GreenMoE: Exploiting Dynamic Load Imbalance for Energy-Efficient Long-Context MoE Training

Laiyi Li [1] **Zhenheng Tang** [2] **Peijie Dong** [3] **Qiang Wang** [1]

Abstract

Large language model training increasingly combines long-context data with mixture-of-experts (MoE) architectures, but this setting also exposes a practical energy-efficiency problem: dynamic sequence packing and expert routing create substantial load imbalance across GPUs. Many energy-saving methods rely on static execution assumptions or reduce frequency on phases that may become critical, which can harm training throughput. We present GreenMoE, a dynamic-load-aware frequency scaling framework that treats non-critical-path waiting time as an energy-saving opportunity. GreenMoE estimates the load of Attention and MoE blocks at runtime, selects GPU core frequencies that satisfy a critical-path time constraint, and applies the selected frequencies through an asynchronous CUDA-event-based controller. In representative single-node configurations, GreenMoE reduces per-GPU step energy by 8.63% with 3.75% step-time overhead in a 64K setting and by up to 10.20% with larger overhead; in a constructed Attention-heavy setting, it reduces non-critical GPU energy by 16.15% with 0.23% overhead.

1. Introduction

Long-context training and mixture-of-experts (MoE) architectures are two important ingredients in modern large language models. Transformer language models use self-attention to aggregate information across tokens, and long-context variants extend this mechanism to documents, conversations, and agent trajectories whose useful evidence may span many thousands of tokens (Vaswani et al., 2017; Dao et al., 2022; Jacobs et al., 2023; Liu et al., 2023). MoE layers increase model capacity by routing each token to a small subset of experts, allowing the total parameter count to grow without activating all parameters for every token (Shazeer et al., 2017; DeepSeek-AI et al., 2025; Yang et al., 2025). These trends improve model capability, but they also make training expensive: large-scale pretraining combines many GPUs, long execution times, and several forms of distributed parallelism (Rajbhandari et al., 2020; Shoeybi et al., 2020; DeepSeek-AI et al., 2025).

Energy efficiency is therefore a practical systems requirement rather than only a hardware concern. Dynamic voltage and frequency scaling (DVFS) can reduce GPU power, but GPU performance and power do not scale linearly with core frequency, and the energy-optimal point depends on the workload (Bridges et al., 2016; Mei et al., 2017; Tang et al., 2019; Wang & Chu, 2020). This makes large-model training difficult to optimize: lowering the frequency of a GPU on the critical path can lengthen compute time, and distributed training exposes many synchronization boundaries through gradient reductions, sequence-parallel collectives, and expert-parallel token exchange (Shoeybi et al., 2020; Jacobs et al., 2023; DeepSeek-AI et al., 2025). Prior energy-saving systems exploit slack from pipeline schedules or profiled execution graphs (Choi et al., 2023; Chung et al., 2024; Wu et al., 2026), but long-context MoE training adds a runtime source of variation that is not fixed by the static model alone.

This paper studies a complementary source of energy-saving opportunity: dynamic load imbalance in long-context MoE training. Long-context data are often mixed with shorter samples because training only on long sequences can degrade short-context behavior (Gao et al., 2024). Sequence packing reduces padding waste in such mixed-length batches, but two packed sequences with the same total length can have different Attention cost because causal Attention is computed within original samples rather than across packing boundaries (Vaswani et al., 2017; Dao et al., 2022). Expert routing creates a second imbalance: after All-to-All token exchange, different expert-parallel ranks can receive different token counts and therefore different expert-compute loads (Shazeer et al., 2017; DeepSeek-AI et al.,

[1]Harbin Institute of Technology, Shenzhen [2]The Hong Kong University of Science and Technology [3]The Hong Kong University of Science and Technology (Guangzhou). Correspondence to: Qiang Wang <qiang.wang@hit.edu.cn>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

2025; Shazeer, 2020). Figure 1 illustrates the two imbalance sources that GreenMoE exploits.

The key observation behind GreenMoE is that waiting time before a synchronization point need not be eliminated to be useful. Workload-balanced long-context systems try to reduce imbalance directly (Ge et al., 2025; Wang et al., 2025); GreenMoE instead asks whether the ranks that would otherwise wait can spend the same wall-clock interval at a lower-energy operating point. This is attractive because the bottleneck rank can remain at the default high-frequency point, while non-critical ranks are slowed only when their predicted completion time remains within the synchronization boundary.

We propose GreenMoE, a dynamic-load-aware frequency scaling framework for long-context MoE training. GreenMoE estimates the current block load for Attention and MoE computation, identifies the critical-path time implied by the heaviest rank, and selects a lower GPU core frequency for non-critical ranks when the lower frequency is predicted to finish before the synchronization boundary. To make this practical inside a training system, GreenMoE uses CUDA events and an asynchronous control thread to align NVML frequency changes with specific model blocks without blocking the training thread. We further validate that GPU core-frequency changes have little effect on the All-to-All communication bandwidth in the measured setting, which supports applying frequency scaling during computation–communication overlap.

Our contributions are:

- We identify dynamic sequence-packing and expert-routing imbalance as exploitable energy slack in long-context MoE training.
- We formulate load-aware frequency selection for Attention and MoE blocks under a critical-path time constraint.
- We implement an asynchronous event-driven controller that applies block-level GPU frequency changes inside a Megatron-LM-style training pipeline.
- We evaluate GreenMoE on representative long-context MoE configurations, including a 64K setting that reduces per-GPU energy by 8.63% with 3.75% step-time overhead and a constructed Attention-heavy setting that reduces non-critical GPU energy by 16.15% with 0.23% overhead.

2. GreenMoE Method

2.1. Overview

GreenMoE converts dynamic rank imbalance into GPU energy savings through a conservative block-level control loop. The method has four parts. First, a load monitor reads quantities already available during training, such as packed sample lengths for Attention and routed token counts for MoE. Second, GreenMoE compares the local load with the maximum load in the relevant parallel group to estimate the synchronization boundary. Third, a frequency selector chooses the lowest-energy GPU core frequency that is predicted to finish before that boundary. Finally, an asynchronous controller applies the selected frequency at block boundaries without blocking the training thread. Figure 2 shows this control path.

