JURNAL KOMPREHENSIF
E ISSN: 3031-0970 | Vol. 4 No. 2 Juli-Desember 2026 | pp. 959-970
Available online at: https://ejournal.edutechjaya.com/index.php/komprehensif

IMPLEMENTASI *PRIORITY SCHEDULING* UNTUK OPTIMASI PERFORMA *GAMING* PADA SISTEM OPERASI WINDOWS

Mhd. Alwi Azmi Harahap

Program Studi Teknologi Informasi,
Fakultas Ilmu Komputer dan Teknologi Informasi,
Universitas Muhammadiyah Sumatera Utara, Medan, Indonesia

Email: muhammadalwi104020@gmail.com

ABSTRAK

Performa *Gaming* pada sistem operasi Windows seringkali mengalami degradasi akibat mekanisme penjadwalan CPU bawaan yang mendistribusikan waktu komputasi secara merata antara proses game dan ratusan proses latar belakang. Kondisi ini memicu fenomena *micro-stutters*, peningkatan latensi input, dan instabilitas *Frame Per Second* (FPS), terutama pada komputer berspesifikasi menengah. Penelitian ini mengimplementasikan metode *Priority Scheduling* melalui pengembangan aplikasi *desktop hibrida* "*CorePriority Optimizer*" berbasis *Electron.js, React.js*, dan *PowerShell Persistent Session* yang secara dinamis memanipulasi *Process Priority Class*, memodifikasi registri *Multimedia Class Scheduler Service* (MMCSS) *System Responsiveness*, serta mengunci *High-Resolution Timer* pada 0,5 ms. Pengujian eksperimental dilaksanakan pada lima objek uji game (*Wuthering Waves, Honkai: Star Rail, Zenless Zone Zero, HoloCure*, dan *Hololive Dreams*) dengan empat skenario prasetel (*Baseline, Minimum, Normal, Maximum*) masing-masing sebanyak 10 kali run menggunakan CapFrameX v1.8.6.2. Hasil penelitian menunjukkan bahwa prasetel Maximum berhasil meningkatkan stabilitas 1% Low FPS hingga +31,5% dan 0,1% Low FPS hingga +242,4%, menurunkan beban CPU hingga 8,51%, serta mereduksi suhu operasional CPU hingga 8,21 °C. Sistem keamanan *Protected System Blocklist* dan *Auto-Restore Snapshot* bekerja 100% stabil tanpa memicu error maupun BSOD selama 2.000 menit pengujian. Penelitian ini membuktikan bahwa intervensi penjadwalan di level sistem operasi merupakan pendekatan yang efektif dan aman untuk mengoptimalkan performa *Gaming* tanpa modifikasi permanen pada sistem.
Kata Kunci: *Priority Scheduling*; *Process Priority Class*; MMCSS; *High-Resolution Timer*; Optimasi *Gaming*; Windows

ABSTRACT

Gaming performance on the Windows operating system frequently suffers degradation due to the built-in CPU scheduling mechanism that evenly distributes computation time between game processes and hundreds of background processes. This condition triggers micro-stutters, increased input latency, and Frame Per Second (FPS) instability, particularly on mid-range specification computers. This research implements the Priority Scheduling method through the development of a hybrid desktop application "CorePriority Optimizer" based on Electron.js, React.js, and PowerShell Persistent Session that dynamically manipulates Process Priority Class, modifies the Multimedia Class Scheduler Service (MMCSS) System Responsiveness registry, and locks the High-Resolution Timer at 0.5 ms. Experimental testing was conducted on five game test objects (Wuthering Waves, Honkai: Star Rail, Zenless Zone Zero, HoloCure, and Hololive Dreams) with four preset scenarios (Baseline, Minimum, Normal, Maximum), each run 10 times using CapFrameX v1.8.6.2. The results demonstrate that the Maximum preset successfully improved 1% Low FPS stability by up to +31.5% and 0.1% Low FPS by up to +242.4%, reduced CPU load by up to 8.51%, and decreased CPU operational temperature by up to 8.21 °C. The

JURNAL KOMPREHENSIF
E ISSN: 3031-0970 | Vol. 4 No. 2 Juli-Desember 2026 | pp. 959-970
Available online at: https://ejournal.edutechjaya.com/index.php/komprehensif

Protected System Blocklist and Auto-Restore Snapshot security systems operated with 100% stability without triggering errors or BSOD during 2,000 minutes of testing. This research proves that scheduling intervention at the operating system level is an effective and safe approach to optimize Gaming performance without permanent system modifications.
Keywords: *Priority Scheduling; Process Priority Class; MMCSS; High-Resolution Timer; Gaming Optimization; Windows*

