# Characterizing Energy and Performance for Distributed Training of Large Language Models

Zhendong Zhang
ETH Zurich
Switzerland
zzhendong@ethz.ch

Foteini Strati
ETH Zurich
Switzerland

Ana Klimovic
ETH Zurich
Switzerland

## Abstract

Distributed LLM training dominates compute, power and energy consumption in modern datacenters. Parallelization strategies are tuned to to maximize training throughput under memory constraints, while GPU frequency can be adjusted to reduce power. However, these knobs are usually tuned in isolation, and their combined effect on energy efficiency is poorly understood. We evaluate how different parallelization strategies interact with GPU frequency scaling, measuring throughput, power draw, and total energy across different transformer models. We find that faster configurations generally reduce energy under fixed hardware settings, but sensitivity to reduced GPU frequency varies widely across parallelism strategies. As a result, achieving Pareto-optimal performance-energy tradeoffs, requires joint optimization of parallelization strategies and GPU frequency. We provide guidelines for scheduling and modeling distributed LLM training to balance performance, power, and energy constraints.

**ACM Reference Format:**
Zhendong Zhang, Foteini Strati, and Ana Klimovic. 2026. Characterizing Energy and Performance for Distributed Training of Large Language Models. In *The 6th Workshop on Machine Learning and Systems (EuroMLSys '26), April 27–30, 2026, Edinburgh, Scotland Uk.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3805621.3807634

## 1 Introduction

The rapid scaling of large language models (LLMs) has made distributed training one of the most compute and energy intensive workloads in modern data centers. Training state-of-the-art transformer models [3, 13, 16] requires multi-GPU clusters and complex combinations of parallelism strategies to meet scalability and memory demands [15, 19, 26, 27]. As models scale and cluster sizes grow, energy consumption and power draw become primary constraints, influencing operational cost, provisioning, and system design [5, 11, 17].



*EuroMLSys '26, Edinburgh, Scotland Uk*



ACM ISBN 979-8-4007-2605-7/26/04
https://doi.org/10.1145/3805621.3807634

Distributed LLM training exposes several knobs that influence performance, resource utilization and power consumption. Practitioners select parallelism strategies such as data, tensor, pipeline, and expert parallelism to partition model parameters and computation across devices [15, 19, 26, 27]. These choices determine the memory footprint and throughput of training workloads. At the hardware level, modern GPUs provide coarse-grained frequency control [31, 32], enabling operators to trade performance for reduced power draw.

Although these knobs are routinely tuned in practice, their interaction remain poorly understood. Parallelism is typically configured to maximize throughput under memory constraints [21, 29, 35], while frequency scaling is either automatically adjusted by firmware or carefully tuned to reduce power when GPU utilization is low [4, 6, 31]. Thus, performance and energy knobs are generally optimized in isolation. It remains unclear how these mechanisms compare as energy efficiency levers and how GPU frequency control interacts with different parallelism strategies.

In this paper, we present a systematic characterization of distributed LLM training across parallelism configurations and GPU core frequency settings. We evaluate different parallelization strategies on NVIDIA GH200 and A100 GPUs and measure throughput, power, and total energy for dense and sparse models of varying sizes. We also analyze the interaction between frequency scaling and parallelization strategies, and characterize the power–performance trade-offs across different scenarios.

Our study reveals three main findings. First, under fixed cluster sizes and GPU frequencies, configurations that reduce runtime, also reduce total energy consumption. Second, because parallelization strategies differ in communication/computation ratios and resource utilization, their sensitivity to frequency scaling varies: some suffer significantly under reduced GPU core frequencies, while others are more robust. As a consequence, our third insight is that achieving Pareto-optimal performance–energy trade-offs requires *joint optimization* of parallelization strategy and GPU frequency. In particular, the configuration that minimizes runtime is not always the one that minimizes energy. For example, we show that a slower pipeline-parallel configuration, when combined with GPU core frequency down-scaling, can consume less total energy than the fastest data-parallel configuration.

Based on these findings, we provide a roadmap and guidance for jointly modeling power consumption along with throughput in distributed LLM training, and for making scheduling decisions that optimize energy use and performance under energy, power, and runtime constraints.

