Neptunus : Jurnal Ilmu Komputer Dan Teknologi Informasi
Volume. 2 No. 3 Agustus 2024
e-ISSN : 3031-898X, dan p-ISSN : 3031-8998, Hal. 285-294
DOI: https://doi.org/10.61132/neptunus.v2i3.280
Available online at: https://journal.arteii.or.id/index.php/Neptunus

Optimasi dan Modifikasi Debian untuk Meningkatkan Kinerja Sistem Operasi *Real-Time*

Rakhmadi Rahman[1], Reski Septiawan[2*], Abriel Yosua Nathanael Leksona[3]
[1,2,3]Institut Teknologi Bacharuddin Jusuf Habibie Parepare, Parepare, Sulawesi Selatan, Indonesia
Jurusan Sistem Informasi Fakultas Sains

Alamat : Jalan Pemuda No.6 Kota Parepare, Sulawesi Selatan, Indonesia
Korespondensi penulis : *reskysptwn89@gmail.com

Abstract: *In today's digital era, operating systems play a crucial role in various technological devices. This paper discusses the optimization and modification of Debian, one of the most stable Linux distributions, to enhance its performance as a real-time operating system (RTOS). The primary objectives of this research are to improve the responsiveness, reliability, and stability of Debian in handling real-time tasks. The study involves modifying the kernel, adjusting scheduling settings, optimizing memory management, and integrating specific real-time software. The results of the tests show that the modified Debian provides lower and more stable task execution latency compared to the standard Debian. Tests were conducted using the cyclictest software to measure system latency and Htop to monitor CPU and memory performance. Additionally, the use of the VLC application as a real-world workload demonstrated that the optimized operating system could handle task priorities more efficiently, allocate resources as needed for real-time demands, and maintain stability under heavy workloads. This research significantly contributes to the development of Debian-based RTOS, which can be applied in fields such as industrial automation, robotics, and IoT devices. The optimizations ensure that the operating system can meet the high-performance and real-time reliability demands of these critical applications.*

Keywords*: Debian, Real-Time Operating System (RTOS), Optimization, Kernel Modification, Latency, System Performance.*

Abstrak: Dalam era digital saat ini, sistem operasi memiliki peran penting dalam berbagai perangkat teknologi. Makalah ini membahas optimasi dan modifikasi Debian, salah satu distribusi Linux yang stabil, untuk meningkatkan kinerja sebagai sistem operasi waktu nyata (RTOS). Tujuan utama dari penelitian ini adalah meningkatkan responsivitas, keandalan, dan stabilitas Debian dalam menangani tugas-tugas real-time. Penelitian ini melibatkan modifikasi kernel, pengaturan penjadwalan, optimasi manajemen memori, dan integrasi perangkat lunak spesifik real-time. Hasil pengujian menunjukkan bahwa Debian yang telah dimodifikasi mampu memberikan latensi eksekusi tugas yang lebih rendah dan stabil dibandingkan Debian normal. Pengujian dilakukan menggunakan perangkat lunak cyclictest untuk mengukur latensi sistem dan Htop untuk memantau kinerja CPU dan memori. Selain itu, penggunaan aplikasi VLC sebagai beban kerja nyata menunjukkan bahwa sistem operasi yang dioptimasi dapat menangani prioritas tugas dengan lebih efisien, mengalokasikan sumber daya sesuai kebutuhan real-time, dan mempertahankan stabilitas di bawah beban kerja tinggi. Penelitian ini memberikan kontribusi signifikan dalam pengembangan RTOS berbasis Debian, yang dapat diaplikasikan pada bidang otomasi industri, robotika, dan perangkat IoT. Optimasi yang dilakukan memastikan sistem operasi dapat memenuhi tuntutan performa tinggi dan keandalan waktu nyata yang dibutuhkan oleh aplikasi-aplikasi kritis tersebut.

Kata Kunci: Debian, *Real-Time Operating System (RTOS),* Optimasi, Modifikasi Kernel, Latensi, Performa Sistem.