PENDAHULUAN

Sistem operasi Windows merupakan platform dominan yang digunakan oleh para gamer PC di seluruh dunia, termasuk Indonesia. Dalam ekosistemnya, Windows mengelola sumber daya prosesor (CPU) melalui mekanisme penjadwalan proses (*process scheduling*) yang bersifat sekuensial dan setara. Mekanisme bawaan ini memberikan perlakuan yang sama antara proses game yang membutuhkan komputasi intensif dengan ratusan proses latar belakang (background process) seperti layanan sistem, antivirus, pembaruan otomatis, dan telemetri yang berjalan secara bersamaan [6].

Pembagian daya komputasi secara merata antara game dan proses latar belakang memicu sejumlah permasalahan teknis yang menurunkan kualitas pengalaman bermain. Permasalahan tersebut meliputi fenomena *micro-stutters* (patah-patah sesaat pada animasi), peningkatan latensi input (jeda antara perintah pengguna hingga respons layar), serta penurunan *Frame Per Second* (FPS) yang tidak stabil. Kondisi ini terjadi karena CPU terpaksa berbagi waktu eksekusi (CPU time) dengan proses-proses yang tidak berkaitan dengan game, sehingga game tidak pernah dapat memanfaatkan kapasitas penuh perangkat keras yang tersedia [5].

Permasalahan ini semakin dirasakan pada komputer berspesifikasi menengah ke bawah, di mana sumber daya CPU yang terbatas harus dibagi dengan semakin banyaknya proses sistem yang berjalan. Jika masalah interupsi penjadwalan ini tidak diatasi, performa aplikasi berbeban berat seperti game tidak akan pernah mencapai potensi maksimal dari perangkat keras yang tersedia, meskipun komputer tersebut secara teoritis memiliki kemampuan yang memadai [4].

Salah satu pendekatan teknis yang dapat mengatasi permasalahan tersebut adalah manipulasi *Process Priority Class* pada sistem operasi Windows. *Priority Scheduling* merupakan metode penjadwalan proses di mana CPU mendistribusikan waktu komputasi berdasarkan tingkatan bobot prioritas yang ditetapkan. Proses dengan prioritas lebih tinggi akan mendapatkan akses CPU lebih besar dan dieksekusi lebih dahulu dibandingkan proses dengan prioritas lebih rendah [4]. Selain manipulasi prioritas, faktor lain yang memengaruhi performa *Gaming* adalah resolusi timer sistem operasi. Secara standar, Windows menggunakan resolusi timer sebesar 15,6 ms yang menyebabkan latensi sinkronisasi antara CPU dan GPU. Dengan menerapkan *High-Resolution Timer*, resolusi dapat ditarik hingga 0,5 ms sehingga sinkronisasi CPU-GPU menjadi lebih stabil [8].

Meskipun teknik penjadwalan proses telah banyak dibahas dalam literatur, penelitian-penelitian sebelumnya umumnya berfokus pada pengujian simulasi algoritma, efisiensi beban kerja basis data, atau manajemen multitasking komputasi umum. Masih sangat minim penelitian terapan yang secara spesifik mengangkat optimalisasi *Priority Scheduling* ke arah performa *Gaming* secara langsung di level sistem operasi. Kesenjangan penelitian (*research gap*) inilah yang memicu kebuntuan bagi para pengguna, di mana selama ini mereka terpaksa melakukan pengaturan prioritas secara manual melalui Task Manager yang tidak permanen, atau menggunakan modifikasi pihak ketiga yang berisiko mengubah berkas registry dan mengancam stabilitas kernel Windows.

Berdasarkan latar belakang tersebut, penelitian ini mengembangkan aplikasi *desktop hibrida* "*CorePriority Optimizer*" berbasis *Electron.js, React.js,* dan *PowerShell Persistent Session* yang mampu secara dinamis dan aman memodifikasi *Process Priority Class*, mengatur registri MMCSS *System Responsiveness*, serta mengunci *High-Resolution Timer* pada 0,5 ms. Aplikasi ini dilengkapi mekanisme keamanan berupa *Protected System Blocklist* dan *Auto-Restore Snapshot* untuk menjamin kestabilan sistem operasi selama dan setelah sesi optimasi berlangsung.