## 2 Background and Related Work

### 2.1 Distributed LLM Training

**Parallelism strategies.** Modern large language models are predominantly based on transformer architectures composed of stacked attention and feed-forward (FFN) layers. Training these models at scale exceeds the memory and compute capacity of a single GPU, necessitating distributed parallelism across multiple devices.

Data parallelism (DP) [19] replicates the full model on each device and partitions the input batch, synchronizing gradients across GPUs at every iteration. Tensor parallelism (TP) [27, 35] partitions individual layers across devices, splitting matrix multiplications or attention heads so that each GPU computes a portion of the layer and exchanges intermediate activations. Pipeline parallelism (PP) [9, 15] divides the model into sequential stages that are assigned to different devices, allowing microbatches to flow through the stages to improve utilization. Expert parallelism [26, 36], used in Mixture-of-Experts (MoE) models, distributes different feed forward experts across GPUs and routes tokens dynamically to the selected experts, introducing token level communication between devices. These parallelism strategies can be combined in hybrid configurations to balance memory capacity, communication cost, and scalability. The resulting execution consists of alternating compute and communication phases, whose duration and overlap determine iteration time, device utilization, average power, and ultimately total energy consumption.

### 2.2 GPU Frequency Control

Modern GPUs support dynamic voltage and frequency scaling (DVFS) [1, 2, 10], allowing operators to adjust the frequency of the streaming multiprocessors (SMs) within a supported range (e.g., an NVIDIA A100 GPU supports 81 discrete core frequency levels). Changing frequency affects both power and execution speed: higher frequency generally increases throughput but raises power consumption, while lower frequency reduces power at the potential cost of longer runtime.

The impact of frequency scaling on energy is workload dependent, since energy is determined by both power and execution time. Prior work suggests that compute-intensive workloads tend to be more sensitive to GPU core frequency changes [31]. In distributed LLM training, different parallelization strategies introduce varying computation and communication ratios, and GPU utilization patterns, suggesting that their response to frequency scaling may also differ.

### 2.3 Related Work

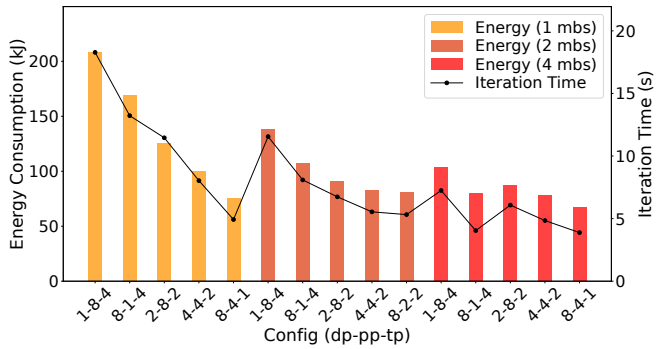
**Energy-efficient LLM training and power management.** Recent work explores reducing the energy consumption of large-scale ML training through hardware and system level power management. EnvPipe [4] leverages pipeline bubbles to safely down-scale GPU core frequency during non-critical computations, while Perseus [6] analyzes pipeline imbalance and reconstructs time-energy trade-off frontiers for pipeline-parallel training. Zeus [33] formulates energy optimization as an online exploration problem, dynamically tuning batch size and GPU power limits. At larger scales, prior work studies datacenter-level power control, including QoS and hardware aware power capping [18, 22]. These approaches primarily optimize device-level power or frequency under fixed training configurations, either within a single node or under a user-specified parallelization strategy. In contrast, distributed LLM training introduces multiple parallelization choices that fundamentally affect device utilization and communication behavior. Our work shows that the effectiveness of GPU core frequency scaling depends strongly on the chosen parallelism structure, requiring joint consideration of both dimensions.

**Characterization of distributed training systems.** Another line of work focuses on understanding the performance, power, and energy consumption of large-scale training workloads. Measurements from production AI clusters reveal substantial power variability induced by the computation and communication phases of distributed training workloads [5]. Complementary efforts focus on carbon and sustainability analysis of AI systems. WattsOnAI [14] enables framework-integrated tracking of energy and carbon metrics, while recent studies evaluate the environmental impact of large language models across hardware platforms [20, 28]. However, existing characterization studies largely analyze performance, power, or optimization techniques in isolation. In contrast, our work provides a systematic characterization across parallelism configurations and GPU core frequency settings, examining how their interaction shapes performance, power, and energy consumption.

