Exploratory Study of CPU–GPU Frequency Interactions and Throughput Stability in GPU-Accelerated 5G O-RAN

Abhiram Elango, Eduardo Baena, Dimitrios Koutsonikolas
Institute for Intelligent Networked Systems
Northeastern University, Boston, USA
{elango.a, e.baena, d.koutsonikolas}@northeastern.edu

***Abstract*—The virtualization of Radio Access Networks (vRAN) on GPU-accelerated platforms enables flexible 5G deployments, but introduces new challenges for power management. While Dynamic Voltage and Frequency Scaling (DVFS) has been studied for CPU-based vRAN, its effects on GPU-accelerated systems remain unexplored. We present an exploratory study of CPU and GPU frequency scaling in a containerized O-RAN testbed based on NVIDIA Aerial and OpenAirInterface. Our measurements reveal that throughput stability exhibits a non-monotonic relationship with CPU frequency, with each GPU frequency presenting a distinct optimal operating point. We validate this finding through multiple runs, identifying two sweet spots with statistical significance. We hypothesize that timing interactions between CPU scheduling, PCIe transfers, and GPU kernel execution relative to 5G slot boundaries explain this behavior. We also find that stable configurations incur a modest energy penalty, presenting an explicit stability-efficiency trade-off. These findings suggest that operators should empirically characterize frequency configurations for their specific hardware.**

***Index Terms*—5G O-RAN, vRAN, GPU acceleration, DVFS, energy efficiency, NVIDIA Aerial.**

I. Introduction

The transition of fifth generation (5G) cellular systems toward softwarized, programmable, and intelligent networks depends on successfully enabling deployments that are fully software-driven while maintaining performance comparable to traditional monolithic systems [1]. GPU-accelerated platforms, such as NVIDIA Aerial, have emerged as a promising approach to address the computational demands of Physical (PHY) layer processing, enabling cloud-native 5G deployments where baseband functions run on commodity hardware in containerized environments [2]. Recent academic deployments have demonstrated the viability of this approach, achieving cell throughputs exceeding 500 Mbps with commercial smartphones [3].

However, this architectural shift introduces new challenges for power management. Dynamic Voltage and Frequency Scaling (DVFS) is the primary mechanism for energy optimization in data centers, allowing processors to reduce power consumption during periods of low utilization. Recent work on CPU-based vRAN has demonstrated significant energy savings through intelligent frequency scaling. RENC [4] achieves a 45% reduction in CPU power consumption by exploiting the observation that cellular traffic exhibits substantial temporal variability, with 50–80% of time intervals presenting less than 1% of peak load. Similarly, Concordia [5] recovers over 70% of idle CPU cycles through fine-grained scheduling with 20μs granularity.

Despite these advances in CPU-based systems, no prior work has characterized the effects of frequency scaling on GPU-accelerated vRAN platforms. This gap is relevant because the architectural differences between CPU-based and GPU-accelerated implementations fundamentally change the DVFS dynamics. In platforms like NVIDIA Aerial, the computationally intensive L1 processing is entirely offloaded to the GPU, while the CPU handles control-plane functions and orchestrates data transfers. Furthermore, recent studies have shown that GPU DVFS transition latencies can reach hundreds of milliseconds [7], which is fundamentally incompatible with the sub-millisecond timing requirements of 5G New Radio (NR).

In this paper, we present an empirical study of CPU and GPU frequency scaling effects in a GPU-accelerated O-RAN testbed. We conduct 79 experiments across 15 frequency configurations, with comprehensive validation through 69 runs at a single GPU frequency. Our measurements reveal that throughput stability exhibits a non-monotonic relationship with CPU frequency, where each GPU frequency has a distinct optimal CPU operating point that minimizes variability. Neither “higher frequency is better” nor “lower frequency is more stable” heuristics hold. We propose a timing synchronization hypothesis to explain this behavior, suggesting that favorable alignment between CPU scheduling, PCIe transfers, GPU kernel execution, and 5G slot boundaries (0.5 ms for 30 kHz subcarrier spacing) determines stability. We also find that stable configurations incur a modest energy penalty, presenting an explicit stability-efficiency trade-off.