2.2. Attention-load-aware scaling

Attention scaling targets imbalance introduced by sequence packing. Let rank i process a packed sequence containing M_i original samples with lengths $\{L_{i,m}\}_{m=1}^{M_i}$, hidden dimension d, and h Attention heads. Standard causal Attention contains linear projections, position encoding, score computation, softmax-weighted value aggregation, and an output projection. The projection terms scale with the number of tokens as $\mathcal{O}((\sum_m L_{i,m})d^2)$, while score computation and value aggregation scale quadratically with each original sample length. Because packed samples are masked from attending to each other, the dominant long-context workload is

$$\mathcal{F}_i^{\mathrm{Attn}} \propto d \sum_{m=1}^{M_i} L_{i,m}^2, \tag{1}$$

rather than $d(\sum_m L_{i,m})^2$. Lower-order projection and normalization costs are absorbed by the offline performance–power table used by the selector. GreenMoE communicates $\mathcal{F}_i^{\mathrm{Attn}}$ inside the relevant sequence-parallel or data-parallel group and obtains $\mathcal{F}_{\max}^{\mathrm{Attn}} = \max_i \mathcal{F}_i^{\mathrm{Attn}}$. This design uses metadata already available to the training pipeline, so the controller does not need to profile the Attention kernel online.

2.3. MoE-load-aware scaling

MoE scaling targets imbalance introduced by sparse expert routing. Let $\mathcal{E}$ be an expert-parallel group and let N_i be the number of routed tokens received by rank $i \in \mathcal{E}$ after dispatch. For a SwiGLU expert, each routed token performs an up projection, a gate projection, element-wise activation/-gating, and a down projection. With hidden dimension d, intermediate dimension d_{ff}, and top-k routing, the dominant per-token expert compute is approximated, up to common FLOP constants, as

$$C_{\mathrm{expert}} = 3kdd_{\mathrm{ff}}. \tag{2}$$

The rank-level MoE workload is therefore

$$\mathcal{F}_i^{\mathrm{MoE}} = N_i C_{\mathrm{expert}}, \tag{3}$$

and the group bottleneck is $\mathcal{F}_{\max}^{\mathrm{MoE}} = \max_{i\in\mathcal{E}} \mathcal{F}_i^{\mathrm{MoE}}$. Since C_{expert} is shared by ranks within the same block, the selector can use routed-token counts as the runtime load signal.

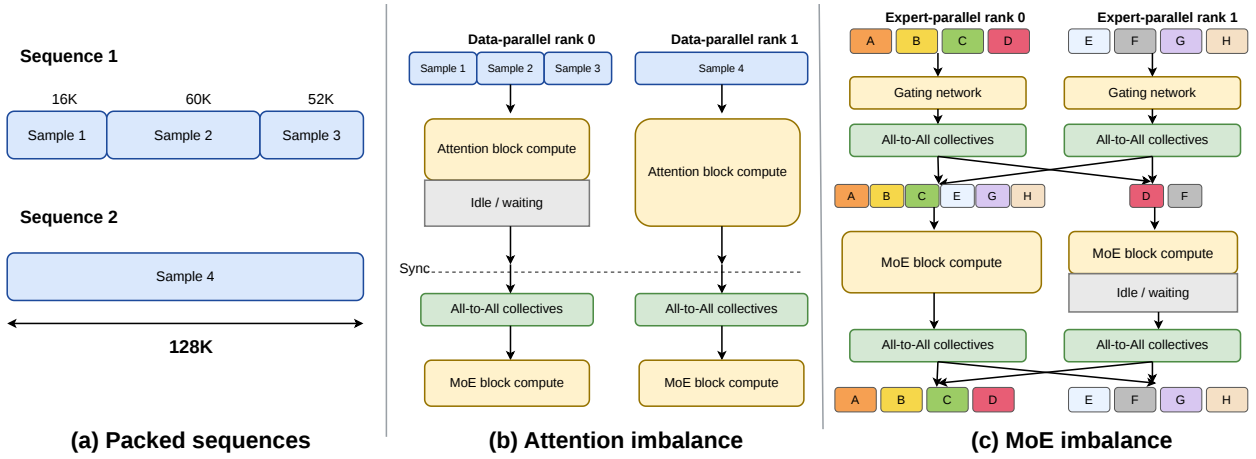


Figure 1. Motivating imbalance in long-context MoE training. Sequence packing changes causal-Attention work across ranks, while MoE routing changes expert-compute work after token dispatch. GreenMoE uses the resulting non-critical waiting time as a frequency-scaling budget instead of trying to eliminate the imbalance itself.

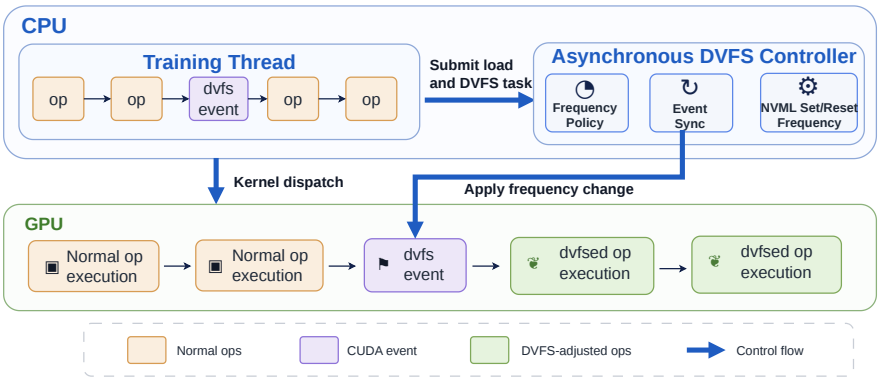


Figure 2. GreenMoE frequency-control mechanism. The training thread records CUDA events at target block boundaries and submits load information to a control thread. The control thread waits on the event and uses NVML to set or reset the GPU core frequency at the intended execution point.

GreenMoE reduces the frequency of ranks whose routed-token load is sufficiently below the expert-parallel maximum, while keeping the heaviest rank at the default high-frequency point.