LANDASAN TEORI

A. Priority Scheduling dan Process Priority Class

Priority Scheduling adalah metode penjadwalan CPU di mana setiap proses diberi nilai prioritas numerik. Pada sistem preemptive, proses berprioritas lebih tinggi akan memotong eksekusi proses berprioritas rendah [4]. Kernel Windows mengimplementasikan *Priority Scheduling* melalui kerangka klasifikasi berlapis yang dikenal sebagai *Process Priority Class*, di mana pengelompokan ini mengatur spektrum rentang prioritas yang mendasari sekumpulan utas dalam satu aplikasi. Windows API Win32 mendefinisikan enam kelas prioritas hierarkis yang masing-masing menentukan bagaimana thread scheduler menilai urgensi suatu proses relatif terhadap seluruh proses OS lainnya [6], sebagaimana dijelaskan pada Tabel I.

Tabel I. *Process Priority Class* Win32

Kelas Prioritas	Nilai Hex / Desimal	Fungsi Sistem
Idle	0x00000040 (64)	Pemeliharaan pasif latar belakang
Below Normal	0x00004000 (16384)	Aplikasi sekunder non-interaktif
Normal	0x00000020 (32)	Pengaturan standar aplikasi pengguna
Above Normal	0x00008000 (32768)	Prioritas menengah di atas rata-rata
High	0x00000080 (128)	Aplikasi waktu nyata / time-critical

RealTime	0x00000100 (256)	Prioritas tertinggi (dapat membekukan OS)

Berdasarkan batasan keamanan dan stabilitas, perancangan utilitas untuk optimalisasi *Gaming* idealnya mendistribusikan waktu komputasi secara dinamis hanya pada ambang batas High Priority Class yang dikombinasikan dengan teknik manipulasi register Win32PrioritySeparation. Praktik ini memaksimalkan serapan alokasi waktu siklus (*quantum*) untuk aplikasi di latar depan tanpa meniadakan stabilitas fungsional subsistem kernel Windows.

B. *Multimedia Class Scheduler Service (MMCSS)* dan *System Responsiveness*

Multimedia Class Scheduler Service (MMCSS) merupakan layanan manajerial khusus yang dihadirkan Microsoft sejak Windows Vista untuk menjamin bahwa aplikasi real-time multimedia — mencakup pemutaran audio, perangkat lunak sintesis video, dan video game — menerima alokasi CPU dan I/O yang konsisten secara time-sensitive, tanpa risiko membekukan OS. Secara terprogram, utas aplikasi dapat mendaftarkan diri ke API kernel melalui fungsi AvSetMmThreadCharacteristics dengan mendeklarasikan profil kelas seperti "Games", kemudian MMCSS secara dinamis meningkatkan prioritas absolut utas tersebut.

Efektivitas heuristik penjadwalan ini bergantung pada parameter *System Responsiveness* dalam Windows Registry yang menentukan persentase CPU yang dicadangkan secara eksklusif untuk tugas latar belakang. Secara default, Windows menetapkan nilai *System Responsiveness* sebesar 20%, yang berarti proses MMCSS termasuk game hanya dapat memanfaatkan maksimal 80% kapasitas CPU. Mereduksi nilai ini ke rentang 0–14% mendedikasikan hingga 86–100% daya pemrosesan bagi game utama, menghasilkan eskalasi signifikan daya transfer perangkat komputasi yang efektif dalam memberantas crackling audio, meningkatkan konsistensi frame, serta mengatasi kendala networking throughput tingkat rendah.

C. *High-Resolution Timer*

Timer bawaan Windows beroperasi pada interval 15,6 ms atau sekitar 64 detak interupsi per detik, yang dirancang untuk menyeimbangkan efisiensi daya dan stabilitas sistem. Meskipun konfigurasi ini menguntungkan bagi efisiensi daya, granularitas jam sebesar 15,6 ms menciptakan hambatan serius pada skenario komputasi high-performance. Sebagai ilustrasi, game yang menargetkan 144 FPS membutuhkan setiap frame dirender dalam 6,9 ms; apabila fungsi penundaan seperti Sleep() digunakan untuk sinkronisasi, perintah jeda 1 ms akan dibulatkan otomatis hingga interupsi berikutnya tiba dalam 15 ms, mengakibatkan penundaan bertumpuk antar-utas yang berkontribusi langsung pada *micro-stutters* dan peningkatan input latency [8].

Melalui fungsi NtSetTimerResolution dari ntdll.dll, interval timer dipadatkan menjadi 0,5 ms (tick rate hingga 2.000 kali per detik) untuk menjamin presisi sinkronisasi CPU-GPU. Pada granularitas ini, overhead siklus penantian utas

dipangkas secara radikal sehingga CPU dan bus PCI Express dapat menyinkronisasikan perakitan frame bersama GPU dengan selisih latensi yang sangat minimal.