The rest of this paper is organized as follows. Section II discusses related work on DVFS in virtualized RAN and GPU-accelerated platforms. Section III describes our experimental methodology. Section IV presents our measurement results. Section V discusses the implications of our findings and their limitations. Section VI concludes the paper.

II. Background and Related Work

Energy efficiency in virtualized RAN has received increasing attention as operators seek to reduce operational costs while meeting sustainability targets. The RAN represents approximately 57% of total network power consumption [6], making it a primary target for optimization. This section reviews relevant work on DVFS for vRAN, GPU acceleration for PHY processing, and the timing constraints that govern real-time cellular systems.

A. DVFS in Virtualized RAN

Recent work on CPU-based vRAN demonstrates significant energy savings through DVFS. RENC [4] achieves 45% CPU power reduction by exploiting traffic variability, observing that 50–80% of time intervals present less than 1% of peak load. Concordia [5] recovers 70% of idle cycles through 20μs scheduling granularity, using quantile decision trees to predict Worst-Case Execution Time. A key insight from both works is that standard OS power management (Intel HWP) operates at 10–60 ms granularity, too coarse for sub-millisecond vRAN deadlines. Both target CPU-based L1; for GPU-accelerated platforms, DVFS dynamics differ fundamentally as CPU frequency affects control-plane and PCIe transfers while GPU frequency impacts PHY latency.

B. GPU-Accelerated vRAN Platforms

GPU acceleration for 5G PHY has matured significantly. Cavallaro et al. [11] demonstrate LDPC decoding on GPU with 87μs latency (51% reduction vs. CPU) with throughput reaching 4 Gbps. The X5G testbed [3] integrates NVIDIA Aerial with OpenAirInterface through FAPI, achieving over 500 Mbps with commercial smartphones. CloudRIC [12] demonstrates 3$\times$ energy efficiency through accelerator pooling across multiple DUs, and Nuberu [13] achieves 95% spectral efficiency while reducing resource consumption by 80%. However, none characterize frequency scaling effects on end-to-end performance.

C. Timing Constraints

CPU DVFS transitions take 10–500μs [8], while GPU DVFS can reach hundreds of milliseconds [7], incompatible with 5G NR's 0.5 ms slots (30 kHz SCS). The FAPI interface [14] requires slot indications every 125μs–1 ms, and O-RAN fronthaul [15] constrains latency below 100μs for certain split options. CPU-GPU interactions add further overhead: cudaMemcpyAsync incurs 1.2μs API latency plus 6μs driver overhead, and kernel launch latency is typically 20μs [9], [10]. While individually small, these accumulate across hundreds of kernel launches per second required for real-time PHY processing.

III. Experimental Methodology

A. Testbed Architecture

Experiments use a single node of the X5G testbed [3]: a Gigabyte E251-U70 server with NVIDIA A100 GPU running containerized OpenAirInterface (L2+) and NVIDIA Aerial cuPHY (L1) connected via FAPI [14]. The radio frontend is a Foxconn RPQN 4T4R RU (3.7–3.8 GHz, O-RAN 7.2 fronthaul). UE is a OnePlus Nord smartphone on n78 band (40 MHz, 30 kHz SCS). Core network uses Open5GS in containers.

TABLE I
CPU Core Assignment for cuPHY and L2 Adapter Threads

Thread Function	**Core(s)**	**Priority**
PHY UL/DL workers	4,5 / 6,7	Real-time
Host-to-Device copy	11	Real-time
L2 timer/message	9	Real-time
DPDK polling	10	Low
Low-priority tasks	10	Background

B. CPU Core Assignment

A critical aspect of GPU-accelerated vRAN deployment is the assignment of CPU cores to real-time processing threads. Through preliminary experiments, we determined that stable UE connectivity requires a minimum of two dedicated cores each for uplink and downlink processing. Table I summarizes the core assignment used in our experiments.

We found that reducing the downlink workers to a single core (e.g., core 6 only) causes TX aggregation failures to the GPU, as the timing and buffer handling for GPU transmission becomes insufficient. Similarly, reducing uplink workers to a single core results in Random Access (RA) processing failures, where the DU cannot process RA requests fast enough, leading to connection failures. These observations highlight the tight coupling between CPU resource allocation and GPU-accelerated PHY processing, and motivated our decision to use a fixed core assignment across all frequency scaling experiments.

