Quantifying System-Level Optimizations for Latency and Frame-Time Stability in Real-Time Workloads

Aldrei Casalini Rigoli[1] **, Edson Luiz Padoin**[1]

[1]Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUI)
Ijuí – RS – Brazil

aldrei.rigoli@sou.unijui.edu.br, padoin@unijui.edu.br

Abstract. *This paper investigates how Windows 11 and GPU driver tuning affect latency-related behavior in a real-time gaming workload. We compare an out-of-the-box profile (OOB) and an optimized profile (OPT) in Counter-Strike 2 (CS2) using PresentMon, FrameView, VProf Lite, and LatencyMon. Across ten deterministic low-preset runs per profile, OPT increases average throughput from 272.64 to 281.61 FPS (+3.29%), but worsens frametime tails (p95: 5.520 to 6.987 ms; p99: 6.017 to 7.795 ms). A complementary preset comparison shows that graphics-load choice has a much larger effect than OS tuning: the high preset drops performance to about 132 FPS with 7.56 ms mean frametime. Overall, the results expose a clear trade-off between average throughput and worst-frame behavior and reinforce the use of tail-aware metrics in latency-sensitive workloads.*

Keywords: latency, frame time, Windows 11, GPU driver, performance analysis.

1. Introduction

Low latency and stable frame delivery are central requirements in interactive real-time applications, especially competitive games. Prior work shows that lower system latency improves first-person targeting, while frametime spikes degrade smoothness and can harm accuracy during critical actions [Spjut et al. 2021, Tokey et al. 2025]. In practice, however, users still rely on operating-system and driver "optimization" guides (service reduction, power-plan changes, and queue-related driver settings) with limited controlled evidence about their real impact.

This paper addresses a performance-engineering question: how low-level software configuration affects throughput and temporal stability in compute-intensive workloads. Although the experiments are game-based, the evaluation protocol is controlled and reproducible, with repeated runs, fixed workload traces, and distribution-aware metrics.

The objective of this paper is to quantify how common OS/driver optimizations change average performance and frametime tails under a controlled commercial workload.

2. Related Work

Latency behavior in Windows systems is shaped by scheduling and interrupt handling (ISR/DPC), presentation paths, and graphics queueing [Terhell 2014, Microsoft 2021]. PresentMon, FrameView, and LatencyMon are commonly used to observe frametime

distributions and kernel latency sources [Montgomery and Intel 2023, NVIDIA 2020, Terhell 2023]. These sources explain how to measure latency-sensitive workloads, but not whether common tuning recipes improve tail stability in practice.

Player-facing studies clarify why tail behavior matters. Lower end-to-end latency improves first-person targeting [Spjut et al. 2021], whereas event-specific frametime spikes reduce smoothness and can hurt gameplay outcomes [Tokey et al. 2025]. In contrast, this paper focuses on a systems question: under repeated deterministic runs of a commercial game, do practical OS/driver changes improve average throughput at the cost of worse frametime tails?

3. Methodology

Platform. Intel Core i7-10700K (8C/16T), NVIDIA GTX 1060 6 GB, 16 GB DDR4, SSD SATA, Windows 11 Pro 23H2 (build 26100), GeForce driver 581.80.

Workload. CS2 was executed on a deterministic benchmark map. The main analysis uses the *Low* preset (competitive-like, high-FPS), with ten repeated executions per system profile. A *High* preset execution is included as an additional heavier-load condition to contrast the magnitude of graphics-load effects.

System profiles.

- **Out-of-the-box (OOB):** balanced power plan, default services/startup, default NVIDIA global settings.
- **Optimized (OPT):** high-performance power plan, reduced background activity (e.g., Search/SysMain/telemetry-related services and startup apps), paused updates during tests, and NVIDIA global tuning (maximum performance, high-performance texture filtering, Low Latency Mode=Ultra, threaded optimization on).

Measurements. PresentMon and FrameView were captured simultaneously for each run; 2 s were trimmed at start/end before analysis. For each trimmed run, FPS is the mean rendered rate, mean frametime is the arithmetic mean of per-frame durations, and p95/p99 are the 95th/99th percentiles of that run's frametime distribution. Spike rate is the share of frames above $2\times$ that run's median frametime. Table 1 reports the arithmetic mean of these run-level summaries across the ten repetitions, rather than pooling all frames together, to avoid bias from small differences in run length. Ten repetitions balance repeatability and collection effort for this deterministic workload.

Scope. The contribution is a reproducible single-platform study of how practical OS/driver tuning changes throughput and tail stability in a latency-sensitive workload.

4. Results

Results are reported on two complementary axes: (i) system-profile effect at fixed low preset (OOB vs OPT), and (ii) graphics-preset effect at fixed optimized profile (Low vs High). Table 1 summarizes the first axis. OPT improves average throughput by 3.29% (272.64 to 281.61 FPS) and slightly reduces mean frametime (3.670 to 3.553 ms), but tail behavior degrades: p95 increases by 1.467 ms and p99 by 1.778 ms.