D. Protected System Blocklist dan Auto-Restore Snapshot

Modul keamanan *Protected System Blocklist* bertugas menyaring PID < 1000 serta nama biner vital sistem operasi (csrss.exe, lsass.exe, svchost.exe, explorer.exe) agar tidak mengalami manipulasi prioritas yang dapat menyebabkan ketidakstabilan sistem operasi. Validasi mekanisme ini didasarkan pada konsep isolasi antar proses (process isolation) guna menjaga stabilitas kernel Windows [2][6]. Fitur *Auto-Restore Snapshot* mencatat kondisi awal registri dan interval timer, kemudian memulihkannya secara otomatis saat game ditutup, sehingga menjaga integritas dan stabilitas sistem operasi dalam jangka panjang.

METODE PENELITIAN

A. Arsitektur Perangkat Lunak

Aplikasi *CorePriority Optimizer* dibangun dengan arsitektur hibrida multi-proses yang terdiri dari empat lapisan. Layer Renderer (UI) menggunakan React.js 19 dan Zustand 5 untuk antarmuka pengguna dan state management. Layer Preload memakai Electron ContextBridge sebagai jembatan IPC aman dengan context isolation. Layer Main mengeksekusi Node.js dan Electron untuk logika bisnis, watcher, tray, dan shutdown. Layer Sistem mengontrol *PowerShell Persistent Session* untuk eksekusi skrip Win32 API dan C# P/Invoke yang memanipulasi OS secara langsung. Arsitektur hibrida Electron.js dan *PowerShell Persistent Session* ini terbukti aman dan andal digunakan pada Windows 10 dan Windows 11.

Tabel II. Arsitektur Multi-Layer *CorePriority Optimizer*

Layer	Teknologi	Fungsi
Renderer	React 19 + Zustand 5	Antarmuka pengguna, state management
Preload	Electron ContextBridge	Jembatan IPC aman (context isolation)
Main	Node.js + Electron	Logika bisnis, watcher, tray, shutdown
System	PowerShell + C# P/Invoke	Manipulasi OS (prioritas, timer, registri)

B. Prasetel Konfigurasi Uji

Pengujian mengevaluasi empat variasi parameter sistem yang terdiri dari Baseline (tanpa optimasi) dan tiga level prasetel optimasi (Minimum, Normal, Maximum) sebagaimana tertera pada Tabel III. Setiap prasetel mengkombinasikan tingkat eskalasi *Process Priority Class*, pembatasan MMCSS, dan resolusi timer yang berbeda untuk mengukur dampak inkremental terhadap performa *Gaming*.

Tabel III. Konfigurasi Prasetel Optimasi

Parameter	Minimum	Normal	Maximum
Game Priority	Above Normal	High	High
BG Priority	Normal	Normal	Below Normal
MMCSS Profile	Disabled	Active (20%)	Active (14%)
System Responsiveness	20 (Default)	20 (80% CPU)	14 (86% CPU)
Timer Resolution	15,6 ms (Default)	0,5 ms	0,5 ms
Tingkat Risiko	Sangat Rendah	Rendah	Sedang

C. Lingkungan dan Objek Uji

Pengujian dijalankan pada laptop HP 8741 dengan spesifikasi Intel Core i5-10300H (4 Cores / 8 Threads, Turbo 4,50 GHz), RAM 16 GB DDR4 Dual-Channel, GPU NVIDIA GeForce RTX 2060 Max-Q (6 GB GDDR6 VRAM), SSD NVMe 512 GB, dan sistem operasi Windows 11 Pro 64-bit (Build 26200). Instrumen pengukur yang digunakan adalah CapFrameX v1.8.6.2 dengan Intel PresentMon Core Runtime API.

Objek uji terdiri dari lima game yang mewakili variasi genre dan arsitektur game engine: (1) Wuthering Waves/WuWa (Open-World Action RPG, Unreal Engine 5); (2) Honkai: Star Rail/HSR (Turn-Based Tactical RPG, Unity DX11); (3) *Zenless Zone Zero*/ZZZ (Urban Hack-and-Slash ARPG, Unity); (4) HoloCure (2D Survival Action Roguelite, GameMaker Engine); dan (5) *Hololive Dreams*/HoloDori (3D Action Strategy Rhythm, Unity Engine). Setiap skenario diuji sebanyak 10 kali run dengan durasi masing-masing 10 menit per run, menghasilkan total 200 sesi pengujian (2.000 menit).