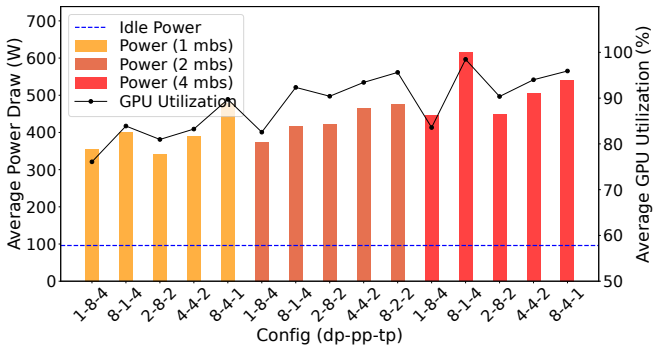
## 3 Energy efficiency analysis for distributed LLM training

In this section, we analyze the impact of parallelization strategies and GPU core frequency scaling in runtime, energy and power consumption of distributed LLM training. We aim to answer the following questions:

1. Under fixed cluster sizes and GPU frequencies, how do parallelization strategies affect energy and power consumption for dense and sparse models?
2. How does GPU core frequency scaling affect energy and power consumption across different parallelization strategies?

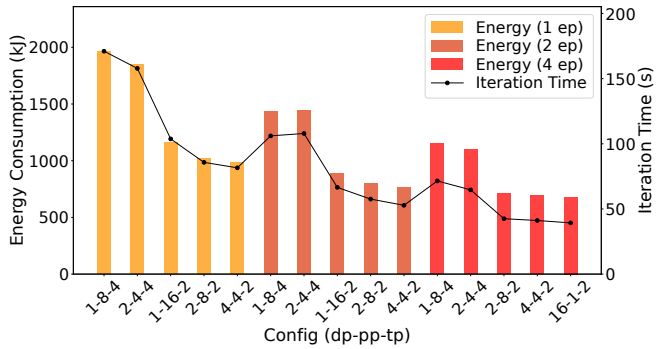



**(a)** Total energy and iteration time

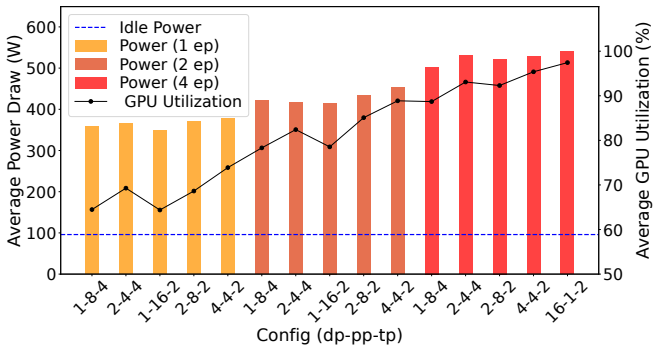



**(b)** Average power and average GPU utilization

**Figure 1.** Energy and performance across parallelism and micro-batch configurations for the OPT-350M model.




**(a)** Total energy and iteration time




**(b)** Average power and average GPU utilization

**Figure 2.** Energy and performance across parallelism configurations for the OLMoE-1B-7B model (mbs=1).

### 3.1 Evaluation Setup

**Cluster Configuration:** Experiments are conducted on two GPU clusters. Our detailed characterization of parallelism strategies is performed on an on-premise cluster comprising 8 nodes, each equipped with 4 NVIDIA GH200. GPU frequency scaling is evaluated on a Google Cloud Platform (GCP) cluster with 2 nodes, each equipped with 4 NVIDIA A100-40GB GPUs [7]. On this platform, we have administrative access to configure the GPU core frequency, enabling controlled DVFS experiments.

**Workloads:** We evaluate a dense transformer, OPT-350M [34], with 350M parameters, and a mixture of experts (MoE) model, OLMoE-1B-7B [23], with 1B active parameters and 7B total parameters. Unless otherwise specified, we use a global batch size of 1024 and sequence lengths of 2048 and 4096, respectively.

**Measurement Methodology:** We use Megatron-DeepSpeed, which supports data (DP), tensor (TP), pipeline (PP), and expert parallelism (EP) [8]. For MoE models, tensor parallelism is applied to attention layers, while expert parallelism is applied to feed forward layers. We vary the degrees of DP, TP, PP, and EP subject to hardware constraints, ensuring that all configurations fit within GPU memory limits.