2.4. Critical-path constrained frequency scaling

GreenMoE selects a lower frequency only when the predicted execution time remains within the synchronization boundary. We write q_i for the phase workload on rank i: $q_i = \mathcal{F}_i^{\text{Attn}}$ for Attention and $q_i = \mathcal{F}_i^{\text{MoE}}$ for MoE. Let $\mathcal{G}$ denote the synchronization group for the current phase, $q_{\max} = \max_{i \in \mathcal{G}} q_i$, and $f_{\max}$ be the default high-frequency point. If $\text{Perf}(f)$ denotes effective throughput at core frequency f, then the bottleneck rank defines the critical-path time

$$T_{\text{crit}} = \frac{q_{\max}}{\text{Perf}(f_{\max})}. \tag{4}$$

For a non-critical rank k, GreenMoE chooses a frequency f_k satisfying

$$\frac{q_k}{\text{Perf}(f_k)} \leq T_{\text{crit}}. \tag{5}$$

Equivalently, the selected frequency must provide enough effective throughput:

$$\text{Perf}(f_k) \geq \text{Perf}(f_{\max}) \frac{q_k}{q_{\max}}. \tag{6}$$

The ratio $\rho_k = q_k / q_{\max}$ is the available slack signal: ranks with small ρ_k can use lower throughput and still reach the same synchronization boundary, while ranks with ρ_k near one should remain close to $f_{\max}$. This constraint is the main safety rule in GreenMoE. The bottleneck rank defines the wall-clock budget, and only non-critical ranks are slowed.

2.5. Discrete frequency selection

GreenMoE uses a discrete search over supported GPU core frequencies. The selector receives the local load L, the max-

Algorithm 1 GreenMoE frequency selection

Input: local load L, max load $L_{\max}$, event e, lookup table $\mathcal{T}$, $f_{\max}$, tolerance τ, range $[f_{\text{lo}}, f_{\text{hi}}]$, step Δf, margin δ
Output: selected core frequency f
$\theta \leftarrow \text{Lookup}(\mathcal{T}, L)$; $\theta_{\max} \leftarrow \text{Lookup}(\mathcal{T}, L_{\max})$
$t_{\text{crit}} \leftarrow L_{\max}/\theta_{\max}.\text{Thr}$
if $L \geq (1-\tau)L_{\max}$ **then**
  $f \leftarrow f_{\max}$
else
  $f \leftarrow f_{\max}$
  for f' from f_{hi} down to f_{lo} by Δf **do**
    predict $t(L, f'; \theta)$ and $P(L, f'; \theta)$
    if $t(L, f'; \theta) \leq t_{\text{crit}}$ and $P(L, f'; \theta)t(L, f'; \theta)$ improves energy **then**
      $f \leftarrow f'$
    end if
  end for
  $f \leftarrow f + \delta$
end if
$\text{sync}(e)$; $\text{SetCoreFreq}(f)$
return f

imum group load $L_{\max}$, a lookup table $\mathcal{T}$ containing offline performance–power parameters, the maximum frequency $f_{\max}$, a bottleneck tolerance τ, a frequency range, and a small safety margin δ. If L is close to $L_{\max}$, GreenMoE leaves the rank at $f_{\max}$. Otherwise it scans lower candidate frequencies, keeps only those predicted to finish before the critical-path time, and chooses the feasible point with the lowest predicted energy. In our experiments, the hardware supports a 375–2100 MHz core-frequency range, the near-bottleneck tolerance is $\tau = 0.05$, and the final safety margin is $\delta = 60$ MHz.

The safety mechanisms are intentionally conservative. The tolerance τ avoids scaling ranks whose load is too close to the bottleneck. The margin δ avoids turning a non-critical rank into a new critical path when model prediction or frequency-change latency is imperfect. We use $\tau = 0.05$ and $\delta = 60$ MHz in the reported experiments.

2.6. Communication-overlap safety

MoE implementations often overlap token exchange with expert computation, so frequency scaling would be unsafe if lowering GPU core frequency also reduced All-to-All bandwidth. GreenMoE therefore includes a communication validation experiment before applying frequency changes around overlapped MoE phases. On the measured RTX A6000 system, All-to-All bandwidth remains nearly stable as GPU core frequency varies from 510 MHz to 2010 MHz in 100 MHz steps across message sizes from 128 MB to 4 GB, with each frequency–message-size pair repeated 20 times. This observation, summarized in Figure 3, supports applying GreenMoE to computation phases that overlap with communication in the evaluated setting.

3. Implementation

GreenMoE is implemented around a Megatron-LM-style Transformer training loop. Frequency decisions are inserted at the boundaries of Attention and MoE sub-blocks in the forward pass. The backward pass mirrors the forward-pass boundaries: positions that were used to lower frequency in the forward pass correspond to frequency restoration points in the backward pass, and positions that restored frequency in the forward pass correspond to later frequency-setting points in the backward pass.

Directly invoking NVML from the training thread would introduce two problems. First, GPU kernel launches are asynchronous, so a synchronous host call can easily change the frequency too early or too late relative to the intended CUDA work. Second, frequent NVML calls can block the training thread and create overhead that overwhelms the energy savings for short blocks. GreenMoE therefore decouples frequency decisions from frequency application. The training thread records a CUDA event at each target boundary and submits the event plus load context to a control thread. The control thread waits for the event, applies the selected core frequency through NVML, and resets the frequency at the next boundary.
GreenMoE also removes redundant frequency resets when a subsequent block will immediately overwrite the frequency. This optimization matters because single-block phases can be short; avoiding unnecessary NVML calls reduces control overhead and improves the chance that a frequency decision affects the intended computation window.

4. Experiments

4.1. Setup

We evaluate GreenMoE on a single server with 8 NVIDIA RTX A6000 GPUs and dual Intel Xeon Platinum 8358 CPUs. The software stack uses Ubuntu 20.04.6, PyTorch 2.1.0, CUDA Toolkit 11.8, CUDA Runtime 12.2, NCCL 2.18.5, and NVIDIA driver/NVML 535.161.08. The training data is EnWiki. Samples are packed into long sequences, and Attention uses FlashAttention variable-length kernels (Dao et al., 2022). Sequence parallelism follows the Ulysses-style sequence split (Jacobs et al., 2023). Each configuration is repeated three times; each run executes 30 training steps and measures steady-state power from step 10 onward with a 20 ms sampling interval. The tables report averages over GPUs and repeated runs. The recorded measurements do not retain per-run standard deviations, so we report deterministic averages rather than error bars.

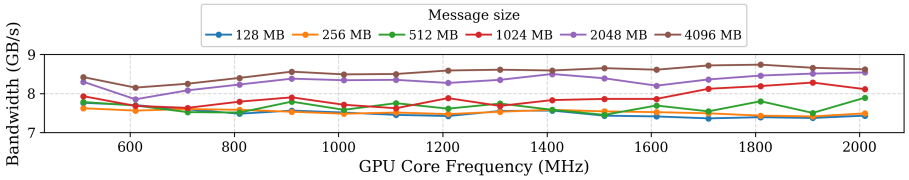


Figure 3. All-to-All bandwidth under different GPU core frequencies on the evaluated RTX A6000 system. Bandwidth changes weakly across the measured frequency range, supporting GreenMoE's use of core-frequency scaling during overlapped MoE computation.