Received: Juni 25, 2024; Revised: Juni 10, 2024; Accepted: Juli 24, 2024; Online Available: Juli 26, 2024;

1. PENDAHULUAN

Di era digital saat ini, perangkat teknologi sudah dilengkapi dengan berbagai sistem operasi yang memiliki keunikan tersendiri. Sistem operasi seperti Windows, MacOS, dan Linux adalah nama-nama umum yang sering kita jumpai dan gunakan setiap waktu. Namun, dalam kinerjanya, sistem operasi dibedakan menjadi dua tipe yaitu GPOS (General Purpose Operating System) dan RTOS (Real-Time Operating System). Pada kesempatan kali ini, kami berfokus pada real-time operating system. Sistem operasi waktu nyata menggunakan deadline (batas waktu) dalam mengeksekusi sebuah task. Jika proses eksekusi telah melewati batas waktu yang telah ditentukan, maka proses tersebut dinyatakan sebagai kegagalan. RTOS tidak hanya berfokus pada hasil (output) proses eksekusi, tetapi juga diprioritaskan agar dapat bekerja dengan baik dalam waktu yang dibutuhkan.

Dalam upaya meningkatkan kinerja real-time operating system, modifikasi Debian menjadi salah satu pendekatan yang efektif. Debian, sebagai salah satu distribusi Linux yang paling stabil dan berguna, menawarkan fondasi yang kokoh untuk dioptimalkan menjadi RTOS. Alasan memilih Debian untuk real-time adalah karena stabilitas dan keandalannya yang tinggi, hasil dari proses pengujian yang ketat sebelum rilis. Versi terbaru, Debian 12 (Bookworm), membawa peningkatan performa dan efisiensi yang signifikan, dukungan perangkat keras yang lebih baik, serta peningkatan keamanan dengan update rutin. Selain itu, Debian memiliki repositori paket yang sangat luas, memungkinkan instalasi dan konfigurasi perangkat lunak spesifik untuk kebutuhan real-time dengan mudah.

Dengan melakukan modifikasi, kita dapat menyesuaikan kernel dan komponen sistem lainnya agar lebih responsif terhadap kebutuhan real-time. Ini termasuk penyesuaian scheduler, optimasi manajemen memori, dan integrasi perangkat lunak real-time yang spesifik. Melalui pendekatan ini, kita dapat menciptakan sebuah sistem operasi yang tidak hanya stabil dan aman, tetapi juga mampu memenuhi tuntutan kinerja yang ketat dari aplikasi real-time. Hal ini membuka peluang bagi berbagai aplikasi baru di bidang otomasi industri, robotika, dan perangkat IoT yang membutuhkan performa tinggi dan keandalan waktu nyata. Sistem operasi (OS) adalah perangkat lunak yang mengelola perangkat keras komputer dan menyediakan layanan umum untuk program aplikasi. Sistem operasi bertindak sebagai perantara antara pengguna dan perangkat keras komputer. Contoh umum dari sistem operasi meliputi Microsoft Windows, macOS, dan berbagai distribusi Linux seperti Debian. (Hapzi & Database, 2024)

RTOS adalah jenis sistem operasi yang dirancang untuk menangani aplikasi yang memerlukan waktu real-time. RTOS bekerja dalam waktu nyata dengan batasan deterministik yang membutuhkan penggunaan waktu dan daya yang efisien. (Esther et al., 2020)

Debian adalah distribusi Linux yang bersifat open-source dan gratis untuk digunakan. Debian dikenal karena stabilitasnya dan banyak digunakan sebagai basis untuk distribusi Linux lainnya. Debian menggunakan kernel Linux dan menawarkan berbagai paket perangkat lunak yang dapat diinstal sesuai kebutuhan pengguna. Versi terbaru yang digunakan dalam penelitian ini adalah Debian 12 (Bookworm). Dibandingkan dengan versi sebelumnya, Debian 12, menawarkan peningkatan yang signifikan dalam efisiensi dan kinerja, dukungan perangkat keras yang lebih baik, dan peningkatan keamanan melalui update rutin. Selain itu, repositorinya lebih besar dan mendukung lebih banyak paket perangkat lunak terbaru. Dibandingkan dengan distribusi Linux lainnya seperti Ubuntu atau Fedora, Debian menawarkan stabilitas dan keandalan yang lebih tinggi berkat proses pengujian yang ketat sebelum rilis. Meskipun Ubuntu sering berfokus pada kemudahan penggunaan, sedangkan Fedora berfokus pada teknologi terbaru, Debian menawarkan kombinasi terbaik dari stabilitas, keamanan, kinerja, dan dukungan komunitas, menjadikannya pilihan unggul untuk berbagai kebutuhan, termasuk aplikasi real-time.