We use the NVIDIA Management Library (NVML) to configure GPU core frequency[1] and to measure per GPU power and total energy per training iteration, and use NVIDIA Data Center GPU Manager (DCGM) to measure GPU utilization [24, 25]. All reported results are averaged over multiple steady state iterations after excluding initialization and warm up phases. Variance is small and omitted for clarity, unless otherwise noted. This methodology enables a direct comparison of parallelism scaling and frequency scaling as system-level knobs affecting both performance and energy.

### 3.2 Observations and Analysis

#### 3.2.1 Energy and Performance Analysis Across Parallelization Strategies.

We first analyze energy, power, and iteration time across parallelism configurations, without GPU frequency scaling, for both a dense model (OPT-350M) and a

[1]We do not vary GPU memory clocks in this work: in our setup, A100-40GB GPUs expose only one usable memory-frequency setting, while GH200 GPUs support two settings, but changing them required NVIDIA driver reloading, introducing seconds-level disruption and making memory-frequency tuning unsuitable during application execution.

mixture-of-experts model (OLMoE-1B-7B) on our 32-GH200 cluster.

**Dense Model:** Figure 1a shows total cluster energy per iteration and iteration time across parallelism configurations for OPT-350M. We vary the micro-batch size (mbs) from 1 to 4. Across all configurations, total energy closely follows iteration time: configurations with shorter iteration time consistently consume less energy.

Figure 1b reports per-device average power alongside average GPU utilization. The idle power of a GH200 GPU with no workload is about 96W, which provides an approximate lower bound on power consumption; average GPU utilization serves as a rough proxy for the dynamic power induced by the workload.

Comparing Figures 1b and 1a, average power varies less than iteration time across parallelism configurations and exhibits an inverse relationship with iteration time. Configurations that achieve higher throughput, typically those with larger micro-batch sizes and higher data parallel degrees (e.g., $dp$=8, $pp$=4, $tp$=1 with $mbs$=4), draw more power due to higher GPU utilization. However, the reduction in iteration time outweighs the increase in power, leading to lower total energy.

In contrast, configurations with higher tensor or pipeline parallel degrees reduce per-device memory footprint but introduce additional communication or pipeline overhead. These configurations exhibit longer iteration times and correspondingly higher total energy, even when their average power is slightly lower. Overall, differences in energy across parallelism strategies are driven primarily by differences in runtime rather than differences in average power.

**MoE Model:** For OLMoE-1B-7B, which has 7B total parameters, we fix the micro-batch size at 1 due to memory constraints and vary the expert parallel degree from 1 to 4. Figures 2a and 2b show total energy and per-device average power, respectively.

As in the dense model, increasing the expert parallel degree slightly increases per-device power (due to higher GPU utilization) but reduces iteration time, resulting in lower total energy. In our hybrid configuration, expert parallelism is applied only to feed forward experts and does not reduce the degree of other parallel dimensions. As a result, expert parallelism provides an additional scaling dimension that improves throughput and saves energy while maintaining feasible memory usage within a node.

***Insight 1.*** Configurations that increase training throughput generally also minimize total energy, as energy variations across parallelism strategies closely track iteration time.

#### 3.2.2 Sensitivity of Parallelism Strategies to Frequency Scaling.

Next, we explore how GPU core frequency affects power and energy consumption across different parallelization strategies. We profile OPT-350M training on NVIDIA A100 GPUs in Google Cloud.