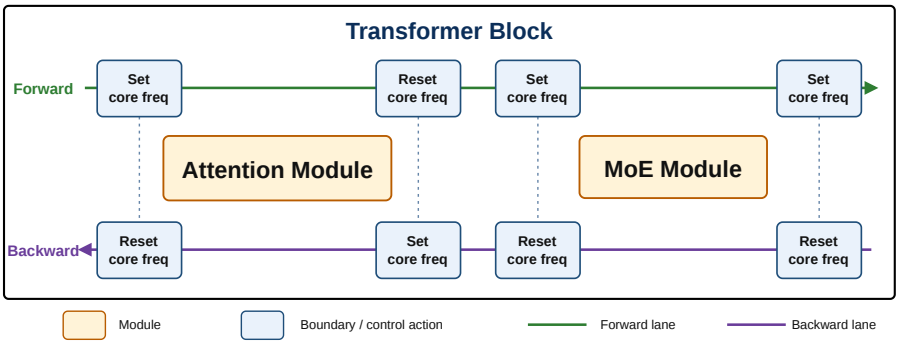


Figure 4. Block-level frequency placement in a Transformer layer. GreenMoE sets frequencies before Attention and MoE computation and restores or overwrites frequencies at subsequent block boundaries. Redundant resets can be removed when a later setting immediately overwrites them.

The baseline is NVIDIA's default boost policy. We compare three GreenMoE variants: ATTN, which scales only Attention blocks; MOE, which scales only MoE blocks; and ATTN+MOE, which scales both. Table 1 reports step time, average per-GPU power, and per-GPU step energy. We do not directly compare against global schedule optimizers such as Perseus or Kareus because our implementation does not include those systems and because their profiled global schedules target a different assumption set. We instead treat the default boost policy as the operational baseline and evaluate whether dynamic block-level scaling provides measurable energy savings under the same training configuration. Static low-frequency baselines are also omitted because they would slow the critical path by construction; GreenMoE's target behavior is specifically to lower non-critical ranks while preserving the synchronization bound.

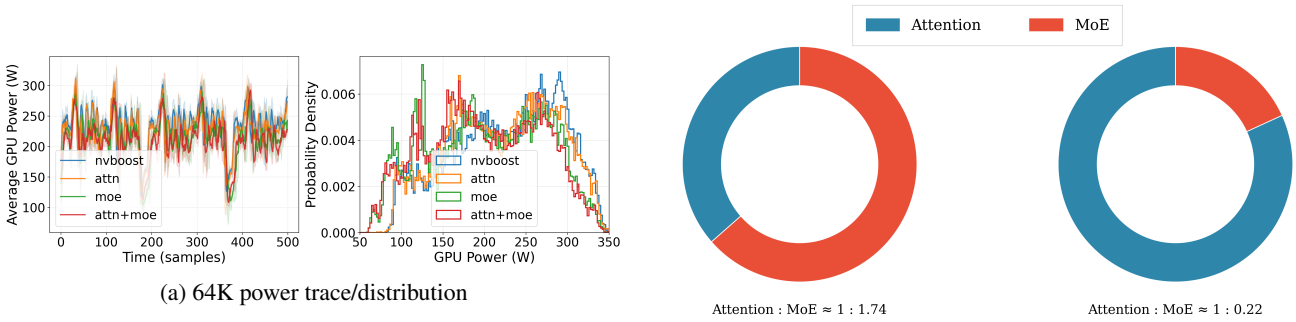
4.2. Single-node long-context MoE training

GreenMoE lowers power across all configurations, but the energy-throughput trade-off depends on block granularity. The largest single-node energy reduction is the 32K ATTN+MOE configuration, which reduces per-GPU energy by 10.20% but increases step time by 9.99%. The 64K setting is a more balanced point: ATTN+MOE reduces energy by 8.63% with 3.75% step-time overhead, while ATTN alone has only 0.18% overhead and still reduces energy by 2.26%. The 128K setting follows the same trend, with MOE reducing energy by 9.29% and ATTN+MOE reducing energy by 8.92%.

Table 1. Single-node GreenMoE results. Energy is average per-GPU step energy; percentages are relative to the default boost baseline in the same configuration.

Seq.	Policy	Step time (ms)	Power (W)	Energy (J)
32K	boost	4108.81	222.67	914.93
32K	ATTN	4155.02 (+1.12%)	212.15 (-4.72%)	881.51 (-3.65%)
32K	MOE	4462.69 (+8.61%)	184.87 (-16.98%)	825.04 (-9.82%)
32K	ATTN+MOE	4520.23 (+9.99%)	181.75 (-18.37%)	821.55 (-10.20%)
64K	boost	3992.95	226.31	903.66
64K	ATTN	4000.09 (+0.18%)	220.81 (-2.43%)	883.26 (-2.26%)
64K	MOE	4117.44 (+3.12%)	203.18 (-10.22%)	836.59 (-7.42%)
64K	ATTN+MOE	4142.57 (+3.75%)	199.32 (-11.93%)	825.70 (-8.63%)
128K	boost	2834.32	248.53	704.42
128K	ATTN	2863.66 (+1.04%)	237.07 (-4.61%)	678.90 (-3.62%)
128K	MOE	2997.81 (+5.76%)	213.16 (-14.23%)	639.01 (-9.29%)
128K	ATTN+MOE	3028.03 (+6.82%)	211.88 (-14.75%)	641.58 (-8.92%)