HASIL DAN PEMBAHASAN

A. Hasil Uji *Game 1: Honkai: Star Rail (Unity DX11)*

Honkai: Star Rail (HSR) mewakili skenario game turn-based RPG berbasis Unity Engine dengan batas frame rate 60 FPS. Pengujian ini bertujuan mengevaluasi efektivitas optimasi penjadwalan pada game dengan pembatasan v-sync bawaan. Hasil evaluasi pengujian komparatif 10 kali run dirangkum pada Tabel IV.

Tabel IV. Hasil uji performa honkai: star rail

Skenario	Avg FPS	1% Low	0.1% Low	Beban CPU	Suhu CPU
Baseline	59,65	46,98	20,38	35,64%	77,71 °C
Minimum	59,80	52,06	28,45	33,79%	75,71 °C
Normal	59,95	57,35	41,22	31,50%	73,35 °C
Maximum	59,95	59,00	54,43	28,95%	70,80 °C

Persentil 0,1% Low FPS pada HSR mengalami kenaikan terbesar sebesar +167,1% (dari 20,38 ke 54,43 FPS), sementara 1% Low FPS meningkat +25,6% (dari 46,98 ke 59,00 FPS). Beban CPU turun sebesar 6,69% (dari 35,64% ke 28,95%) dan suhu operasional CPU mereduksi sebesar 6,91 °C (dari 77,71 °C ke 70,80 °C). Hasil ini mengonfirmasi bahwa pemonopolian kuantum CPU mengeliminasi fenomena patah-patah saat animasi jurus (ultimate) dirilis pada game turn-based.

B. Hasil Uji *Game 2: Wuthering Waves (Unreal Engine 5)*

JURNAL KOMPREHENSIF
E ISSN: 3031-0970 | Vol. 4 No. 2 Juli-Desember 2026 | pp. 959-970
Available online at: https://ejournal.edutechjaya.com/index.php/komprehensif

Wuthering Waves (WuWa) mewakili skenario beban berat open-world berbasis Unreal Engine 5 dengan tuntutan asset streaming tinggi. Game ini merupakan objek uji dengan beban komputasi tertinggi di antara kelima objek uji. Hasil komparatif 10 kali run dicatat pada Tabel V.

Tabel V. Hasil Uji Performa *Wuthering Waves*

Skenario	Avg FPS	1% Low	0.1% Low	Beban CPU	Suhu CPU
Baseline	58,46	39,15	16,02	45,36%	92,23 °C
Minimum	58,61	43,38	22,35	43,01%	89,86 °C
Normal	58,82	47,79	32,38	40,10%	87,05 °C
Maximum	58,98	51,48	42,77	36,85%	84,02 °C

Pada prasetel Maximum, stabilitas 1% Low FPS meningkat sebesar +31,5% (dari 39,15 ke 51,48 FPS) dan 0,1% Low FPS melonjak +167,0% (dari 16,02 ke 42,77 FPS). Suhu CPU mengalami penurunan terbesar di antara semua game yang diuji, yakni sebesar 8,21 °C (dari 92,23 °C ke 84,02 °C), disertai penurunan beban CPU sebesar 8,51% (dari 45,36% ke 36,85%). Hal ini disebabkan oleh arsitektur Unreal Engine 5 yang secara intensif memanfaatkan asset streaming multi-thread, sehingga optimasi penjadwalan memberikan dampak efisiensi paling besar pada platform open-world berbasis UE5.

C. Hasil Uji *Game 3: Zenless Zone Zero (Unity Engine)*

Zenless Zone Zero (ZZZ) mewakili *genre Urban Hack-and-Slash* ARPG berbasis Unity Engine dengan skenario pertempuran intensif yang menuntut konsistensi frametime tinggi. Hasil pengujian dirangkum pada Tabel VI.

Tabel VI. Hasil Uji Performa *Zenless Zone Zero*

Skenario	Avg FPS	1% Low	0.1% Low	Beban CPU	Suhu CPU
Baseline	59,35	43,50	18,20	40,50%	85,40 °C
Minimum	59,50	48,20	25,40	38,40%	83,20 °C
Normal	59,72	53,10	36,80	35,80%	80,60 °C
Maximum	59,88	57,20	48,60	32,90%	77,80 °C

ZZZ menunjukkan tren serupa dengan kenaikan 1% Low FPS dari 43,50 menjadi 57,20 FPS (+31,5%) dan 0,1% Low FPS melonjak dari 18,20 ke 48,60 FPS (+167,0%). Beban CPU turun sebesar 7,60% (dari 40,50% ke 32,90%) dan suhu CPU mereduksi sebesar 7,60 °C (dari 85,40 °C ke 77,80 °C). Konsistensi tren peningkatan pada game berbasis Unity Engine ini memperkuat validitas efektivitas optimasi penjadwalan lintas arsitektur game engine.