Kernel adalah inti dari sistem operasi yang mengelola operasi dasar sistem, seperti manajemen proses, manajemen memori, dan kontrol perangkat keras. Kernel Linux adalah salah satu kernel paling populer yang digunakan dalam berbagai distribusi Linux. Modifikasi kernel untuk keperluan real-time melibatkan penyesuaian scheduler, manajemen memori, dan pengaturan interrupt untuk mengurangi latensi dan meningkatkan responsivitas. Optimasi sistem operasi melibatkan berbagai teknik untuk meningkatkan kinerja sistem. Dalam konteks RTOS, optimasi bertujuan untuk meminimalkan latensi, meningkatkan kecepatan respons, dan memastikan keandalan sistem di bawah beban kerja yang tinggi. Teknik-teknik ini meliputi isolasi CPU, pengaturan interrupt handling, penggunaan penjadwalan real-time, dan penyetelan parameter kernel. Pengujian kinerja adalah proses mengevaluasi kinerja sistem operasi dengan menggunakan alat dan metode tertentu. Dalam penelitian ini, alat yang digunakan meliputi cyclictest untuk mengukur latensi sistem dan Htop untuk memantau penggunaan CPU dan memori. Pengujian dilakukan untuk membandingkan kinerja Debian standar dengan Debian yang telah dioptimasi untuk real-time. Aplikasi real-time adalah program yang memerlukan waktu respons yang ketat dan prediktabilitas tinggi dalam operasinya. Contoh aplikasi real-time

meliputi sistem kontrol industri, robotika, dan perangkat IoT (Internet of Things). Penggunaan aplikasi real-time dalam pengujian membantu menilai efektivitas optimasi yang dilakukan pada sistem operasi.

2. METODE PENELITIAN

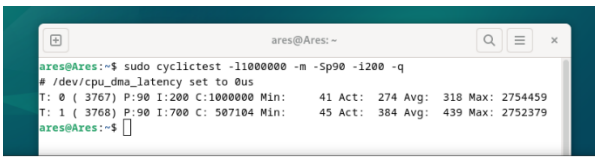
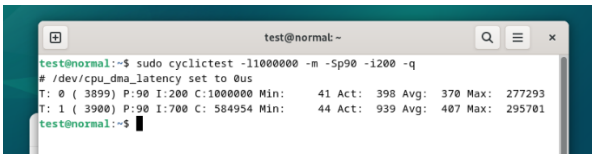
Metode yang digunakan dalam penelitian ini yaitu Penelitian kepustakaan atau studi literatur, di mana penulis mengandalkan literatur untuk mendapatkan data penelitian dan menggunakan pendekatan kualitatif karena data berupa kata atau deskripsi. Setelah mengumpulkan literatur yang relevan dengan penelitian, penulis melakukan implementasi dan pengujian untuk memastikan bahwa penulis memiliki pemahaman yang mendalam tentang topik penelitian dan arah penelitian yang tepat.

3. HASIL DAN PEMBAHASAN

Penggunaan Cyclictest

Cyclictest adalah salah satu program didalam *package* rt-tests yang berfungsi untuk mengukur perbedaan antara waktu *deadline* thread dan saat thread mulai bekerja kembali untuk memberikan statistik tentang latensi sistem. Pada metode pertama ini, penulis menggunakan metode *cyclictest* untuk mengukur latensi sistem antara Debian normal dengan Debian real-time.

Tabel 4.1.1 Pengujian Spesifik per CPU

Menjalankan cyclictest untuk menguji latensi spesifik per CPU	
Jenis Debian	Hasil
Real-Time	
Normal	