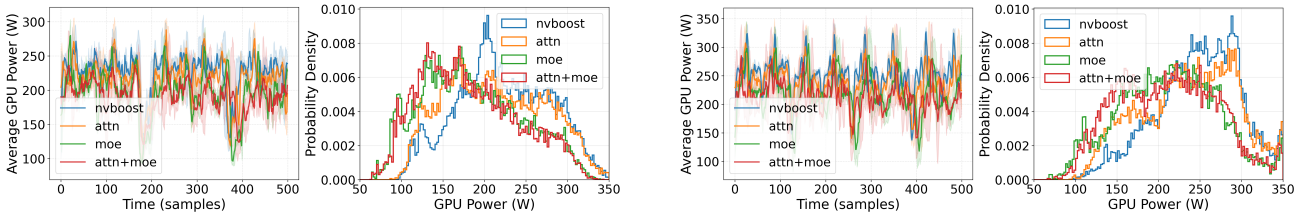
Figure 5 helps explain where GreenMoE is most effective. The 64K power plot directly shows that GreenMoE lowers average GPU power in a representative single-node setting. The workload distributions explain the policy preference behind that reduction: in the real-data configurations, MoE scaling reduces power more than Attention scaling because MoE computation occupies a larger share of the layer workload and because most expert-parallel ranks can be non-critical when routing is imbalanced. By contrast, the constructed Attention-heavy distribution shifts more workload into Attention, which is why the next experiment favors ATTN. MoE scaling can also introduce larger step-time overhead because more ranks participate in frequency changes and because lightly loaded expert phases can be short. The 32K setting is the clearest example: block-level compu-



(a) 64K power trace/distribution

(b) real-data and constructed workload distributions

Figure 5. Representative evidence for GreenMoE. The 64K power plot shows lower power under GreenMoE policies. The workload distributions explain why the best scaling target changes with workload composition: MoE occupies a larger share in the real-data setting, matching the stronger gains from MoE and ATTN+MOE in Table 1, while Attention occupies a larger share in the constructed Attention-heavy setting, matching the stronger gains from ATTN in Table 2.



(a) 32K power trace/distribution

(b) 128K power trace/distribution

Figure 6. Additional single-node power traces and power distributions. Together with the 64K plot in Figure 5, these plots show the same qualitative trend: GreenMoE policies lower average GPU power across the evaluated sequence lengths.

tation is short enough that NVML latency and frequency transition time become more visible. Our measurements do not separately log controller latency or the number of NVML calls per step, so this overhead attribution is based on the timing trend and implementation structure rather than a direct microbenchmark.

4.3. Constructed Attention-heavy setting

To isolate the Attention opportunity, we construct a 64K setting where one sequence-parallel group processes one 64K sample and another processes two 32K samples. This increases the Attention workload imbalance while keeping the rest of the configuration close to the 64K experiment. Figure 5 visualizes this shift, and Table 2 reports both the non-critical GPU group and all-GPU average.

Table 2. Constructed Attention-heavy experiment. Attention scaling is most effective on the non-critical GPU group and remains beneficial in the all-GPU average.

Scope	Policy	Step time (ms)	Power (W)	Energy (J)
GPU 4–7	boost	8171.59	217.64	1778.43
GPU 4–7	ATTN	8190.71 (+0.23%)	182.06 (-16.34%)	1491.24 (-16.15%)
GPU 4–7	MOE	8458.18 (+3.51%)	200.61 (-7.82%)	1696.82 (-4.59%)
All GPUs	boost	8171.59	247.95	2026.15
All GPUs	ATTN	8190.71 (+0.23%)	225.73 (-8.96%)	1848.85 (-8.75%)
All GPUs	MOE	8458.18 (+3.51%)	231.98 (-6.44%)	1962.10 (-3.16%)

The constructed setting confirms that Attention scaling is valuable when long samples dominate the workload imbalance. On the non-critical GPU group, ATTN reduces step energy by 16.15% while increasing step time by only 0.23%. Averaged across all GPUs, the same policy reduces energy by 8.75%. This result supports the central design choice of GreenMoE: the best target block depends on the current workload distribution, so the controller should reason about dynamic load rather than applying one static policy to all phases.

Additional two-node supporting results are deferred to Appendix E.2.

5. Conclusion

GreenMoE exploits dynamic load imbalance in long-context MoE training as an energy-saving opportunity. By selecting block-level GPU frequencies under a critical-path constraint and applying them through an asynchronous controller, it reduces training energy in both representative single-node runs and a constructed Attention-heavy setting. The results suggest that energy-efficient long-context MoE training benefits from reasoning about runtime load variation rather than using one fixed frequency policy for all phases.

References

Braun, L., Nikas, S., Song, C., Heuveline, V., and Fröning, H. A simple model for portable and fast prediction of execution time and power consumption of gpu kernels. *ACM Transactions Architecture and Code Optimization*, 18(1), dec 2021.

Bridges, R. A., Imam, N., and Mintz, T. M. Understanding gpu power: A survey of profiling, modeling, and simulation methods. *ACM Computing Surveys*, 49(3):27, September 2016.

Choi, S., Koo, I., Ahn, J., Jeon, M., and Kwon, Y. Envpipe: Performance-preserving dnn training framework for saving energy. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, pp. 851–864, 2023.

Chung, J.-W., Gu, Y., Jang, I., Meng, L., Bansal, N., and Chowdhury, M. Reducing energy bloat in large model training. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, pp. 144–159, 2024.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. Flashattention: fast and memory-efficient exact attention with io-awareness. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

DeepSeek-AI, Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Guo, D., Yang, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Zhang, H., Ding, H., Xin, H., Gao, H., Li, H., Qu, H., Cai, J. L., Liang, J., Guo, J., Ni, J., Li, J., Wang, J., Chen, J., Chen, J., Yuan, J., Qiu, J., Li, J., Song, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Xu, L., Xia, L., Zhao, L., Wang, L., Zhang, L., Li, M., Wang, M., Zhang, M., Zhang, M., Tang, M., Li, M., Tian, N., Huang, P., Wang, P., Zhang, P., Wang, Q., Zhu, Q., Chen, Q., Du, Q., Chen, R. J., Jin, R. L., Ge, R., Zhang, R., Pan, R., Wang, R., Xu, R., Zhang, R., Chen, R., Li, S. S., Lu, S., Zhou, S., Chen, S., Wu, S., Ye, S., Ye, S., Ma, S., Wang, S., Zhou, S., Yu, S., Zhou, S., Pan, S., Wang, T., Yun, T., Pei, T., Sun, T., Xiao, W. L., Zeng, W., Zhao, W., An, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Li, X. Q., Jin, X., Wang, X., Bi, X., Liu, X., Wang, X., Shen, X., Chen, X., Zhang, X., Chen, X., Nie, X., Sun, X., Wang, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yu, X., Song, X., Shan, X., Zhou, X., Yang, X., Li, X., Su, X., Lin, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhu, Y. X., Zhang, Y., Xu, Y., Xu, Y., Huang, Y., Li, Y., Zhao, Y., Sun, Y., Li, Y., Wang, Y., Yu, Y., Zheng, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Tang, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Wu, Y., Ou, Y., Zhu, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Zha, Y., Xiong, Y., Ma, Y., Yan, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Wu, Z. F., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Huang, Z., Zhang, Z., Xie, Z., Zhang, Z., Hao, Z., Gou, Z., Ma, Z., Yan, Z., Shao, Z., Xu, Z., Wu, Z., Zhang, Z., Li, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Gao, Z., and Pan, Z. Deepseek-v3 technical report. Technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Fernandez, J., Na, C., Tiwari, V., Bisk, Y., Luccioni, S., and Strubell, E. Energy considerations of large language model inference and efficiency optimizations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32556–32569, 2025.