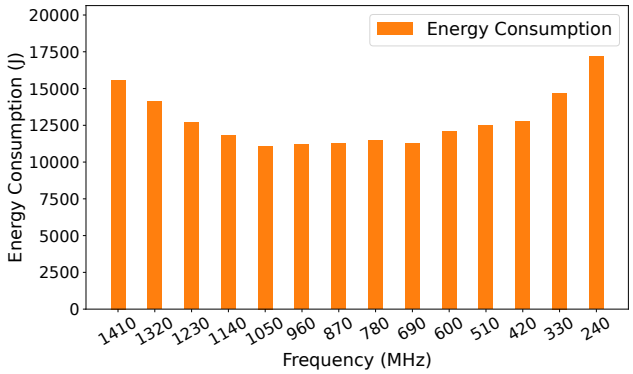



**Figure 3.** Energy vs. GPU core frequency for OPT-350M training on an NVIDIA A100 GPU.

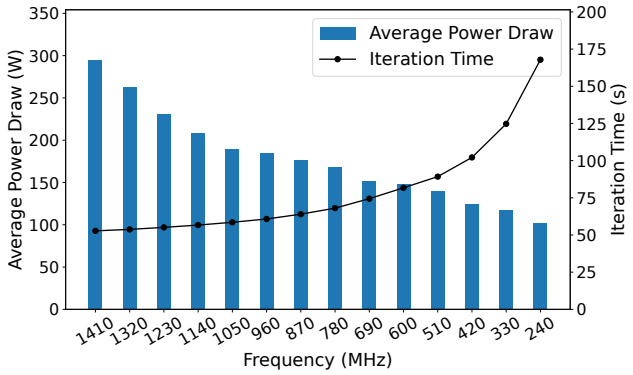



**Figure 4.** Power and iteration time vs. GPU core frequency for OPT-350M training on an NVIDIA A100 GPU.

We start by analyzing a single GPU configuration, with a global batch size of 256 and a micro-batch size of 1, under varying GPU frequencies. Figure 3 shows the total energy per iteration across the supported GPU core frequency range of 240-1410 MHz. Figure 4 shows the corresponding average power draw and iteration time. As GPU core frequency decreases, average power draw decreases monotonically, while iteration time increases monotonically due to reduced compute throughput. As a result, total energy exhibits a non-monotonic relationship with frequency: lowering the frequency initially reduces total energy because the reduction in power dominates, but beyond a certain point, the increase in iteration time outweighs the power savings, causing total energy to rise. This trend aligns with prior empirical studies of GPU DVFS for deep learning workloads, indicating that careful frequency scaling can achieve energy savings with minimal performance loss [4, 6, 31].

Having a clear understanding of the *valley*-like relationship between GPU core frequency and energy consumption, we now examine how different parallelism strategies respond

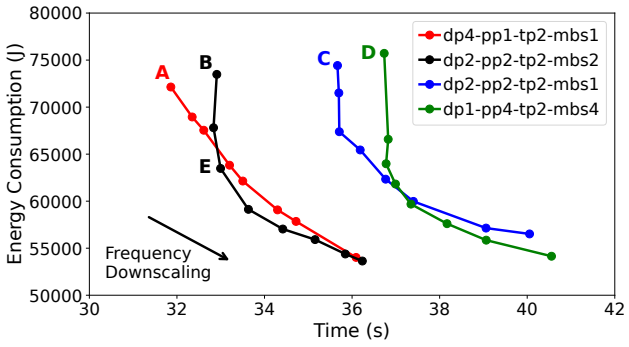



**Figure 5.** Energy-iteration time trade-offs under frequency scaling for different parallelism configurations. Each point corresponds to a different GPU core frequency setting. The separation between the blue/green and red/black curves is mainly driven by iteration-time differences across parallelism settings.

to GPU core frequency down-scaling. Parallelism configurations exhibit different computation and communication characteristics that lead to varying performance sensitivity under reduced GPU core frequencies.

Figure 5 shows the energy-iteration time trade-offs of several representative parallelism configurations evaluated across multiple frequency settings for OPT-350M training on our A100-40GB cluster (2 nodes with 4 GPUs per node). Each point corresponds to a different GPU core frequency setting, where pipeline stages may operate at different frequencies. All configurations start at the maximum frequency (points A–D), after which frequency is gradually reduced. For pipeline-parallel configurations, we first reduce the frequency of stages with lower GPU utilization (more pipeline bubbles) before applying uniform down-scaling across all stages.

Frequency sensitivity differs significantly across parallelism strategies. Data-parallel dominant configurations exhibit substantial performance degradation as frequency decreases, leading to rapidly increasing iteration time (e.g., *dp*4-*pp*1-*tp*2-*mbs*1 and *dp*2-*pp*2-*tp*2-*mbs*1). These configurations maintain high device utilization and are therefore more sensitive to reductions in compute throughput.