Dari tabel diatas, terlihat bahwa kernel real-time menunjukkan performa latensi yang lebih baik dibanding kernel normal. Latensi minimum kedua kernel cukup mirip dibanding latensi maksimum, dimana latensi maksimum kernel real-time lebih rendah dibanding kernel normal, terutama pada Thread 1. Latensi aktual yang dihasilkan kernel

real-time juga lebih rendah dibandingkan kernel normal. Untuk latensi rata-rata, kernel real-time sedikit lebih baik pada Thread 0 namun sedikit buruk pada Thread 1 dibandingkan kernel normal. Meskipun kedua kernel menunjukkan thread kedua tidak mencapai jumlah loop yang sama dengan thread pertama, hal ini bisa disebabkan oleh berbagai faktor seperti adanya *interrupt* yang terjadi selama pengujian. Secara keseluruhan ditunjukkan bahwa kernel real-time memiliki kinerja latensi yang lebih baik dibandingkan dengan kernel normal. Selain cara sebelumnya, metode cyclictest dapat ditambahkan opsi '--smp' agar dapat dilakukan pengujian multiprosesor. Hasil ujinya dapat dilihat pada tabel berikut :

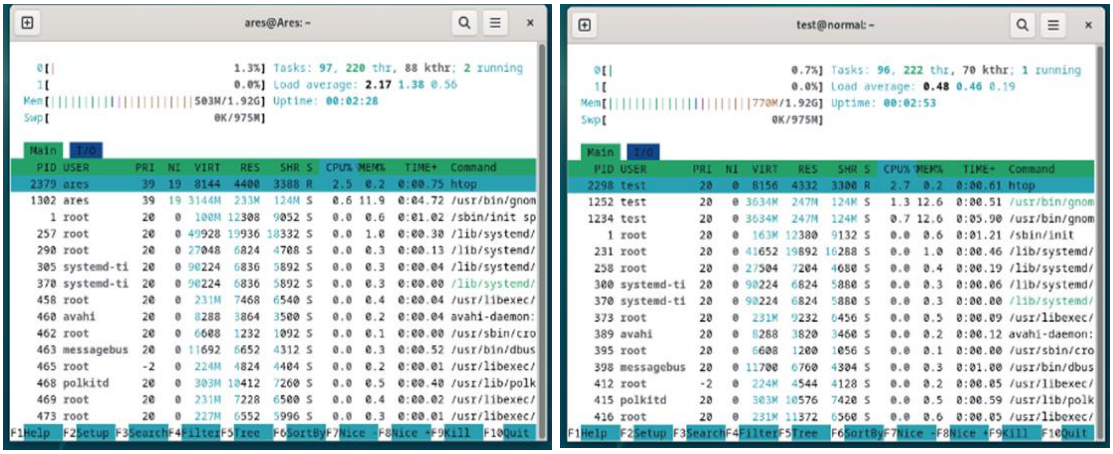
Tabel 4.1.2 Pengujian Secara Multiprosesor

Menjalankan cyclictest untuk menguji latensi secara multiprosesor	
Jenis Debian	Hasil
Real-Time	ares@Ares:~$ sudo cyclictest --smp -l1000000 -m -Sp90 -i200 -q [sudo] password for ares: # /dev/cpu_dma_latency set to 0us T: 0 (4419) P:90 I:200 C:1000000 Min: 40 Act: 966 Avg: 277 Max: 33897 T: 1 (4420) P:90 I:700 C: 525015 Min: 41 Act: 738 Avg: 366 Max: 34551 ares@Ares:~$
Normal	test@normal:~$ sudo cyclictest --smp -l1000000 -m -Sp90 -i200 -q [sudo] password for test: # /dev/cpu_dma_latency set to 0us T: 0 (4110) P:90 I:200 C:1000000 Min: 41 Act: 613 Avg: 340 Max: 49061 T: 1 (4111) P:90 I:700 C: 581479 Min: 43 Act: 101 Avg: 351 Max: 49571 test@normal:~$

Pada tabel diatas ditunjukkan bahwa latensi minimum kedua kernel memiliki perbedaan yang tidak terlalu signifikan meskipun kernel real-time memiliki latensi minimum yang sedikit lebih rendah. Untuk latensi aktual, kernel normal cenderung lebih rendah dibandingkan kernel real-time. Pada latensi rata-rata, kernel real-time lebih rendah pada Thread 0 tetapi lebih tinggi pada Thread 1 dibandingkan kernel normal. Namun, pada latensi maksimum, kernel real-time cenderung lebih rendah dibandingkan kernel normal pada kedua thread. Kedua kernel pada thread kedua (T: 1) tidak mencapai loop yang sama dengan thread pertama, tetapi kernel normal menunjukkan hasil yang sedikit lebih baik karena jumlah loop yang lebih banyak. Secara keseluruhan, kernel real-time menunjukkan performa lebih baik dalam hal latensi maksimum, sedangkan kernel normal menunjukkan hasil yang baik pada aspek lain. Namun, kernel normal mungkin mengalami banyak gangguan dan variasi yang mempengaruhi hasil latensi.