Gao, T., Wettig, A., Yen, H., and Chen, D. How to train long-context language models (effectively). In *Annual Meeting of the Association for Computational Linguistics*, 2024.

Ge, H., Feng, J., Huang, Q., Fu, F., Nie, X., Zuo, L., Lin, H., Cui, B., and Liu, X. Bytescale: Communication-efficient scaling of llm training with a 2048k context length on 16384 gpus. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pp. 963–978, 2025.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Lakhotia, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh,

M., Paluri, M., Kardas, M., Tsimpoukelli, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Zhang, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Maheswari, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Albiero, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Wang, X., Tan, X. E., Xia, X., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Srivastava, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Teo, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Dong, A., Franco, A., Goyal, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Liu, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Gao, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Le, E.-T., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Kokkinos, F., Ozgenel, F., Caggioni, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Inan, H., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Zhan, H., Damlaj, I., Molybog, I., Tufanov, I., Leontiadis, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Lam, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Jagadeesh, K., Huang, K., Chawla, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Liu, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Mehta, N., Laptev, N. P., Dong, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Parthasarathy, R., Li, R., Hogan, R., Battey, R., Wang, R., Howes, R., Rinott, R., Mehta, S., Siby, S., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Mahajan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Patil, S., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Deng, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Koehler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wu, X., Wang, X., Wu, X., Gao, X., Kleinman, Y., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Zhao, Y., Hao, Y., Qian, Y., Li, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., Zhao, Z., and Ma, Z. The llama 3 herd of models. Technical report, 2024. URL https://arxiv.org/abs/2407.21783.

Guerreiro, J., Ilic, A., Roma, N., and Tomas, P. Gpgpu power modeling for multi-domain voltage-frequency scaling. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 789–800, 2018.

Hong, S. and Kim, H. An integrated gpu power and performance model. In *Annual International Symposium on Computer Architecture*, volume 38, pp. 280–289. ACM, 2010.

Jacobs, S. A., Tanaka, M., Zhang, C., Zhang, M., Song, S. L., Rajbhandari, S., and He, Y. Deepspeed ulysses: System optimizations for enabling training of extreme

long sequence transformer models. *arXiv preprint arXiv:2309.14509*, 2023.

Kakolyris, A. K., Masouros, D., Vavaroutsos, P., Xydis, S., and Soudris, D. Slo-aware gpu frequency scaling for energy efficient llm inference serving. *arXiv preprint arXiv:2408.05235*, 2024.

Liu, H., Zaharia, M., and Abbeel, P. Ring attention with blockwise transformers for near-infinite context. 2023.

Mei, X., Wang, Q., and Chu, X. A survey and measurement study of gpu dvfs on energy conservation. *Digital Communications and Networks*, 3(2):89 – 100, 2017.

NVIDIA Corporation. Nvidia profiler user's guide. Technical report, NVIDIA Corporation, Santa Clara, CA, USA, 2023. Online: http://docs.nvidia.com/cuda/profiler-users-guide, accessed: 2026-03-07.

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, J. H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokorny, Pokrass, M., Pong, V. H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M. B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. Gpt-4 technical report. Technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Patel, P., Choukse, E., Zhang, C., Goiri, Í., Warrier, B., Mahalingam, N., and Bianchini, R. Characterizing power management opportunities for llms in the cloud. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 207–222, 2024.

Qiu, H., Mao, W., Patke, A., Cui, S., Jha, S., Wang, C., Franke, H., Kalbarczyk, Z., Başar, T., and Iyer, R. K. Power-aware deep learning model serving with μ-Serve. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pp. 75–93, 2024.

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press, 2020. ISBN 9781728199986.

Shazeer, N. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2020.

Song, S., Su, C., Rountree, B., and Cameron, K. W. A simplified and accurate model of power-performance efficiency on emergent gpu architectures. In *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, pp. 673–686. IEEE, 2013.

Tang, Z., Wang, Y., Wang, Q., and Chu, X. The impact of gpu dvfs on the energy and performance of deep learning: An empirical study. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, e-Energy '19, pp. 315–325, New York, NY, USA, 2019. Association for Computing Machinery.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Wang, Q. and Chu, X. Gpgpu performance estimation with core and memory frequency scaling. *IEEE Transactions on Parallel and Distributed Systems*, 31(12):2865–2881, 2020.

Wang, X., Huang, K., Knoll, A., and Qian, X. A hybrid framework for fast and accurate gpu performance estimation through source-level analysis and trace-based simulation. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 506–518. IEEE, 2019.

Wang, Z., Cai, A., Xie, X., Pan, Z., Guan, Y., Chu, W., Wang, J., Li, S., Huang, J., Cai, C., et al. Wlb-llm:workload-balanced 4d parallelism for large language model training. In *19th USENIX Symposium on Operating Systems Design and Implementation (OSDI 25)*, pp. 785–801, 2025.

Wu, R., Chung, J.-W., and Chowdhury, M. Kareus: Joint reduction of dynamic and static energy in large model training. *arXiv preprint arXiv:2601.17654*, 2026.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report. Technical report, 2025. URL https://arxiv.org/abs/2505.09388.