C. Frequency Configurations

We study static frequency configurations, where both CPU and GPU frequencies are locked for the duration of each experiment. Dynamic frequency scaling is disabled to isolate the effects of specific operating points. GPU frequencies are set using `nvidia-smi` at three levels: 765, 1020, and 1410 MHz. CPU frequencies are controlled via `cpupower`, ranging from 2.4 to 3.2 GHz. The exploratory phase tests 5 CPU frequencies in 200 MHz increments, while the validation phase extends to 8 frequencies with 100 MHz granularity.

D. Traffic Generation and Metrics

Each configuration is evaluated using a 120-second UDP downlink transfer at 50 Mbps target bitrate, generated by iperf3. We selected this moderate bitrate rather than saturating the channel because preliminary experiments showed that higher offered loads increase sensitivity to channel errors, making it difficult to isolate the effects of frequency scaling from radio-induced variability. At 50 Mbps, the system operates well within its capacity (the theoretical maximum exceeds 500 Mbps), allowing us to focus on stability rather than peak throughput.

We record throughput at the receiver with 1-second granularity (120 samples per test), from which we derive our primary stability metric: the Coefficient of Variation (CV), defined as:

$$\mathrm{CV} = \frac{\sigma}{\mu} \times 100\% \qquad (1)$$

where σ is the standard deviation and μ is the mean throughput over the 120-second test. CV captures relative variability independent of absolute throughput values, making it suitable for comparing stability across configurations with potentially different mean rates. A CV below 5% indicates stable operation, while values above 20% represent significant instability. We also record 5th and 95th percentile throughput (P5, P95) to characterize tail behavior, packet loss rate, and jitter.

Power consumption is measured using Intel RAPL for CPU package power (specifically the `x86_pk` counter, which reports CPU package power excluding platform overhead) and `nvidia-smi power.draw` for GPU power, both sampled at 1 Hz. Energy efficiency η is computed as:

$$\eta = \frac{(P_{\text{CPU}} + P_{\text{GPU}}) \times T}{D} \quad \text{[J/MB]} \qquad (2)$$

where P_{CPU} and P_{GPU} are the average power draws in Watts, T is the test duration (120 s), and D is the total data transferred in Megabytes. This metric captures the energy cost of the compute components directly involved in PHY processing, excluding system overhead such as fans, DRAM, and platform controller.

E. Experimental Design Rationale

Our experimental methodology follows a two-phase approach: broad exploration followed by focused validation. This design reflects the practical constraints of testbed availability and the need to balance coverage with statistical rigor.

Phase 1: Exploratory Survey. We began with single runs across all 15 configurations (3 GPU frequencies $\times$ 5 CPU frequencies at 200 MHz intervals from 2.4 to 3.2 GHz). The goal was to identify which regions of the configuration space exhibited non-trivial behavior warranting deeper investigation. This phase revealed a notable finding: at GPU 1410 MHz, the configuration with CPU 2.8 GHz showed CV of 0.2%, while CPU 3.0 GHz (just 200 MHz higher) showed CV of 60%. This 300-fold difference in a single exploratory run warranted validation.

Phase 2: Focused Validation. Given this observation at GPU 1410 MHz, we concentrated validation efforts on this GPU frequency for two reasons. First, GPU 1410 MHz represents the highest performance operating point, most relevant for production deployments. Second, the high variability observed (CV ranging from 0.2% to 60%) suggested this frequency was most sensitive to CPU timing, making it a suitable candidate for understanding the underlying mechanism.

The validation proceeded in two rounds. Round 1 performed 10 runs each at CPU 2.8 GHz and 3.0 GHz to confirm whether the exploratory difference was reproducible or an artifact. Results confirmed the pattern: CPU 2.8 GHz maintained low CV (mean 3.0%) while CPU 3.0 GHz showed high variability (mean 16.5%). Round 2 extended coverage to 100 MHz granularity across the full CPU range (2.4–3.2 GHz) with 5–6 runs each, to characterize the shape of the stability curve and identify the precise location of the sweet spots.