Penggunaan Htop

Dalam pengembangan aplikasi pemesanan digital seperti Grab, penggunaan kernel Linux adalah salah satu opsi yang digunakan. “Kernel Linux adalah bagian sentral dari sistem operasi Linux Ini adalah kernel sistem operasi monolitik mirip Unix yang menyediakan layanan penting dan mengelola sumber daya sistem” (Pratama, 2023). Android memanfaatkan kernel ini untuk menghubungkan perangkat seluler dengan aplikasi dan layanan. Kernel Linux menjadi salah satu pilihan yang digunakan karena fleksibilitasnya dan dukungan komunitas yang luas. Pengembangan sistem manajemen yang didukung aplikasi pemesanan digital seperti Grab memiliki beberapa komponen utama untuk diintegrasikan dengan layanan cloud. Grab menjalin kemitraan dengan Microsoft untuk menggunakan Azure sebagai platform komputasi awan. Dengan layanan cloud dan kecerdasan buatan (AI) Microsoft, Grab dapat secara efektif mengembangkan platformnya serta meningkatkan kapasitas dan fungsinya.



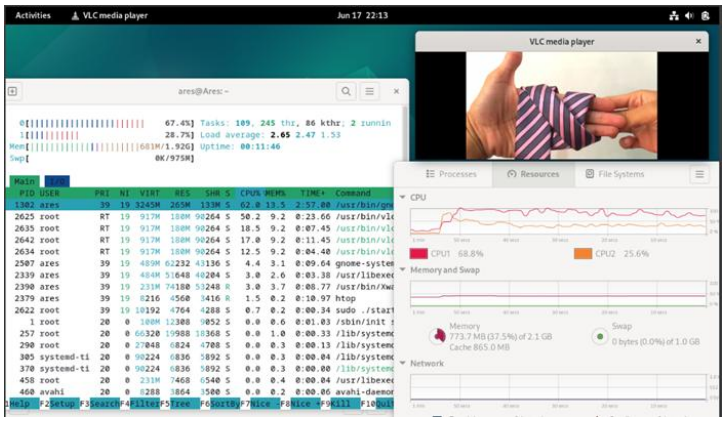
Gambar 4.2.1 Htop pada Kernel Real-Time (Kiri) & Kernel Normal (Kanan)

Pada metode kedua ini, penulis menggunakan metode htop untuk memantau dan mengelola sistem yang berjalan di Linux. Htop membantu penulis dalam menampilkan beban kerja saat proses bekerja pada kernel real-time dan normal. Htop adalah aplikasi interaktif untuk memantau daftar proses yang sedang berjalan pada sistem lengkap dengan informasi detail seperti penggunaan jumlah cpu ataupun memori.

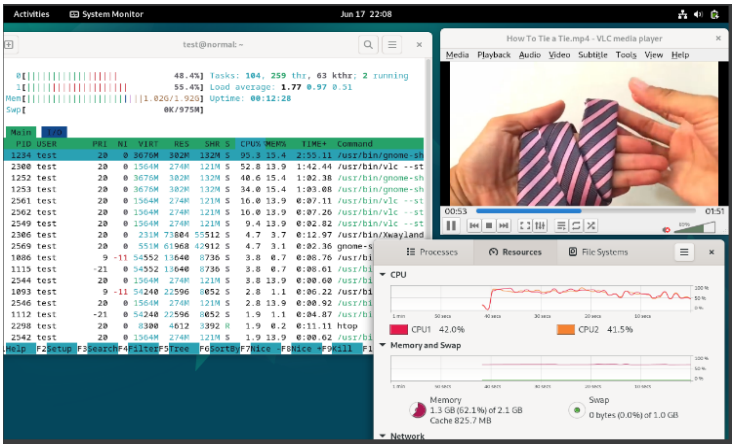
Dari gambar diatas, ditampilkan bahwa *load average* kernel real-time mencapai **2.17** pada interval waktu 1 menit pertama, sementara *uptime* sistem masih 2 menit. Ini menunjukkan bahwa ketika sistem baru dinyalakan, tugas dengan prioritas tertinggi diprioritaskan dan tugas lain ditunda. Selain itu, tampak bahwa karena sistem baru dihidupkan, %CPU belum terlalu tinggi. Untuk kernel normal, terlihat bahwa *load average* sistem pada interval 1 menit hanya mencapai **0.48** dengan uptime yang sama yaitu 2 menit.

