E-dLLM: Energy-Efficient Diffusion LLM Inference via Phase-Aware DVFS

Mona Moghadampanah, Xiangchen Li, Farhana Amin, and Dimitrios S. Nikolopoulos
Department of Computer Science, Virginia Tech
Blacksburg, VA, USA
{monamp, lixiangchen, afarhana, dsn}@vt.edu

***Abstract*—Existing diffusion LLM (dLLM) serving systems focus on optimizing latency and throughput, while their energy efficiency remains less studied. We show that cached dLLM decoding alternates between compute-bound cache-refresh and memory-bound cache-reuse phases, while decoding configurations expose broad accuracy–energy trade-offs. We present E-dLLM, which combines energy-aware configuration selection with phase-aware dynamic voltage and frequency scaling (DVFS). On LLaDA-8B, reactive DVFS reduces energy by 17% with 0.9% latency overhead; across seven configurations, it saves 11.2–20.7% energy with at most 1.1% latency overhead, while preserving model outputs.**

***Index Terms*—diffusion large language models, energy-efficient inference, phase-aware DVFS, GPU power management, LLM serving**

I. Motivation

Energy-aware LLM serving has been explored for autoregressive models using techniques such as DVFS, frequency scaling, and power-aware control. Prior work such as DynamoLLM reduces GPU energy while meeting performance requirements [2]. However, these approaches primarily target the prefill and decode phases of autoregressive inference. In contrast, with block-wise KV caching, dLLM decoding repeatedly alternates between compute-intensive cache-refresh and lighter cache-reuse phases within a request [3, 1], creating a finer-grained opportunity for frequency control. Our profiling shows that cache refresh is compute-bound, while cache reuse is memory-bound. Yet the GPU remains near its maximum frequency during both phases, suggesting that reuse can operate at a lower frequency with little latency impact.

dLLMs also expose decoding choices, including confidence threshold, denoising steps, and block size, that affect accuracy, latency, and energy. These two opportunities motivate E-dLLM, which combines decoding-configuration selection with phase-aware GPU frequency control while satisfying accuracy and latency requirements.

II. Research Questions

We investigate how dLLM serving can reduce energy while meeting accuracy and latency requirements. We ask:

RQ1: How do decoding configurations shape the trade-offs among accuracy, energy, and latency, and can configuration selection reduce energy at a given accuracy target?

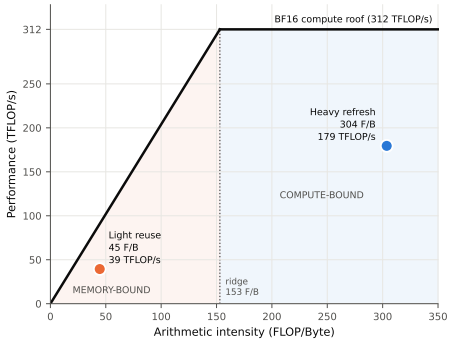


Fig. 1: Roofline characterization of cache-refresh and cache-reuse steps. Refresh is compute-bound, while reuse has lower arithmetic intensity and falls in the memory-bound region.

RQ2: Can phase-aware DVFS reduce dLLM inference energy with minimal latency overhead while preserving model outputs?

III. Experimental Setup

We evaluate LLaDA-8B-Instruct on an NVIDIA A100 80GB GPU using Fast-dLLM [3], BF16 precision, DualCache, and batch size 1. We use GSM8K with temperature 0. We vary confidence threshold $\tau \in \{0.6, 0.8, 0.9, 0.95\}$, denoising steps $\{64, 128, 256\}$, and block size $\{16, 32, 64\}$, excluding configurations with more than 5% leftover masks as incomplete generations. We measure accuracy, per-request energy using Zeus/NVML, and per-request latency; profile refresh and reuse phases with NVIDIA Nsight; and evaluate Auto, fixed-frequency, and reactive DVFS policies. Results are averaged over three runs per configuration.

IV. Characterization

Phase behavior. Cached dLLM decoding alternates between a compute-intensive cache-refresh step at the start of each block and lighter cache-reuse steps. Roofline profiling in Figure 1 shows that refresh is compute-bound at 304 FLOP/byte and 179 TFLOP/s, while reuse has lower arithmetic intensity at 45 FLOP/byte and 39 TFLOP/s. This 6.8× difference motivates phase-aware frequency control.

Configuration trade-offs. We vary the confidence threshold, denoising steps, and block size, resulting in 66.7–79.3%

TABLE I: Energy and latency changes relative to Auto frequency. The first row corresponds to the representative configuration used in Figure 4; the remaining rows evaluate DVFS across six additional decoding configurations.

	Reactive vs. Auto		Fixed 1050 MHz vs. Auto	
Configuration (Accuracy)	**Energy**	**Latency**	**Energy**	**Latency**
τ0.9 / 256 / 32 (77.3%)	−17.0%	+0.9%	−27.5%	+7.9%
τ0.6 / 256 / 64 (66.7%)	−18.9%	+1.1%	−29.9%	+8.3%
τ0.6 / 64 / 32 (72.7%)	−13.7%	+0.9%	−30.7%	+12.3%
τ0.9 / 128 / 64 (73.3%)	−20.7%	+0.7%	−26.1%	+4.6%
τ0.8 / 128 / 32 (75.0%)	−17.0%	+0.7%	−30.0%	+8.8%
τ0.9 / 128 / 32 (76.0%)	−16.1%	+0.4%	−26.3%	+7.2%
τ0.8 / 128 / 16 (79.3%)	−11.2%	+0.4%	−29.0%	+13.4%

Configuration format is τ / denoising steps / block size. Each policy is compared with the same configuration under Auto frequency. Accuracy is unchanged across frequency policies.