This iterative approach (broad exploration followed by hypothesis-driven validation) allowed us to discover and confirm the sweet spot phenomenon with 69 validation runs at GPU 1410 MHz while maintaining exploratory coverage across the full configuration space with the remaining 10 runs at GPU 765 MHz and GPU 1020 MHz.

TABLE II
EXPLORATORY RESULTS (14 SINGLE RUNS, SORTED BY CV)

GPU (MHz)	CPU (GHz)	Mean (Mbps)	CV (%)	Loss (%)
1410	2.8	50.0	**0.2**	0.02
1020	3.2	49.9	3.4	0.17
1020	2.4	49.9	5.7	0.42
1410	3.2	49.3	10.1	1.42
1020	2.6	49.1	12.3	1.92
765	2.4	49.5	13.0	1.26
1020	2.8	49.2	14.3	1.90
765	3.2	49.6	18.6	0.81
765	2.8	47.8	21.8	4.51
1410	2.6	47.4	23.6	5.07
1020	3.0	48.0	25.0	4.02
765	2.6	49.2	36.8	3.20
765	3.0	49.7	40.7	1.40
1410	3.0	46.3	**59.8**	4.00

IV. RESULTS

This section presents our measurement results, beginning with exploratory observations across all configurations and proceeding to detailed validation of the frequency sweet spot phenomenon.

A. Exploratory Observations

Table II summarizes results from the exploratory phase, showing a representative subset of single-run configurations sorted by CV. The most notable observation is the large difference in throughput stability between seemingly similar configurations. At GPU 1410 MHz, the configuration with CPU at 2.8 GHz achieves very low variability with CV of 0.2%, while CPU at 3.0 GHz exhibits CV of 59.8%, a 300-fold difference. This observation motivated our comprehensive validation study.

The exploratory data reveals that the optimal CPU frequency varies depending on GPU frequency: GPU 765 MHz performs best at CPU 2.4 GHz, GPU 1020 MHz at CPU 3.2 GHz, and GPU 1410 MHz shows the largest variation with CV ranging from 0.2% to $\sim$60%. This pattern suggests that the relationship between CPU and GPU frequencies is not simply additive, but involves complex timing interactions.

B. Validation of the Sweet Spot

To validate the sweet spot phenomenon (i.e. specific CPU frequencies that yield anomalously low variability) observed at

TABLE III
GPU 1410 MHz Validation Results (69 Runs Total)

CPU (GHz)	N	Mean CV (%)	Std CV (%)	Range (%)
2.4	**5**	**0.2**	**0.0**	**0.2–0.3**
2.5	5	16.4	18.0	1–46
2.6	6	21.0	11.9	10–43
2.7	5	11.9	8.1	2–21
2.8	**16**	**4.3**	**6.2**	**0.1–19**
2.9	5	16.0	7.4	9–26
3.0	16	18.1	17.3	0.2–60
3.1	5	8.8	5.2	0–13
3.2	6	13.6	5.7	8–21

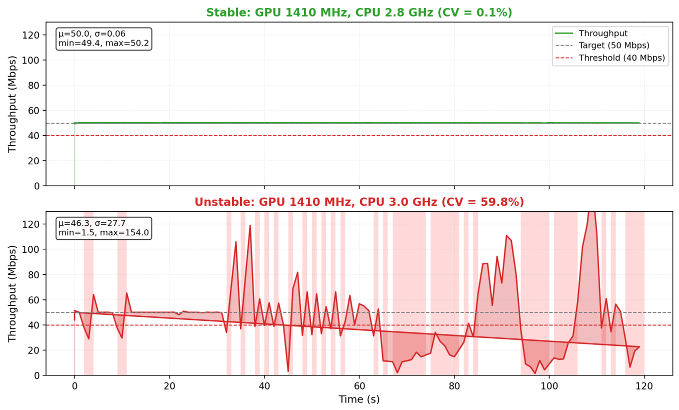

Fig. 1. Throughput time series for stable (top, **CV=0.1%**) and unstable (bottom, **CV=59.8%**) configurations. The stable case maintains near-constant throughput; the unstable case oscillates with bursts followed by degradation.