Namun, ditunjukkan bahwa memory usage pada kernel normal mencapai 770M yang lebih tinggi dibanding kernel real-time. Ini menunjukkan bahwa dalam kernel normal, tugas tidak tertunda karena semuanya diproses secara bersamaan sehingga jumlah memori yang digunakan cukup besar selama proses.

Untuk menguji lebih lanjut, digunakan software VLC sebagai salah satu aplikasi dengan sistem real-time untuk melihat bagaimana beban kerja aplikasi tersebut saat dijalankan pada kernel real-time dan kernel normal :



Gambar 4.2.2 Pengujian VLC pada Kernel Real-Time

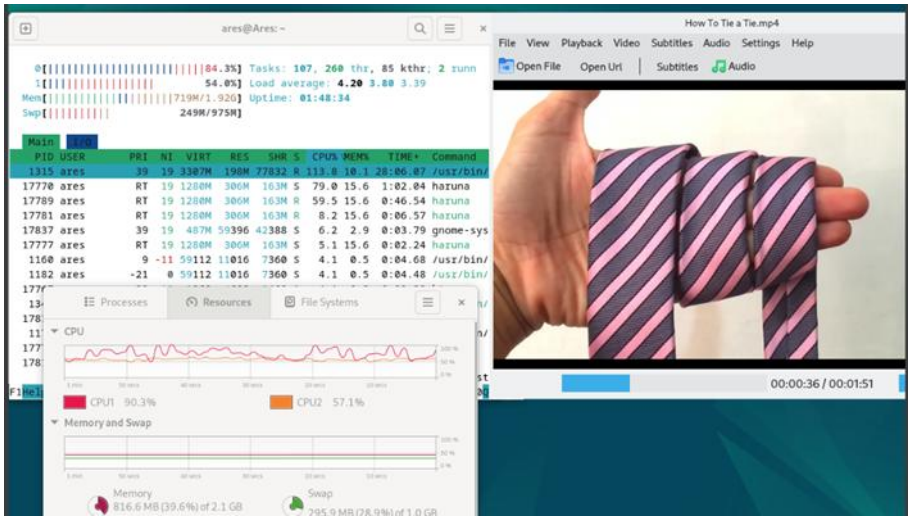


Gambar 4.2.3 Pengujian VLC pada Kernel Normal

Saat menjalankan aplikasi VLC dengan prioritas real-time, aplikasi tersebut berjalan dengan *priority* RT yang dilihat dari kolom ‘PRI’ pada terminal htop. Terlihat bahwa load average sistem saat menjalankan VLC beserta proses lainnya mencapai 2.65 pada interval 1 menit, 2.47 pada interval 5 menit, dan 1.53 pada interval 15 menit. Ini menunjukkan sama seperti sebelumnya bahwa sistem real-time mengeksekusi tugas dengan prioritas tertinggi, seperti aplikasi VLC, sementara tugas-tugas lain harus menunggu. Selama aplikasi berjalan, penggunaan memori sedikit meningkat menjadi 600-700M dari sebelumnya hanya ±500M. Selain itu, %CPU berubah selama aplikasi VLC

berjalan. Jika dilihat dari grafik, CPU 1 atau Cpu0 mencapai ±68%, yang terkadang meningkat tetapi tidak mencapai 50%, hanya berkisar antara 60% dan 100%. Sebaliknya, CPU 2 atau Cpu1 mencapai ±25%, yang terkadang meningkat tetapi tidak mencapai 50%. Oleh karena itu, CPU 1 memiliki persentase yang tinggi karena mengoptimalkan CPU 2 untuk mengerjakan tugas-tugas real-time.