A. Background and Related Work

Long-context MoE training. Transformer language models use self-attention to mix information across tokens, and decoder-only variants are the foundation of many current large language models (Vaswani et al., 2017; OpenAI et al., 2024; DeepSeek-AI et al., 2025; Yang et al., 2025). Long-context training extends this mechanism to sequences that may reach tens or hundreds of thousands of tokens, typically relying on IO-aware or sequence-parallel Attention implementations to keep memory use tractable (Dao et al., 2022; Jacobs et al., 2023; Grattafiori et al., 2024; Liu et al., 2023). Because padding all samples to a maximum length wastes compute, long-context training often packs multiple shorter samples into one sequence. The model still prevents cross-sample Attention, so the true Attention workload depends on the squared lengths of the packed segments rather than only the packed total length (Vaswani et al., 2017; Dao et al., 2022). MoE layers introduce another dynamic workload source: a router sends each token to a small number of experts, and expert-parallel training redistributes tokens through All-to-All communication before expert computation (Shazeer et al., 2017; DeepSeek-AI et al., 2025; Shazeer, 2020). These mechanisms improve training scalability but make per-rank compute load step-dependent.

Parallel training and sequence organization. Large models are trained with combinations of data, tensor, pipeline, sequence, and expert parallelism. ZeRO shards optimizer and parameter states across data-parallel workers (Rajbhandari et al., 2020), while Megatron-LM popularizes tensor and pipeline model parallelism for large Transformer training (Shoeybi et al., 2020). Long-context systems such as Ulysses, GatherKV, and Ring Attention split or reorganize the sequence dimension to reduce memory pressure and communication cost (Jacobs et al., 2023; Grattafiori et al., 2024; Liu et al., 2023). Expert-parallel MoE systems distribute experts across GPUs and use token exchange to make sparse expert activation scalable (Shazeer et al., 2017; DeepSeek-AI et al., 2025). Workload-balanced long-context training attempts to reduce imbalance directly by changing sequence assignment or scheduling (Ge et al., 2025; Wang et al., 2025). GreenMoE is complementary: when imbalance remains before a synchronization point, it uses the non-critical ranks' waiting time as an energy budget.

GPU energy modeling and DVFS. GPU DVFS has been studied through runtime-counter models, static analysis, simulation, and microarchitectural reasoning (Bridges et al., 2016; Mei et al., 2017; Guerreiro et al., 2018; Hong & Kim, 2010; Song et al., 2013; Braun et al., 2021; Wang et al., 2019; Wang & Chu, 2020). Runtime-counter models can capture detailed workload behavior but often require profiling overhead or application instrumentation (Bridges et al., 2016; Mei et al., 2017; NVIDIA Corporation, 2023). Static and simulator-based models avoid online profiling but can lose accuracy when runtime cache, occupancy, or communication behavior changes (Hong & Kim, 2010; Song et al., 2013; Braun et al., 2021; Wang et al., 2019). These studies show that GPU power and performance do not scale linearly with core frequency and that the energy-optimal point depends on the workload (Tang et al., 2019; Wang & Chu, 2020). GreenMoE uses this observation operationally: instead of choosing one frequency for a whole job, it selects frequencies for short Attention and MoE phases based on current load and a precomputed performance–power table.

Energy-efficient large-model systems. Recent systems reduce training or inference energy by exploiting slack, scheduling, or power-aware execution. In inference, power provisioning and frequency-control systems exploit the gap between provisioned peak power and serving-time utilization while respecting latency or SLO constraints (Patel et al., 2024; Qiu et al., 2024; Kakolyris et al., 2024; Fernandez et al., 2025). In training, ENVPIPE uses pipeline idle time for energy savings, but its slack source is pipeline scheduling rather than data-dependent Attention or expert-routing imbalance (Choi et al., 2023). Perseus models training as a DAG and schedules lower-frequency execution under a profiled execution model, whereas GreenMoE recomputes the relevant load relation at block granularity because the bottleneck rank changes with packed samples and routed tokens (Chung et al., 2024). Kareus jointly reasons about compute, communication overlap, and GPU frequency; GreenMoE is narrower, focusing on runtime rank imbalance inside long-context MoE layers and using a conservative critical-path constraint instead of a global schedule (Wu et al., 2026). GreenMoE is closest in spirit to slack-aware training systems, but targets a different slack source: runtime load imbalance induced by sequence packing and expert routing in long-context MoE training.

B. Supplementary Workload Illustrations

Section 3 gives the Attention and MoE workload definitions used by GreenMoE. This appendix keeps the complementary timing illustrations, which show the same critical-path idea visually for Attention and MoE phases without repeating the main-text derivation.

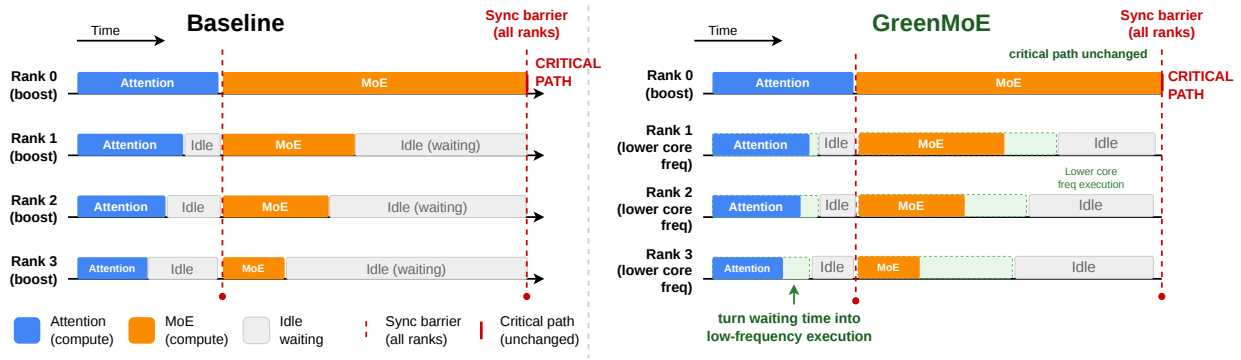


Figure 7. Idealized GreenMoE schedules. Non-critical ranks execute at lower frequency while still reaching the synchronization boundary no later than the critical rank.

C. Frequency-Selection Details

The main paper gives the compact form of GreenMoE frequency selection. This appendix records additional operational details.

Lookup table. GreenMoE uses a performance–power parameter table $\mathcal{T}$ built offline for representative matrix-multiplication-like workloads. At runtime, the selector looks up the row associated with the current load. If the load does not exactly match a table key, the implementation uses the nearest lower load bucket. The table provides the throughput at the maximum frequency and parameters for predicting time and power at lower candidate frequencies. This paper specifies the lookup interface and runtime policy, but exact reproduction of the selector also requires the offline table coefficients.