GPU 1410 MHz, we conducted 69 runs across 9 CPU frequencies. Table III presents the results, showing mean CV, standard deviation, and range for each frequency. The varying sample sizes (N) reflect our iterative methodology: we allocated more runs to the frequencies showing the most extreme behavior (2.8 and 3.0 GHz) to confirm reproducibility, while intermediate frequencies received fewer runs to characterize the shape of the stability curve.

The validation reveals two distinct sweet spots: CPU 2.4 GHz achieves the best stability (CV=0.2%, virtually no variance across 5 runs), while CPU 2.8 GHz provides the second-best performance (CV=4.3%). Between these sweet spots, CPU 2.5 GHz and 2.6 GHz exhibit poor stability (CV=16.4% and 21.0%, respectively). Similarly, frequencies above 2.8 GHz show degraded performance, with 3.0 GHz reaching CV=18.1%. A one-way ANOVA test yields F(8,60)=2.54 with p=0.018, confirming statistical significance. This pattern of multiple sweet spots separated by unstable regions suggests complex timing interactions rather than simple monotonic relationships.

The validation also reveals substantial run-to-run variability at most frequencies, with standard deviations ranging from 5.2% to 17.3%. This variability likely reflects sensitivity to initial conditions and transient system states. Notably, the sweet spots at 2.4 GHz and 2.8 GHz exhibit not only low mean CV but also low variance (Std=0.0% and 6.2%, respectively), suggesting robust operation at these frequencies.

We also analyzed the correlation between stability metrics and other system parameters. CV correlates strongly with packet loss (ρ=0.60), jitter (ρ=0.80), and the number of degradation events (ρ=0.84), confirming that throughput variability is a meaningful proxy for overall connection quality. The 5th percentile throughput correlates with mean throughput (ρ=0.81), indicating that configurations with higher average performance also tend to have better worst-case behavior.

Temporal autocorrelation analysis reveals that unstable configurations exhibit *bursty* degradation patterns: the lowest-stability configuration (GPU 1410 MHz, CPU 3.0 GHz) shows lag-1 autocorrelation of 0.58, meaning low-throughput seconds cluster together rather than occurring randomly. This has practical implications: a user experiencing degradation is likely to suffer multiple consecutive bad seconds, making the user experience worse than isolated drops of equivalent total duration.

C. Cross-GPU Frequency Analysis

An important observation from the exploratory data is that CPU 3.0 GHz consistently exhibits poor stability across all three GPU frequencies tested. As shown in Table II, GPU 1410 MHz at CPU 3.0 GHz shows the highest CV (60%), while preliminary tests at GPU 765 MHz and GPU 1020 MHz at CPU 3.0 GHz also exhibit elevated variability. We hypothesize that 3.0 GHz may create a resonance condition where CPU cycle timing interferes destructively with the 0.5 ms slot boundary, causing systematic deadline misses. This frequency-specific pathology warrants further investigation with detailed timing instrumentation. While these observations are based on single exploratory runs for GPU 765/1020 MHz and require further validation, the consistent pattern across all three GPU frequencies suggests this finding merits investigation on other platforms.

Figure 2 illustrates the temporal behavior of throughput for representative stable and unstable configurations. The stable configuration (GPU 1410 MHz, CPU 2.8 GHz) maintains consistent throughput near the target 50 Mbps throughout the 120-second test, with zero degradation events (defined as transitions below 40 Mbps). In contrast, the unstable configuration (GPU 1410 MHz, CPU 3.0 GHz) exhibits large oscillations with up to 20 degradation events and maximum consecutive degradation of 15 seconds below the 40 Mbps threshold. The 5th percentile throughput drops from 50.0 Mbps in the stable case to just 8.5 Mbps in the unstable case, representing a 6× degradation in worst-case performance despite only a 200 MHz difference in CPU frequency.

The throughput spikes exceeding 100 Mbps represent recovery bursts where the system catches up after missed deadlines. This oscillatory pattern is characteristic of timing-induced instability rather than random channel variations.