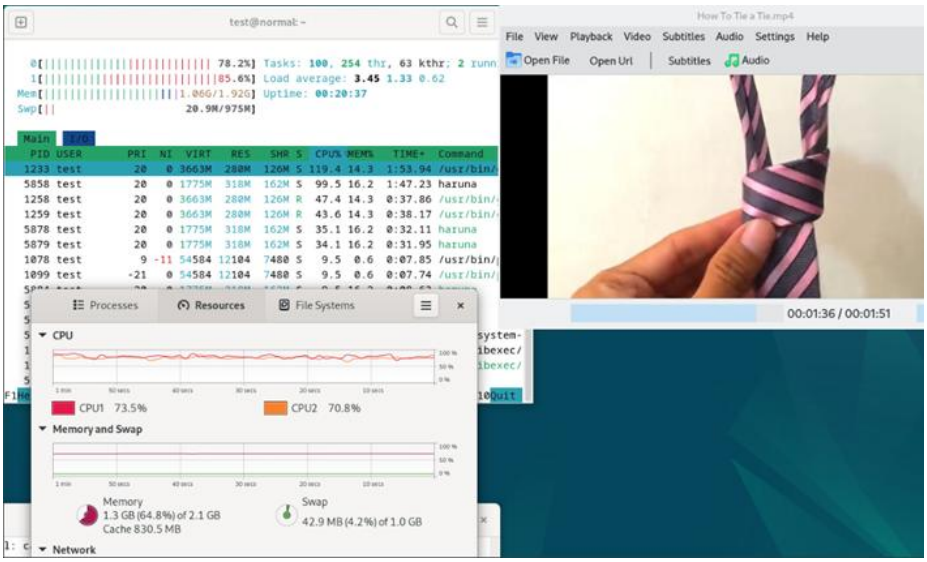
Pada kernel normal tidak ada *priority* RT yang muncul pada kolom `PRI` terminal htop saat menjalankan aplikasi VLC. Hal ini berarti bahwa aplikasi berjalan dengan prioritas biasa seperti proses lain. Saat menjalankan aplikasi, *load average* sistem mencapai 1.77 pada interval 1 menit, 0.97 pada interval 5 menit, dan 0.51 pada interval 15 menit, yang menunjukkan bahwa *load average* sistem kernel normal cenderung lebih rendah dibandingkan dengan kernel real-time sebelumnya. Saat aplikasi berjalan, %CPU sangat mirip antara CPU 1 dan CPU 2. Grafik menunjukkan bahwa kedua CPU bergerak dengan cara yang sama saat naik dan turun. Ini menunjukkan bahwa selama menjalankan aplikasi, kedua CPU digunakan bersama untuk menjalankan proses tersebut dan proses lainnya. Untuk penggunaan memori, selama menjalankan aplikasi terjadi penambahan yang signifikan menjadi ±1.2G setelah sebelumnya hanya 770M. Karena proses lain yang membutuhkan sumber daya selama pengerjaannya, penggunaan memori selama proses berjalan juga meningkat. Untuk membandingkan, digunakan aplikasi tambahan yaitu Haruna, untuk melihat perbedaan beban kerja antara aplikasi VLC dan aplikasi Haruna :



Gambar 4.2.4 Pengujian VLC pada Kernel Normal

Aplikasi Haruna, sama seperti aplikasi VLC sebelumnya, dijalankan dengan prioritas "RT" pada kolom PRI terminal htop, seperti yang ditunjukkan pada gambar di atas. Saat menjalankan aplikasi dan proses lainnya, load average sistem mencapai 4.20 pada interval satu menit, 3.80 pada interval lima menit, dan 3.39 pada interval lima belas

menit. Ini, seperti sebelumnya, menunjukkan bahwa sistem real-time mengeksekusi tugas dengan prioritas tertinggi sementara tugas lain ditunda. Penggunaan memori aplikasi Haruna meningkat sedikit dari VLC sebelumnya menjadi 700-800M. Selain itu, dapat dilihat bahwa tingkat CPU yang berbeda terjadi selama aplikasi Haruna berjalan. Untuk CPU 1, atau Cpu0, mencapai ±90%, yang terkadang meningkat hingga 50% jika dilihat dari grafik. Untuk CPU 2, atau Cpu1, mencapai ±50%, yang terkadang meningkat hingga 600% jika dilihat dari grafik. Ini menunjukkan perbedaan dengan aplikasi VLC: %CPU saat menjalankan aplikasi Haruna lebih tinggi daripada aplikasi VLC.