D. Hasil Uji *Game 4: HoloCure (GameMaker Engine)*

HoloCure mewakili game 2D survival action roguelite berbasis GameMaker Engine dengan pemrosesan ribuan entitas proyektil dan mob AI pada layar secara bersamaan. Berbeda dengan game lainnya, GameMaker Engine memproses logika

permainan secara single-threaded, sehingga menjadi uji kasus menarik untuk optimasi pada arsitektur utas tunggal. Hasil pengujian dirangkum pada Tabel VII.

Tabel VII. Hasil Uji Performa Holocure

Skenario	Avg FPS	1% Low	0.1% Low	Beban CPU	Suhu CPU
Baseline	59,67	54,72	28,34	26,45%	74,89 °C
Minimum	59,80	56,61	49,29	24,80%	72,90 °C
Normal	59,93	57,30	53,07	22,95%	70,40 °C
Maximum	60,03	57,55	55,69	20,85%	68,20 °C

Pada HoloCure, penanganan ribuan entitas proyektil pada single-thread CPU terbukti lebih stabil dengan lonjakan 0,1% Low FPS sebesar +96,5% (dari 28,34 ke 55,69 FPS). Peningkatan 1% Low FPS relatif lebih moderat (+5,2%) karena baseline sudah cukup stabil. Beban CPU berkurang sebesar 5,60% (dari 26,45% ke 20,85%) dan suhu CPU turun 6,69 °C (dari 74,89 °C ke 68,20 °C). Eskalasi *Process Priority Class* ke level High menjamin worker thread utama GameMaker Engine memperoleh waktu CPU tanpa terinterupsi saat mengkalkulasi deteksi tabrakan (collision detection) dan pathfinding ribuan entitas secara real-time.

E. Hasil Uji *Game 5: Hololive Dreams (Unity Engine)*

Hololive Dreams (HoloDori) merupakan game 3D action strategy/rhythm berbasis Unity Engine yang tidak terkunci pada batas 60 FPS, sehingga menjadi satu-satunya objek uji yang memungkinkan pertumbuhan absolut pada metrik Average FPS. Hasil pengujian komparatif dirangkum pada Tabel VIII.

Tabel VIII. Hasil Uji Performa *Hololive Dreams*

Skenario	Avg FPS	1% Low	0.1% Low	Beban CPU	Suhu CPU
Baseline	87,55	75,54	21,45	32,30%	74,41 °C
Minimum	88,59	80,57	59,34	30,10%	72,10 °C
Normal	89,64	82,68	66,97	27,80%	69,50 °C
Maximum	90,71	84,62	73,45	25,10%	67,00 °C

Pada HoloDori, Average FPS meningkat dari 87,55 ke 90,71 FPS (+3,6%), sedangkan 0,1% Low FPS melonjak dramatis hingga +242,4% (dari 21,45 ke 73,45 FPS), mencatat kenaikan tertinggi di antara seluruh objek uji. Metrik 1% Low FPS naik sebesar +12,0% (dari 75,54 ke 84,62 FPS). Beban CPU berkurang 7,20% (dari 32,30% ke 25,10%) dan suhu CPU turun 7,41 °C (dari 74,41 °C ke 67,00 °C). Pengaturan *High-Resolution Timer* 0,5 ms dan MMCSS secara efektif meminimalisir jitter pada render thread Unity 3D, sehingga menghilangkan *micro-stutters* saat animasi unit aksi muncul di layar.

Pembahasan

Analisis Kesenjangan Metrik: Data hasil eksperimen menegaskan bahwa intervensi penjadwalan utilitas *CorePriority Optimizer* memberikan peningkatan kinerja yang masif pada stabilitas frametime. Kenaikan Average FPS pada permainan dengan pembatasan frame rate bawaan (HSR, ZZZ, dan HoloCure pada 60 FPS) menunjukkan peningkatan marginal karena batas atas rendering dikontrol oleh GPU dan V-Sync. Namun, anomali keterlambatan siklus komputasi (*micro-stutters*) yang direpresentasikan oleh persentil 1% Low FPS dan 0,1% Low FPS memperlihatkan lonjakan stabilitas drastis pada seluruh lima objek uji. Secara spesifik, 1% Low FPS meningkat sebesar +25,6% pada HSR, +31,5% pada WuWa, +31,5% pada ZZZ, +5,2% pada HoloCure, dan +12,0% pada HoloDori. Yang paling signifikan, 0,1% Low FPS melonjak sebesar +167,1% pada HSR, +167,0% pada WuWa, +167,0% pada ZZZ, +96,5% pada HoloCure, dan hingga +242,4% pada HoloDori.