D. Worst-Case Performance

While CV captures variability, operators also need worst-case guarantees. Figure 3 shows the 5th percentile throughput

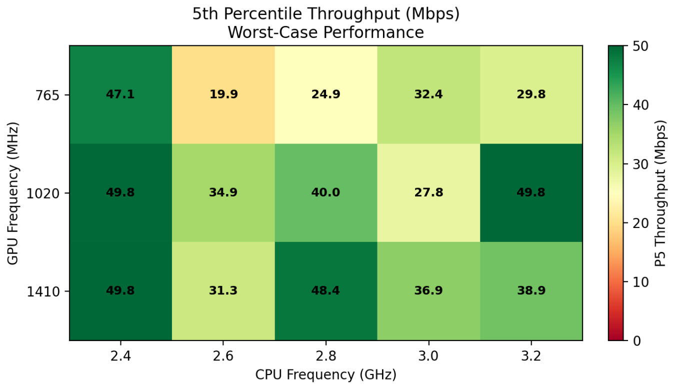

Fig. 2. 5th percentile throughput (P5) across configurations. Sweet spots achieve high P5; unstable frequencies show degraded worst-case performance.

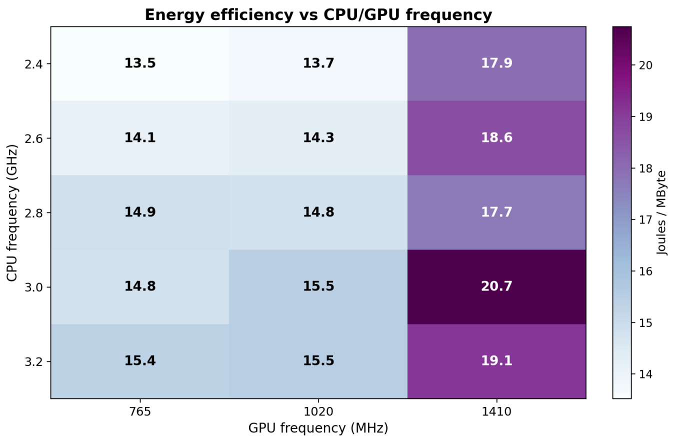

Fig. 3. Energy efficiency (J/MB) across CPU and GPU frequency configurations. Lower values indicate better efficiency. GPU 1020 MHz configurations are most efficient, while GPU 1410 MHz consumes more energy due to higher GPU power draw.

(P5), the value exceeded 95% of the time. The sweet spot configurations consistently achieve high P5 values, while unstable frequencies show significantly degraded tail performance.

E. Energy Efficiency

Figure 4 shows energy efficiency across all configurations. Energy efficiency ranges from 13.5 J/MB at GPU 765 MHz with CPU 2.4 GHz to 20.7 J/MB at GPU 1410 MHz with CPU 3.0 GHz. Lower GPU frequencies are generally more energy-efficient due to reduced GPU power draw, but the relationship with stability is not straightforward. The two sweet spot configurations at GPU 1410 MHz achieve similar energy efficiency: CPU 2.4 GHz at 17.9 J/MB and CPU 2.8 GHz at 17.7 J/MB, both representing a reasonable trade-off between energy consumption and performance reliability.

Interestingly, the most energy-efficient configurations (GPU 765 MHz and GPU 1020 MHz at low CPU frequencies) do not coincide with the most stable configurations. This presents operators with a trade-off: for example, GPU 1020 MHz at CPU 2.4 GHz achieves 13.7 J/MB but with CV of 5.7%, and GPU 765 MHz at CPU 2.4 GHz achieves 13.5 J/MB but with CV of 13.0%, while GPU 1410 MHz at CPU 2.8 GHz achieves only 17.7 J/MB but with CV of 0.2% in the best case. The 29% energy penalty for the stable configuration may be acceptable for latency-sensitive applications (e.g., industrial automation, remote surgery) where consistent QoS is paramount, while the efficient configuration may suffice for best-effort traffic.

V. Discussion

Our measurements reveal a non-intuitive relationship between frequency configuration and throughput stability in GPU-accelerated vRAN. This section discusses a hypothesis to explain the observed behavior, its implications for O-RAN deployments, and the limitations of our study.

