least Recently Used (LRU) lebih efektif digunakan pada DNS server dengan beban tinggi, namun kurang optimal pada kondisi beban rendah. Oleh karena itu, penerapan algoritma LRU dalam optimasi DNS server perlu disesuaikan dengan karakteristik beban jaringan yang dihadapi. Penelitian ini memiliki beberapa keterbatasan yang perlu diperhatikan. Pertama, pengujian dilakukan pada lingkungan virtual dengan spesifikasi sumber daya terbatas (1 vCPU dan 1 GB RAM), sehingga hasil yang diperoleh mungkin berbeda apabila diterapkan pada server produksi dengan kapasitas yang lebih besar. Kedua, jumlah domain yang digunakan dalam pengujian relatif terbatas dan hanya mencakup situs-situs populer, sehingga belum sepenuhnya merepresentasikan variasi pola akses pada jaringan skala besar. Ketiga, penelitian ini hanya membandingkan algoritma LRU dengan kondisi tanpa cache, tanpa melibatkan algoritma penggantian cache lain sebagai pembanding.Untuk penelitian selanjutnya, disarankan untuk menguji algoritma pengelolaan cache lain seperti Least Frequently Used (LFU) atau Adaptive Replacement Cache (ARC) guna memperoleh perbandingan yang lebih komprehensif. Selain itu, pengujian pada lingkungan jaringan produksi nyata dengan variasi jumlah client yang lebih luas (di atas 100) juga perlu dilakukan untuk memvalidasi konsistensi hasil penelitian ini pada skala yang lebih besar.

DOI: https://doi.org/10.31004/riggs.v5i2.11086
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

Referensi

1. P. Mockapetris, “Domain Names - Implementation and Specification,” RFC 1035, Nov. 1987.
2. J. Postel, “Domain Name System Structure and Delegation,” RFC 1591, Mar. 1994.
3. Internet Systems Consortium (ISC), BIND 9 Administrator Reference Manual. St. Paul, MN, USA: ISC, 2024.
4. A. S. Tanenbaum and D. J. Wetherall, Computer Networks, 6th ed. London, UK: Pearson, 2021.
5. García, K. Hynek, D. Vekshin, T. Čejka, dan A. Wasicek, “Large Scale Measurement on the Adoption of Encrypted DNS,” 2021.
6. N. Kernan, J. Li, R. Al-Dalky, dan M. Rabinovich, “Public DNS Resolvers Meet Content Delivery Networks: A Performance Assessment of the Interplay,” 2025.
7. J. Lee, S. Hoque, A. Aydeger, dan E. Zeydan, “Quantum-Resistant Domain Name System: A Comprehensive System-Level Study,” 2025.
8. D. A. Patterson dan J. L. Hennessy, Computer Organization and Design, 6th ed. Cambridge, MA, USA: Morgan Kaufmann, 2021.
9. Google, “Google Public DNS Documentation,” 2024. [Online]. Available: https://developers.google.com/speed/public-dns
10. Cloudflare, “1.1.1.1 Public DNS Resolver Documentation,” 2024. [Online]. Available: https://developers.cloudflare.com/1.1.1.1/
11. Y. Wang, Y. Yang, C. Han, L. Ye, Y. Ke, and Q. Wang, "LR-LRU: A PACS-Oriented Intelligent Cache Replacement Policy,"2019.
12. B. Feng, H. Zhou, G. Li, H. Zhang, and H. Chao, "Least Popularly Used: A Cache Replacement Policy for Information-Centric Networking," *Journal of Internet Technology*,2016.
13. E. Friedlander and V. Aggarwal, "Generalization of LRU Cache Replacement Policy with Applications to Video Streaming," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*,2018.
14. D. Xiao, X. Li, D. Cline, and D. Loguinov, "Estimating DNS Source and Cache Dynamics under Interval-Censored Age Sampling," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 10, pp. 1–29, 2025.
15. G. Merugu, K. Gopal, J. U. S. Charan, and P. Devi, "Geo-Intelligent DNS Resolver Using Deep Learning," in *Proc. 2025 International Conference on Computing Technologies & Data Communication (ICCTDC)*, 2025, pp. 1–6.
16. M. Chopra, A. Dhawan, S. K. Behera, S. Kumar, A. Visvanathan, and R. Majji, "DNS Response Time Analysis Across Diverse Top-Level Domains," *Journal of Internet Services and Information Security*, 2025.
17. F. Yang, C. Wang, B. Zheng, and Y. Tian, "DNS Service Optimization Through In-Network Caching on Programmable Switches," in *Proc. 2024 10th International Conference on Computer and Communications (ICCC)*, 2024, pp. 1568–1572.
18. F. Zhang, B. Liu, E. Alowaisheq, J. Chen, C. Lu, L. Song, Y. Liu, H. Duan, and M. Yang, "Silence is not Golden: Disrupting the Load Balancing of Authoritative DNS Servers," in *Proc. 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023.
19. K. Man, X. Zhou, and Z. Qian, "DNS Cache Poisoning Attack: Resurrections with Side Channels," in *Proc. 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.
20. D. Kartika, R. Riska, and Y. Mardiana, "DNS Server and Web Server Simulation with Debian Operating System on Local Area Network," *Jurnal Media Computer Science*, 2023.
21. T. Jinmei and P. Vixie, "Implementation and Evaluation of Moderate Parallelism in the BIND9 DNS Server," 2006.

DOI: https://doi.org/10.31004/riggs.v5i2.11086
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)