Table 1. Key low-preset metrics (10 runs per profile).

Profile	FPS	Mean ms	p95 ms	p99 ms	Spike rate
OOB	272.64	3.670	5.520	6.017	0.09%
OPT	281.61	3.553	6.987	7.795	11.75%

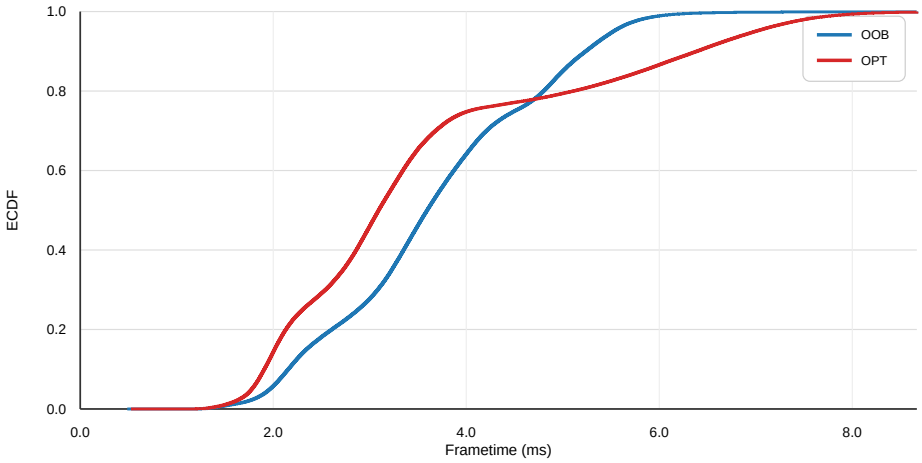

Figure 1. Pooled ECDF of PresentMon frametimes for the ten low-preset OOB and OPT runs. The x-axis is truncated at the global p99.9 to emphasize where the tails diverge.

Table 1 and Figure 1 show a consistent trade-off: OPT improves average throughput, but its frametime distribution develops a heavier upper tail and the fraction of long frames rises from 0.09% to 11.75%. Both profiles remain in a highly responsive average regime (about 3.5–3.7 ms mean frametime), so the practical difference appears in occasional longer frames associated with micro-stutter and less consistent aim correction during critical actions [Tokey et al. 2025].

LatencyMon idle measurements in both profiles remained in the expected microsecond range for ISR/DPC averages, indicating that baseline Windows 11 kernel latency was already suitable for this workload. Therefore, the observed differences are better interpreted as changes in load distribution and queueing dynamics, not as a fundamental shift in real-time capability.

These observations suggest a practical tuning order. First, graphics settings define most of the achievable performance envelope on this hardware. Second, OS/driver tuning changes are incremental and should be validated with distribution-aware indicators, because average FPS gains can coexist with heavier frametime tails.

Graphics preset choice shows a substantially larger effect in this dataset. In the optimized profile, VProf summaries collected in this campaign (Low: 33,192 frames; High: 15,041 frames) show that changing from low to high preset reduces FPS from approximately 292 to 132 and raises mean frametime from 3.43 to 7.56 ms, a much larger shift than the OOB-vs-OPT gap.

This separation matters when interpreting optimization advice: for this platform, graphics-load selection is the first-order control knob, while OS/driver tuning is a second-order adjustment that must be validated with tail-aware metrics.

5. Conclusion

Within this platform and workload, common Windows/driver optimizations provide small but consistent average-FPS gains, yet may worsen frametime tails. For competitive usage, optimization should therefore be evaluated by p95/p99 behavior, not only FPS mean. The largest performance lever in this study is graphics workload selection, not background-service tuning. A low-risk order of actions is to tune game and GPU settings first, then apply reversible system changes and reassess p95/p99 before keeping aggressive service-level modifications.

Limitations include a single hardware platform, one game, a deterministic offline benchmark path, and no direct end-to-end latency instrumentation. Future studies should replicate the analysis on newer CPU/GPU generations, multiple workload types, and direct input-to-display latency measurements.

References

Microsoft (2021). Dxgi flip model. Microsoft Learn.

Montgomery, J. and Intel (2023). Presentmon: Capture and analysis tool for frame timing and latency on windows. GitHub / Intel / Open-source tool.

NVIDIA (2020). Frameview user guide: Performance, latency and power analysis tool. NVIDIA Documentation.

Spjut, J., Boudaoud, B., and Kim, J. (2021). A case study of first person aiming at low latency for esports. NVIDIA Research.

Terhell, D. (2014). Windows and real-time.

Terhell, D. (2023). Latencymon technical overview.

Tokey, S. S., Boudaoud, B., Kim, J., Spjut, J., and Claypool, M. (2025). Timing matters: The impact of event-specific frametime spikes in first-person shooter games. In *2025 17th International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–7.