A. Timing Synchronization Hypothesis

The non-monotonic relationship between CPU frequency and throughput stability, combined with the observation that each GPU frequency has a distinct optimal CPU frequency, suggests a timing synchronization mechanism. We hypothesize that at certain CPU frequencies, the timing of thread scheduling, context switches, and PCIe data transfers aligns favorably with GPU kernel execution and 5G slot boundaries, reducing the probability of missed deadlines.

Several observations support this hypothesis. First, the non-monotonic pattern rules out simple explanations based on computational capacity: if the issue were insufficient CPU resources, we would expect monotonic improvement with increasing frequency. Second, the GPU-specific optima suggest that different GPU kernel execution times require different CPU timing to maintain synchronization. Third, we verified that GPU temperature remained constant at 31–32°C across all frequencies, ruling out thermal throttling as a cause of instability. Fourth, while context switch rate correlates strongly with CPU frequency (ρ=0.99), it does not correlate with CV (ρ=-0.21), indicating that the mechanism is more subtle than simple scheduling overhead.

To illustrate concretely: at GPU 1410 MHz, the PHY kernels complete faster than at 765 MHz. However, faster completion does not translate to more slack; instead, the system operates with tighter timing margins because the CPU must prepare the next batch of data sooner to keep the GPU fed. If CPU-side operations (buffer preparation, FAPI message handling) take slightly longer at certain frequencies due to cache behavior or memory access patterns, they may occasionally miss the narrow window for timely GPU submission. The optimal CPU frequency is thus the one where CPU-side latency distributions best complement the GPU's execution time to consistently meet the 0.5 ms deadline. This explains why GPU 765 MHz (slower kernels, more slack) tolerates lower CPU frequencies, while GPU 1410 MHz (faster kernels, tighter margins) requires precise CPU timing.

Confirming this hypothesis would require detailed instrumentation of CPU-GPU latencies using tools such as NVIDIA Nsight Systems, capture of FAPI timing violations at the L1/L2 interface, and correlation of throughput drops with kernel scheduling events via ftrace. We leave this instrumentation to future work.

B. Implications for O-RAN Deployments

Our findings have practical implications for operators deploying GPU-accelerated vRAN. Standard OS power management policies may inadvertently select suboptimal operating points; we recommend disabling dynamic CPU frequency scaling (e.g., setting the Linux governor to `performance`) and locking frequencies to empirically validated sweet spots. The GPU-specific nature of optimal CPU frequencies means each hardware combination must be characterized during commissioning, while O-RAN interfaces are standardized, optimal operating parameters remain hardware-specific. From an O-RAN RIC perspective, xApps focused on energy optimization should incorporate frequency-stability awareness to avoid inadvertently degrading service quality.

C. Limitations

Our study has several limitations. Comprehensive validation was performed only at GPU 1410 MHz; sweet spots at GPU 765/1020 MHz are based on single runs. The timing hypothesis remains unconfirmed pending Nsight Systems instrumentation. Results are from a single hardware configuration (A100, OnePlus Nord) and may differ on other platforms. We studied only static frequencies with a single UE; dynamic transitions and multi-UE scenarios remain unexplored. Radio conditions varied (RSRP -77 to -85 dBm), but we found no correlation with CV (ρ=0.12), suggesting frequency effects dominate in our controlled environment.

VI. Conclusion

This paper presents the first empirical characterization of CPU and GPU frequency scaling effects in GPU-accelerated 5G O-RAN. Through 79 experiments on a containerized testbed with NVIDIA Aerial and OpenAirInterface, we discover that throughput stability exhibits a non-monotonic relationship with CPU frequency, with multiple sweet spots at specific frequencies. For GPU 1410 MHz, we identify two optimal operating points at CPU 2.4 GHz and CPU 2.8 GHz, with statistical significance confirmed through 69 validation runs. Between and beyond these sweet spots, adjacent frequencies exhibit significantly higher variability, suggesting complex timing interactions.

We hypothesize that this phenomenon results from timing synchronization between CPU scheduling, GPU kernel execution, and 5G slot boundaries. Our findings suggest that standard DVFS policies may be unsuitable for GPU-accelerated vRAN, and that operators should empirically characterize optimal frequency configurations for their specific hardware deployments.