In contrast, pipeline-parallel configurations are more robust to moderate frequency reduction (e.g., *dp*2-*pp*2-*tp*2-*mbs*2 and *dp*1-*pp*4-*tp*2-*mbs*4). Pipeline parallel execution introduces idle intervals between stages, commonly referred to as pipeline bubbles. Reducing the frequency of stages with such bubbles allows part of the additional latency to be absorbed within these idle intervals, resulting in a smaller increase in end-to-end iteration time (e.g., from point B to E for *dp*2-*pp*2-*tp*2-*mbs*2).[2]

Therefore, the achievable energy savings and performance degradation under frequency scaling depend on the underlying parallelism strategy. Some configurations incur significant runtime penalties under GPU frequency reduction and achieve limited energy savings, while others maintain comparable performance while benefiting from reduced power consumption.

***Insight 2.*** Sensitivity to GPU core frequency scaling depends on the parallelism structure of distributed training. Configurations with pipeline bubbles or communication-induced slack tolerate frequency reduction more effectively, whereas highly compute-saturated configurations experience larger performance degradation.

While optimizing parallelism alone improves both throughput and energy efficiency (§3.2.1), this relationship does not necessarily hold once frequency scaling is introduced. As shown in Figure 5, the throughput-optimal data-parallel configuration (*dp*4-*pp*1-*tp*2-*mbs*1) achieves the lowest runtime and energy at maximum frequency, but becomes less energy-efficient than the pipeline-parallel configuration (*dp*2-*pp*2-*tp*2-*mbs*2) at reduced frequencies beyond point E. This crossover highlights the importance of jointly considering parallelism and GPU frequency when optimizing energy efficiency.

***Insight 3.*** Achieving Pareto-optimal energy-performance trade-offs requires jointly selecting the parallelism configuration and GPU core frequency. In particular, a slower pipeline-parallel configuration combined with frequency down-scaling can consume less total energy than a throughput-optimal data-parallel configuration.

## 4 Roadmap

Our characterization suggests that improving energy efficiency in distributed LLM training requires jointly reasoning about two system knobs: parallelism configuration and GPU operating frequency, as their interaction leads to different performance–energy trade-offs. We outline two directions toward this goal: jointly modeling throughput and energy during configuration planning, and enabling energy-aware scheduling that jointly optimizes parallelism configurations and GPU frequencies.

### 4.1 Joint Modeling of Throughput and Energy

A promising direction is to develop joint models that capture how parallelism configuration and GPU frequency together influence iteration time, device utilization, and power consumption. Such models can enable systems to predict

[2]We note that the drop from point B to the immediately following point is accompanied by a slight runtime decrease; this isolated effect is likely due to measurement variability.

performance–energy trade-offs before deployment and select configurations according to performance targets, energy budgets, or operational constraints.

However, accurately modeling throughput and energy in distributed training remains challenging. First, GPU energy consumption depends not only on computation but also on communication. Training iterations alternate between compute and communication phases, resulting in varying device utilization and power behavior that are difficult to capture. Second, different parallelism strategies exhibit distinct performance and energy sensitivity to GPU frequency scaling, requiring models to capture workload-dependent responses to frequency changes. Finally, temperature variation can further influence power behavior during long-running training jobs, as higher temperatures increase leakage current and consequently raise power consumption [12, 30, 32].

Together, these challenges suggest that accurate prediction of throughput and energy requires holistic analysis of computation–communication energy attribution, workload-dependent sensitivity to GPU frequency scaling, and temperature effects.

### 4.2 Toward Energy-Aware Training Scheduling

Our analysis further suggests opportunities for joint, energy-aware scheduling of parallelism configurations and GPU operating frequencies. Existing systems such as Perseus [6] and EnvPipe [4] leverage pipeline bubbles to safely downscale GPU core frequency for energy savings, but they typically operate under a user-specified parallelization strategy. However, optimizing GPU frequency on a suboptimal parallelization configuration, or a configuration that maximizes throughput without energy-constraint awareness, may yield suboptimal results.

For example, in Figure 5, if a user specifies the $dp2$-$pp2$-$tp2$-$mbs1$ plan (point C), reducing its frequency yields limited energy savings and may miss more energy-efficient operating points available under alternative parallelism configurations. As shown in Figure 5, the Pareto-optimal frontier is achieved by combining $dp4$-$pp1$-$tp2$-$mbs1$ and $dp2$-$pp2$-$tp2$-$mbs2$ under different frequency settings.