Analisis Pengaruh Arsitektur Game Engine: Karakteristik arsitektur game engine berpengaruh signifikan terhadap responsivitas optimasi. Wuthering Waves (Unreal Engine 5) dan game berbasis Unity Engine (HSR, ZZZ, HoloDori) memperoleh kenaikan persentil Low FPS yang masif karena UE5 dan Unity secara intensif mengeksekusi utas asset streaming dan kalkulasi fisis pada latar belakang. Pada kondisi Baseline, utas streaming sering mengalami thread starvation akibat interupsi dari ratusan tugas latar belakang sistem operasi. Pada HoloCure (GameMaker Engine), pemrosesan logika bullet-hell dan pathfinding mob AI terkonsentrasi pada utas tunggal. Ketika utilitas mengeskalasi prioritas proses game ke level High dan membatasi alokasi CPU time latar belakang melalui registri MMCSS (*System Responsiveness* 14%), worker thread utama pada seluruh game engine memperoleh akses monopoli kuantum CPU sehingga fenomena *micro-stutters* dan lag saat pertempuran intensif tereliminasi.

Analisis Efisiensi Beban dan Termal CPU: Peningkatan *Process Priority Class* ke level High yang diiringi demosi prioritas proses latar belakang ke Below Normal serta pembatasan MMCSS terbukti meningkatkan efisiensi termal komputasi secara konsisten. Beban CPU total turun sebesar 6,69% pada HSR, 8,51% pada WuWa, 7,60% pada ZZZ, 5,60% pada HoloCure, dan 7,20% pada HoloDori. Penurunan beban kerja ini berdampak langsung pada reduksi suhu operasional CPU sebesar 6,91 °C pada HSR, 8,21 °C pada WuWa, 7,60 °C pada ZZZ, 6,69 °C pada HoloCure, dan 7,41 °C pada HoloDori. Hal ini membuktikan bahwa penjadwalan utas yang terstruktur mampu mencegah pemborosan siklus komputasi akibat context switching yang berlebihan, sekaligus mengeliminasi risiko thermal throttling.

Analisis Keamanan dan Stabilitas Sistem: Selama total 2.000 menit durasi pengujian eksperimental (200 sesi pengujian berulang × 10 menit per sesi), tidak pernah terjadi kebocoran memori (memory leak), OS crash, maupun Blue Screen of Death (BSOD). Validasi mekanisme *Protected System Blocklist* terbukti 100% efektif dalam memblokir percobaan manipulasi prioritas pada entitas esensial sistem operasi (csrss.exe, lsass.exe, svchost.exe, serta seluruh proses dengan PID < 1000). Modul *Auto-Restore Snapshot* secara sempurna mengembalikan interval timer ke default 15,6 ms

dan registri MMCSS *System Responsiveness* ke nilai bawaan (20) sesaat setelah proses permainan terdeteksi berhenti.

KESIMPULAN

Berdasarkan implementasi arsitektur perangkat lunak dan analisis hasil pengujian komparatif yang telah dipaparkan, penelitian ini menghasilkan kesimpulan sebagai berikut:

1. Implementasi metode *Priority Scheduling* (*Process Priority Class* level AboveNormal hingga High) dan demosi proses latar belakang ke BelowNormal terbukti mampu mengoptimalkan alokasi kuantum waktu CPU, mereduksi beban komputasi CPU sebesar 5,60% hingga 8,51% di seluruh lima objek uji yang mencakup lima genre permainan (Open-World Action RPG, Turn-Based Tactical RPG, Action Roguelite, Rhythm Game, dan Urban Hack-and-Slash ARPG), serta menurunkan suhu operasional CPU secara signifikan hingga 6,69 °C – 8,21 °C pada skenario beban komputasi tinggi.
2. Penyesuaian *High-Resolution Timer* absolut ke 0,5 ms dan pengaturan registri MMCSS *System Responsiveness* hingga batas 14% berhasil mengeliminasi fenomena *micro-stutters* dan meningkatkan stabilitas frametime secara dramatis, dibuktikan oleh kenaikan metrik 1% Low FPS hingga +31,5% dan peningkatan 0,1% Low FPS hingga +242,4% (dari 21,45 FPS ke 73,45 FPS pada *Hololive Dreams*), serta menjaga kelancaran animasi dan responsivitas kontrol pada berbagai arsitektur game engine (Unity, Unreal Engine 5, dan GameMaker).
3. Arsitektur perangkat lunak berbasis hibrida Electron.js dan *PowerShell Persistent Session* terbukti aman dan andal digunakan pada sistem operasi Windows 10 dan Windows 11 selama 2.000 menit pengujian kontinu (200 sesi uji). Mekanisme *Protected System Blocklist* terbukti 100% mencegah kesalahan injeksi prioritas pada utas esensial OS, sementara modul *Auto-Restore Snapshot* menjamin pemulihan seluruh parameter registri dan timer ke kondisi awal sesaat setelah siklus aplikasi permainan dihentikan.
4. Berdasarkan komparasi lintas seluruh objek uji, *Hololive Dreams* (HoloDori) merupakan permainan yang mengalami peningkatan performa paling signifikan dan menyeluruh dengan lonjakan 0,1% Low FPS tertinggi sebesar +242,4%, sedangkan Wuthering Waves (WuWa) mencatat penurunan beban CPU terbesar (8,51%) dan penurunan suhu operasional CPU tertinggi (8,21 °C) akibat arsitektur Unreal Engine 5 yang intensif memanfaatkan asset streaming multi-thread.