Gambar 4.2.5 Pengujian VLC pada Kernel Normal

Saat menjalankan aplikasi Haruna, prioritas "RT" tidak muncul pada kolom "PRI" terminal htop pada kernel normal. Sangat mirip dengan aplikasi VLC sebelumnya. Ditunjukkan bahwa beban rata-rata sistem saat menjalankan aplikasi mencapai 3.45 pada interval satu menit, 1.33 pada interval lima menit, dan 0.62 pada interval lima belas menit. Ini menunjukkan bahwa beban rata-rata sistem pada kernel normal cenderung lebih rendah daripada kernel real-time sebelumnya. Selama menjalankan aplikasi Haruna, terjadi penambahan sedikit untuk penggunaan memori menjadi ±1,3G, sama seperti saat menjalankan VLC sebelumnya. Selama aplikasi Haruna berjalan, terlihat %CPU yang sangat mirip antara CPU 1 dan CPU 2. Dilihat dari grafik, kedua cpu memiliki pergerakan yang sama selama naik dan turun. Hal ini berarti selama menjalankan aplikasi Haruna, kedua cpu digunakan secara bersama-sama untuk mengeksekusi proses tersebut beserta proses lainnya. Karena hal ini juga penggunaa memori selama proses berjalan mengalami peningkatan yang sangat tinggi disebabkan proses lain yang membutuhkan sumber daya selama pengerjaannya.

4. KESIMPULAN

Saat mengeksekusi proses, kernel real-time memiliki performa latensi yang lebih baik dibandingkan kernel normal. Ini dapat dilihat selama proses pengujian, di mana kernel real-time memiliki latensi aktual dan maksimum yang rendah dan lebih konsisten. Debian real-time mampu menangani proses dengan prioritas tertinggi secara efisien dan optimal. Selain itu, pembagian sumber daya yang dilakukan selama proses berjalan terbukti lebih diprioritaskan untuk tugas dengan prioritas tertinggi, sehingga lebih sedikit sumber daya yang digunakan dibandingkan dengan Debian biasa. Beberapa saran untuk penelitian Debian kernel real-time yang lebih lanjut: Pertama, teruskan penelitian menggunakan parameter tambahan yang dapat dioptimalkan untuk mengurangi latensi, sehingga debian real-time ini dapat lebih optimal. Kedua, gunakan alat pengujian tambahan selain cyclictest dan htop untuk mendapatkan gambaran yang lebih detail tentang penyebab latensi dan kinerja kernel real-time. Ketiga, lakukan pengujian dengan menggunakan alat-alat yang ada pada dunia nyata agar dilihat hasil kinerjanya dalam situasi yang lebih dinamis. Ini akan membantu dalam mengidentifikasi manfaat yang diperoleh selama proses modifikasi dan optimasi.

REFERENCE

Mubarok, F. (2017). *Sistem waktu nyata (real time system)*. Retrieved June 1, 2024, from https://fauzania5.blogspot.com/2017/11/sistem-waktu-nyata-real-time-system.html?m=1

BasuMallick, C. (2024). *What is a real-time operating system (RTOS)?* Retrieved June 3, 2024, from https://www.spiceworks.com/tech/hardware/articles/what-is-rtos/amp/#_004

Hasan, M. (2024). *How to install real time kernel patch on Linux (Ubuntu 16.04)*. Retrieved June 14, 2024, from https://hmenn.github.io/pages/UbuntuRT_patch.html

Binus.ac.id. (2022, November 11). *Real-time operating system*. Retrieved June 3, 2024, from https://binus.ac.id/bandung/2022/11/real-time-operating-system/

Dahlke, P. (2018). *Realtime Linux*. Retrieved June 15, 2024, from https://medium.com/@patdhlk/realtime-linux-e97628b51d5d

Barbieri, E. (2023). *Tuning a real-time kernel*. Retrieved June 15, 2024, from https://ubuntu.com/blog/real-time-kernel-tuning

Linux Foundation. (2024). *Cyclictest*. Retrieved June 16, 2024, from https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/cyclictest/start