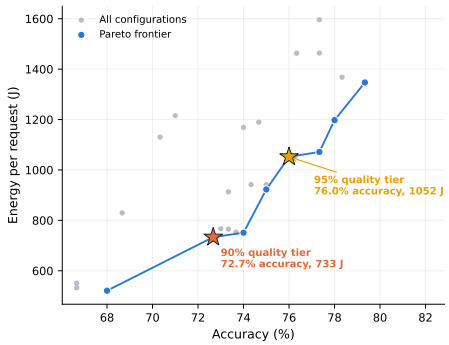


Fig. 2: Accuracy–energy trade-off across dLLM configurations on GSM8K. The Pareto frontier shows energy-efficient operating points, with markers for the 90% and 95% relative-accuracy targets.

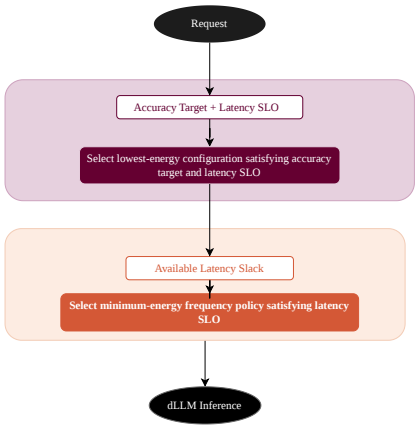


Fig. 3: E-dLLM control flow. The decoding configuration is selected to satisfy accuracy and latency requirements, followed by selection of the minimum-energy GPU frequency policy that remains within the latency SLO.

accuracy and 521–1596 J/request on GSM8K. The Pareto frontier in Figure 2 shows the trade-off between accuracy and energy across these configurations. At the 90% and 95% relative-accuracy targets, defined relative to the best observed accuracy, selecting the lowest-energy configuration that meets the target reduces energy while maintaining the required accuracy.

V. E-dLLM: Energy-Aware Control

E-dLLM policy. Figure 3 shows E-dLLM's two-stage control. It first selects the lowest-energy decoding configuration that meets the accuracy target and latency SLO. It then selects the lowest-energy GPU frequency policy that meets the latency SLO. The frequency policy may use different clocks for cache-refresh and cache-reuse phases.

Phase-aware DVFS. For a representative configuration with $\tau = 0.9$, 256 denoising steps, block size 32, DualCache, and batch size 1, Figure 4 shows that Reactive DVFS reduces energy by 17.0% with only 0.9% latency overhead. Fixed 1050 MHz reduces energy by 27.5%, but increases latency by 7.9%. Reactive DVFS lowers the reuse frequency while keeping refresh at a higher clock, reducing energy with much smaller latency overhead.

DVFS across configurations. Across seven decoding configurations, Table I shows that Reactive saves 11.2–20.7% energy with at most 1.1% latency overhead, while fixed 1050 MHz saves 26.1–30.7% with 4.6–13.4% latency overhead. These results show that DVFS provides additional energy savings across different decoding configurations.

VI. Conclusion and Ongoing Work

Our results show that dLLM decoding configurations create trade-offs among accuracy, energy, and latency, while phase-aware DVFS can reduce energy independently of the decoding configuration. Across seven configurations, Reactive DVFS reduces energy by 11–21% with at most 1.1% latency overhead, while fixed 1050 MHz provides larger energy savings when more latency slack is available. E-dLLM combines these two controls by first selecting a decoding configuration based on accuracy and latency requirements, and then selecting the lowest-energy frequency policy that meets the latency SLO. Ongoing work includes finer-grained frequency control, larger-

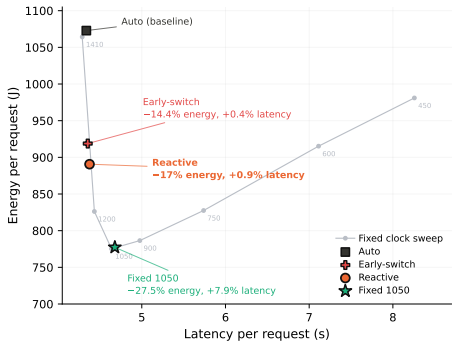


Fig. 4: Energy–latency trade-off of fixed-frequency and phase-aware DVFS policies. Reactive DVFS reduces energy by 17.0% with 0.9% latency overhead, while fixed 1050 MHz reduces energy by 27.5% with 7.9% latency overhead. The energy axis is truncated at 700 J for readability.

batch and serving evaluations, and testing additional dLLM models and GPUs.

Acknowledgements

This work was supported by NSF Grant No. 2315851, a Sony Faculty Innovation Award, an Amazon Research Award, and the Virginia Tech College of Engineering. OpenAI ChatGPT was used for language editing; the authors are responsible for all technical content, analysis, and results.

References

[1] Jiakun Fan et al. “dLLM-Serve: Bridging the Memory Gap in Diffusion Language Model Serving”. In: *Proceedings of the 40th ACM International Conference on Supercomputing*. ICS ’26. Association for Computing Machinery, 2026, pp. 515–526. ISBN: 9798400725227. DOI: 10.1145/3797905.3800532. URL: https://doi.org/10.1145/3797905.3800532.

[2] Jovan Stojkovic et al. “Dynamollm: Designing llm inference clusters for performance and energy efficiency”. In: *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE. 2025, pp. 1348–1362.

[3] Chengyue Wu et al. “Fast-dllm v2: Efficient block-diffusion llm”. In: *arXiv preprint arXiv:2509.26328* (2025).