Critical-path time. The selector computes the critical-path bound from the maximum load in the same synchronization group. For Attention, the group is determined by the data/sequence-parallel organization. For MoE, the group is the expert-parallel group participating in token dispatch and combine.

Discrete search. Supported GPU core frequencies are discrete; the RTX A6000 system used here exposes a 375–2100 MHz core-frequency range. GreenMoE scans from a high candidate frequency down to a low candidate frequency with step Δf. A candidate is feasible only if the predicted execution time does not exceed the critical-path time. Among feasible points, GreenMoE minimizes estimated phase energy $E = P \cdot t$, subject to any configured power cap.

Safety margin. GreenMoE adds a small positive frequency margin δ after selecting the best feasible point. We use $\delta = 60$ MHz in the reported experiments. This margin is not intended to improve energy directly; it reduces the risk that model error or frequency-transition latency turns a non-critical rank into the bottleneck.

Controller timing. The selected frequency is applied only after the control thread observes the CUDA event associated with the block boundary. This event synchronization is what makes a host-side NVML operation line up with asynchronous GPU execution.

D. Experimental Details

The experiments use reduced but representative configurations due to hardware limits. The number of experts is set equal to the expert-parallel size to simplify implementation. This choice does not remove the key load-imbalance property because different expert-parallel ranks still receive different token counts after routing.

E. Additional Results

E.1. Constructed workload distribution

Figure 5 now combines the real-data and constructed Attention-heavy workload distributions in one place. The constructed distribution shifts more of the layer workload into Attention than the real-data case, which is why Attention-only scaling is

Table 3. Hardware configuration used in the experiments.

Component	Configuration
CPU	Dual Intel Xeon Platinum 8358 @ 2.60GHz
GPU	8× NVIDIA RTX A6000 48GB
GPU core frequency	375–2100 MHz
Memory	512 GB DDR4
NVLink	112.5 GB/s (4×)
PCIe	PCIe 4.0 x16
Network	Mellanox MT28908 @ 200 Gb/s

Table 4. Software stack used in the experiments.

Component	Version
Operating system	Ubuntu 20.04.6 LTS
Python	3.10.13
CUDA Toolkit	11.8
CUDA Runtime	12.2
PyTorch	2.1.0
cuDNN	8.7
NCCL	2.18.5
NVIDIA Driver / NVML	535.161.08

most effective in Table 2.

E.2. Two-node supporting results

We also evaluate GreenMoE on a two-node, 16-GPU setting as supporting evidence for the same trend. Table 7 reports average per-GPU step energy over all 16 GPUs. These results are not used as the primary claim because the main analysis in the paper focuses on the single-node setting and the controlled Attention-heavy construction, but they show that GreenMoE continues to reduce per-GPU energy when the experiment is extended to two nodes.

The two-node results are consistent with the single-node observations. MoE scaling gives the strongest energy reduction in the 32K and 128K settings, while the combined ATTN+MOE policy is strongest at 64K and 128K in average per-GPU energy. The time overhead remains non-negligible for some configurations, especially 32K ATTN+MOE, so we present these numbers as evidence of the same energy-throughput trade-off rather than as a throughput-free scaling claim.

F. Broader Impacts

GreenMoE targets energy efficiency in distributed MoE training, which carries both positive and negative societal implications.

Positive impacts. Reducing the energy consumption of large-scale model training directly lowers operational costs and carbon emissions associated with GPU compute. As model sizes and training durations continue to grow, system-level energy optimizations like GreenMoE can contribute to making AI research more environmentally sustainable. The method is also applicable beyond MoE architectures: any distributed training scenario with dynamic load imbalance could potentially benefit from the same critical-path-constrained frequency scaling principle.

Negative impacts. GreenMoE is a systems-level optimization with no direct path to harmful applications. However, we acknowledge that any technology that reduces the cost of training large models could indirectly lower the barrier for producing models that are later misused (e.g., for generating disinformation or deepfakes). This concern is not specific to GreenMoE and applies broadly to any efficiency improvement in training infrastructure. We believe the positive environmental benefits outweigh this indirect risk, and that addressing model misuse is better tackled through governance and deployment-stage safeguards rather than by withholding efficiency improvements.

Table 5. EnWiki sample-length distribution used to motivate sequence packing.

Length range	Samples	Fraction
0–1K	217,108	27.78%
1K–5K	415,258	53.14%
5K–10K	97,858	12.52%
10K–50K	49,190	6.29%
50K–100K	1,783	0.23%
>100K	234	0.03%

Table 6. Model configurations for the single-node experiments.

Parameter	Config 1	Config 2	Config 3
Transformer layers	8	4	4
Hidden dimension	4096	4096	2048
Attention heads	32	32	32
FFN dimension	16384	16384	8192
MoE experts	8	8	8
Top-k routing	1	1	1
MoE capacity factor	1.5	1.5	1.25
Sequence length	32K	64K	128K
Sequence parallel size	4	4	4
Expert parallel size	8	8	8
Micro batch size	1	1	1
Global batch size	16	16	16

Table 7. Two-node GreenMoE results. Energy is average per-GPU step energy over 16 GPUs; percentages are relative to the default boost baseline in the same sequence-length configuration.

Seq.	Policy	Step time (ms)	Power (W)	Energy (J)
32K	boost	3664.46	184.18	674.93
32K	ATTN	3883.72 (+5.98%)	172.21 (-6.50%)	668.81 (-0.91%)
32K	MOE	3977.51 (+8.54%)	152.90 (-16.98%)	608.17 (-9.89%)
32K	ATTN+MOE	4068.89 (+11.04%)	151.70 (-17.64%)	617.25 (-8.55%)
64K	boost	3540.99	187.59	664.26
64K	ATTN	3551.48 (+0.30%)	184.08 (-1.87%)	653.75 (-1.58%)
64K	MOE	3783.52 (+6.85%)	162.65 (-13.30%)	615.39 (-7.36%)
64K	ATTN+MOE	3787.40 (+6.96%)	160.81 (-14.28%)	609.04 (-8.31%)
128K	boost	2425.66	202.96	492.31
128K	ATTN	2450.48 (+1.02%)	193.94 (-4.44%)	475.24 (-3.47%)
128K	MOE	2546.74 (+4.99%)	173.57 (-14.48%)	442.03 (-10.21%)
128K	ATTN+MOE	2579.80 (+6.35%)	170.21 (-16.14%)	439.10 (-10.81%)