Future work includes: (1) Nsight Systems instrumentation to validate the timing hypothesis; (2) validation on GPU 765/1020 MHz and newer GPUs (L40S, H100); (3) ML-based frequency optimization; (4) multi-UE scenarios; and (5) O-RAN RIC integration via xApps. Scripts and data will be available upon publication.

Acknowledgment

This work was partially supported by the U.S. National Science Foundation Grant CNS-2117814.

References

[1] O-RAN Alliance, "O-RAN Architecture Description," O-RAN.WG1.O-RAN-Architecture-Description-v07.00, 2023.

[2] NVIDIA, "NVIDIA AI Aerial Software," 2024.

[3] D. Villa *et al.*, "X5G: An Open, Programmable, Multi-vendor, End-to-end, Private 5G O-RAN Testbed with NVIDIA ARC and OpenAirInterface," *IEEE Trans. Mobile Comput.*, pp. 1–18, 2025, doi: 10.1109/TMC.2025.3580764.

[4] A. Kalia, N. Lazarev, L. Xue, X. Foukas, B. Radunovic, and F. Y. Yan, "Towards Energy Efficient 5G vRAN Servers," in *Proc. 22nd USENIX Symp. Netw. Syst. Des. Implement. (NSDI)*, 2025, pp. 1205–1219.

[5] X. Foukas and B. Radunovic, "Concordia: Teaching the 5G vRAN to Share Compute," in *Proc. ACM SIGCOMM*, 2021, pp. 580–596, doi: 10.1145/3452296.3472894.

[6] A. Fehske, G. Fettweis, J. Malmodin, and G. Biczók, "The Global Footprint of Mobile Communications: The Ecological and Economic Perspective," *IEEE Commun. Mag.*, vol. 49, no. 8, pp. 55–62, 2011, doi: 10.1109/MCOM.2011.5978416.

[7] S. Mittal, "A Survey of Techniques for Improving Energy Efficiency in Embedded Computing Systems," *Int. J. Comput. Aided Eng. Technol.*, vol. 6, no. 4, pp. 440–459, 2014, doi: 10.1504/IJCAET.2014.065419.

[8] A. Mazouz, A. Laurent, B. Pradelle, and W. Jalby, "Evaluation of CPU Frequency Transition Latency," *Comput. Sci. Res. Dev.*, vol. 29, pp. 187–195, 2014, doi: 10.1007/s00450-013-0240-x.

[9] D. Lustig and M. Martonosi, "Reducing GPU Offload Latency via Fine-Grained CPU-GPU Synchronization," in *Proc. IEEE HPCA*, 2013, pp. 354–365, doi: 10.1109/HPCA.2013.6522332.

[10] NVIDIA, "Nsight Systems User Guide," 2024.

[11] C. Tarver, M. Tonnemacher, H. Chen, J. Zhang, and J. R. Cavallaro, "GPU-Based, LDPC Decoding for 5G and Beyond," *IEEE Open J. Circuits Syst.*, vol. 2, pp. 278–290, 2021, doi: 10.1109/OJCAS.2020.3042448.

[12] L. Lo Schiavo *et al.*, "CloudRIC: Open Radio Access Network (O-RAN) Virtualization with Shared Heterogeneous Computing," in *Proc. ACM MobiCom*, 2024, pp. 558–572, doi: 10.1145/3636534.3649381.

[13] G. Garcia-Aviles *et al.*, "Nuberu: Reliable RAN Virtualization in Shared Platforms," in *Proc. ACM MobiCom*, 2021, pp. 749–761, doi: 10.1145/3447993.3483266.

[14] Small Cell Forum, "5G FAPI: PHY API Specification," Tech. Rep. 222.10.04, 10th ed., 2021.

[15] O-RAN Alliance, "O-RAN Fronthaul Control, User and Synchronization Plane Specification," O-RAN.WG4.CUS.0-R003-v12.00, 2023.

[16] Z. Qi, Y. Yao, Y. Li, C.-H. Tung, J. Zheng, D. Zhuo, and T. Chen, "DecodeX: Exploring and Benchmarking of LDPC Decoding across CPU, GPU, and ASIC Platforms," *arXiv preprint*, 2024. doi: 10.48550/arXiv.2511.02952.