A scheduler that jointly considers performance and energy objectives could instead explore both parallelism configurations and GPU operating frequencies to identify Pareto-optimal operating points. Such a scheduler naturally solves constrained optimization problems: depending on system objectives, it may maximize throughput under an energy budget, or minimize total energy subject to throughput or runtime constraints.

For instance, when the objective is to maximize throughput, data-parallel dominant configurations operating at maximum GPU frequency typically provide the best performance. In contrast, when moderate performance degradation is acceptable or energy budgets are tight, pipeline-parallel configurations combined with frequency down-scaling can significantly reduce total energy consumption and operating cost. Designing schedulers that jointly select parallelization strategies and GPU frequencies therefore represents an important direction for future distributed training systems.

## 5 Conclusion

We presented a comprehensive study of energy efficiency in distributed LLM training, examining how parallelization strategies, GPU frequency scaling, and their interaction shape throughput, power, and total energy consumption. Our results show that parallelization strategies differ substantially in their sensitivity to frequency scaling, resulting in distinct performance–energy trade-offs across configurations. These findings highlight the need for coordinated optimization of parallelism and GPU frequency to achieve Pareto-optimal performance and energy efficiency. We further outline a roadmap for energy-aware modeling and scheduling of distributed LLM training. As LLM training continues to scale under tightening power constraints, such cross-layer co-optimization will be essential for building sustainable distributed training systems.

## References


[1] Yuki Abe, Hiroshi Sasaki, Shinpei Kato, Koji Inoue, Masato Edahiro, and Martin Peres. 2014. Power and performance characterization and modeling of GPU-accelerated systems. In *2014 IEEE 28th international parallel and distributed processing symposium*. IEEE, 113–122.

[2] Robert A Bridges, Neena Imam, and Tiffany M Mintz. 2016. Understanding GPU power: A survey of profiling, modeling, and simulation methods. *ACM Computing Surveys (CSUR)* 49, 3 (2016), 1–27.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[4] Sangjin Choi, Inhoe Koo, Jeongseob Ahn, Myeongjae Jeon, and Youngjin Kwon. 2023. {EnvPipe}: Performance-preserving {DNN} training framework for saving energy. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. 851–864.

[5] Esha Choukse, Brijesh Warrier, Scot Heath, Luz Belmont, April Zhao, Hassan Ali Khan, Brian Harry, Matthew Kappel, Russell J Hewett, Kushal Datta, et al. 2025. Power stabilization for AI training datacenters. *arXiv preprint arXiv:2508.14318* (2025).

[6] Jae-Won Chung, Yile Gu, Insu Jang, Luoxi Meng, Nikhil Bansal, and Mosharaf Chowdhury. 2024. Reducing energy bloat in large model training. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*. 144–159.

[7] Google Cloud. 2026. Google Cloud - About GPUs. https://cloud.google.com/compute/docs/gpus/about-gpus.

[8] deepspeedai. 2025. Megatron-DeepSpeed. https://github.com/deepspeedai/Megatron-DeepSpeed.

[9] Shiqing Fan, Yi Rong, Chen Meng, Zongyan Cao, Siyu Wang, Zhen Zheng, Chuan Wu, Guoping Long, Jun Yang, Lixue Xia, et al. 2021. DAPPLE: A pipelined data parallel approach for training large models. In *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. 431–445.

[10] Rong Ge, Ryan Vogt, Jahangir Majumder, Arif Alam, Martin Burtscher, and Ziliang Zong. 2013. Effects of dynamic voltage and frequency scaling on a k20 gpu. In *2013 42nd international conference on parallel processing*. IEEE, 826–833.

[11] Seokjin Go, Joongun Park, Spandan More, Hanjiang Wu, Irene Wang, Aaron Jezghani, Tushar Krishna, and Divya Mahajan. 2025. Characterizing the Efficiency of Distributed Training: A Power, Performance, and Thermal Perspective. In *Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture*. 626–642.

[12] Ricardo Gonzalez, Benjamin M Gordon, and Mark A Horowitz. 1997. Supply and threshold voltage scaling for low power CMOS. *IEEE Journal of Solid-State Circuits* 32, 8 (1997), 1210–1216.

[13] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[14] Hongzhen Huang, Kunming Zhang, Hanlong Liao, Kui Wu, and Guoming Tang. 2025. WattsOnAI: Measuring, Analyzing, and Visualizing Energy and Carbon Footprint of AI Workloads. *arXiv e-prints* (2025), arXiv–2506.