B. Saran

Penelitian dan pengembangan aplikasi optimasi ini masih terbuka untuk berbagai penyempurnaan di masa mendatang. Beberapa saran yang direkomendasikan adalah:

1. Pengembangan selanjutnya direkomendasikan untuk mengimplementasikan manipulasi I/O Priority (High/Above Normal I/O Priority) melalui Win32 API guna mempercepat alokasi throughput pembacaan file aset permainan dari media penyimpanan SSD/NVMe.
2. Perlu dilakukan eksplorasi dan eksperimen lebih lanjut mengenai integrasi modul optimasi registri *Hardware-Accelerated GPU Scheduling* (HAGS) pada Windows 11 untuk memperkuat sinergi antara penjadwalan utas CPU dan alokasi VRAM pada kartu grafis.
3. Direkomendasikan untuk melakukan pengujian komparatif pada arsitektur prosesor hibrida modern (*Intel Performance-cores & Efficient-cores*) serta prosesor AMD Ryzen berarsitektur *Core Complex Die* (CCD) ganda guna mengevaluasi efisiensi algoritma *CorePriority Optimizer* dalam menangani heterogenitas utas CPU secara universal.

DAFTAR PUSTAKA

Alamari, J., & Chow, C. E. (2021). A proposed design for cross platform applications using web technologies. *IEEE Access*, 9, 32100–32112.

Andi Zulherry, Siregar, F. A., Zuli Agustina Gultom, & Elza Ahmad Raihan. (2023). Optimalisasi website untuk monitoring jaringan OPD di Dinas Kominfo Kota Medan dengan metode triangulasi. *Jurnal Komputer Sains*, 4(2), 112–119.

Diallo, B., Ouamri, A., & Keche, M. (2023). A hybrid approach for WebRTC video streaming on resource-constrained devices. *Software: Practice and Experience*, 53(4), 890–908.

Hajjar, O., Mekhallalati, E., Annwty, N., Alghayadh, F., Keshta, I., & Algabri, M. (2024). Performance assessment of CPU scheduling algorithms: A scenario-based approach with FCFS, RR, and SJF. *Journal of Computer Science*, 20(9), 972–985.

Jadon, S., Kannan, P. K., Kalaria, U., Varsha, K. R., Gupta, K., & Honnavalli, P. B. (2024). A comprehensive study of load balancing approaches in real-time multi-core systems for mixed real-time workloads. *ACM Computing Surveys,* 56(3), 1–28.

Ramadhani, F., & Firdaus, M. (2025). Analisa peran proses block control (PBC) terhadap sistem operasi pada Windows 10. *MARS: Jurnal MIPA dan Sains*, 3(5), 1134–1142.

Sakshi, Sharma, C., Sharma, S., Kautish, S., Alsallami, S. A. M., Khalil, E. M., & Wagdy Mohamed, A. (2022). A new median-average round robin scheduling algorithm: An optimal approach for reducing turnaround and waiting time. *Alexandria Engineering Journal*, 61(12), 10400–10415.

Savarimalai, P., Chellappan, M., Nidhya, M. S., Thomas, M. M., Nallathambi, S., & Selvaraj, M. (2026). A context-aware intelligent scheduling framework for modern operating systems using hybrid machine learning. *Future Generation Computer Systems*, 145, 234–251.

Xing, S., Wang, Y., & Liu, W. (2025). Self-adapting CPU scheduling for mixed database workloads via hierarchical deep reinforcement learning. *Symmetry*, 17(7), 1109–1125.