[15] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. 2019. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems* 32 (2019).

[16] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088* (2024).

[17] Seonho Lee, Jihwan Oh, Seokjin Go, and Divya Mahajan. 2025. Characterizing compute-communication overlap in gpu-accelerated distributed deep learning: Performance and power implications. In *2025 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 353–355.

[18] Shaohong Li, Xi Wang, Xiao Zhang, Vasileios Kontorinis, Sreekumar Kodakara, David Lo, and Parthasarathy Ranganathan. 2020. Thunderbolt:{Throughput-Optimized},{Quality-of-Service-Aware} power capping at scale. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. 1241–1255.

[19] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, et al. 2020. Pytorch distributed: Experiences on accelerating data parallel training. *arXiv preprint arXiv:2006.15704* (2020).

[20] Yueying Lisa Li, Omer Graif, and Udit Gupta. 2024. Towards carbon-efficient llm life cycle. In *Proceedings of the 3rd Workshop on Sustainable Computer Systems*.

[21] Guodong Liu, Youshan Miao, Zhiqi Lin, Xiaoxiang Shi, Saeed Maleki, Fan Yang, Yungang Bao, and Sa Wang. 2024. Aceso: Efficient parallel DNN training through iterative bottleneck alleviation. In *Proceedings of the Nineteenth European Conference on Computer Systems*. 163–181.

[22] J McDonald, B Li, N Frey, D Tiwari, V Gadepally, and S Samsi. 2022. Great power, great responsibility: Recommendations for reducing energy for training language models. arXiv 2022. *arXiv preprint arXiv:2205.09646* (2022).

[23] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. 2024. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060* (2024).

[24] NVIDIA. 2026. NVIDIA Data Center GPU Manager (DCGM). https://developer.nvidia.com/dcgm.

[25] NVIDIA. 2026. NVIDIA Management Library (NVML). https://developer.nvidia.com/management-library-nvml.

[26] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. 2022. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*. PMLR, 18332–18346.

[27] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053* (2019).

[28] Jovan Stojkovic, Esha Choukse, Chaojie Zhang, Inigo Goiri, and Josep Torrellas. 2024. Towards greener llms: Bringing energy-efficiency to the forefront of llm inference. *arXiv preprint arXiv:2403.20306* (2024).

[29] Foteini Strati, Zhendong Zhang, George Manos, Ixeia Sánchez Périz, Qinghao Hu, Tiancheng Chen, Berk Buzcu, Song Han, Pamela Delgado, and Ana Klimovic. 2025. Sailor: Automating distributed training over dynamic, heterogeneous, and geo-distributed clusters. In *Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles*. 204–220.

[30] Hameedah Sultan, Shashank Varshney, and Smruti R Sarangi. 2018. Is leakage power a linear function of temperature? *arXiv preprint arXiv:1809.03147* (2018).

[31] Zhenheng Tang, Yuxin Wang, Qiang Wang, and Xiaowen Chu. 2019. 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*. 315–325.

[32] Zibo Wang, Yijia Zhang, Fuchun Wei, Bingqiang Wang, Yanlin Liu, Zhiheng Hu, Jingyi Zhang, Xiaoxin Xu, Jian He, Xiaoliang Wang, et al. 2025. Using analytical performance/power model and fine-grained dvfs to enhance ai accelerator energy efficiency. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*. 1118–1132.

[33] Jie You, Jae-Won Chung, and Mosharaf Chowdhury. 2023. Zeus: Understanding and optimizing {GPU} energy consumption of {DNN} training. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 119–139.

[34] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).

[35] Lianmin Zheng, Zhuohan Li, Hao Zhang, Yonghao Zhuang, Zhifeng Chen, Yanping Huang, Yida Wang, Yuanzhong Xu, Danyang Zhuo, Eric P Xing, et al. 2022. Alpa: Automating inter-and {Intra-Operator} parallelism for distributed deep learning. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. 559–578.

[36] Ruidong Zhu, Ziheng Jiang, Chao Jin, Peng Wu, Cesar A Stuardo, Dongyang Wang, Xinlei Zhang, Huaping Zhou, Haoran Wei, Yang Cheng, et al. 2025. Megascale-infer: Serving mixture-of-experts at scale with disaggregated expert parallelism. *arXiv preprint arXiv:2504.02263* (2025).