Towards Carbon-efficient LLM Life Cycle

Yueying (Lisa) Li
yl3469@cornell.edu
Cornell University
Ithaca, New York, USA

Omer Graif
og88@cornell.edu
Cornell Tech and Technion
New York, New York, USA

Udit Gupta
ugupta@cornell.edu
Cornell Tech
New York, New York, USA

ABSTRACT

With the rise of generative AI, sustainability concerns have intensified due to the computational demands and the need for advanced GPUs. While recent studies have quantified carbon emissions from data centers, a gap exists in fully understanding the lifecycle emissions of generative models and hardware systems.

This paper introduces refined carbon models for CPUs and GPUs, aiming to optimize the design space during the machine learning lifecycle, particularly for multi-GPU systems in generative inference. We present a parameterized embodied carbon model that emphasizes the substantial impact of general-purpose CPUs (2x for lifetime). Our findings suggest model-dependent strategies for carbon-efficient generative inference, such as optimized batching, model sharding, and parallelization. These strategies, combined together appropriately, can achieve a 17% improvement in carbon footprint without negligible degradation in throughput. Additionally, we propose an asymmetric lifetime extension strategy for GPUs to amortize CPU embodied carbon, which enhances energy efficiency despite higher initial carbon costs. This approach highlights the potential for sustainable practices in AI, emphasizing the importance of lifecycle-aware optimization in the era of resource-intensive generative models.

KEYWORDS

Sustainable Computing, Cloud, ML System

ACM Reference Format:
Yueying (Lisa) Li, Omer Graif, and Udit Gupta. 2024. Towards Carbon-efficient LLM Life Cycle. In *Proceedings of 3rd Workshop on Sustainable Computer Systems (HotCarbon'24).* ACM, New York, NY, USA, 8 pages.

1 INTRODUCTION

Data center development has fueled the rapid advancement of technologies like artificial intelligence, which come with high energy costs and corresponding carbon footprints [12, 37, 48]. With the new wave of generative AI, sustainability has become a paramount concern [21, 39, 40]. Due to increasing user traffic and diverse applications, generative models incur high computational costs and require new GPUs with large memory capacities during inference to process long contexts.

To understand the environmental impact of computing platforms, researchers have begun to study the life cycle emissions of computing. Across the life cycle, researchers have identified the majority of emissions are attributed to one of two categories: embodied or operational [24, 50]. *Embodied carbon* refers to the total amount of greenhouse gas emissions (GHG) associated with the production of materials and equipment used in a system [22]. *Operational carbon* owes to the electricity consumed by hyperscalar data centers. To better understand the impact of both embodied and operational carbon, recent efforts have developed tools and methodologies to quantify the end-to-end carbon footprint of computing hardware [23, 24, 27, 31, 32, 40, 42]. Crucially, the research has observed the need to co-optimize across the lifecycle of hardware to balance embodied and operational emissions in data centers.

As generative AI models are rapidly evolving and consuming significant portions of data center infrastructure capacity, there is a dire need to understand the environmental footprint of AI. Furthermore, the high compute, memory, and storage requirements levied by generative models demand specialized solutions to quantify and optimize the embodied and operational carbon from a full-stack, life-cycle-aware characterization and optimization perspective. Recent studies studies [29, 49] have quantified the carbon footprint for LLM inference across different models and hardware systems. However, due to the lack of fine-grained carbon modeling, there remain significant opportunities for model-system-hardware co-design to enable a sustainable ML life cycle.

This paper aims to investigate model-system-hardware co-design opportunities to enable sustainable generative AI inference in hyperscalar datacenters. We begin by proposing an extensible and parameterized model to estimate the operational and embodied carbon footprint of full systems, including general purpose CPUs, memory, storage, and specialized AI engines such as GPU's. Using the model, we identify unique opportunities to optimize end-to-end generative AI systems for both operational and embodied carbon for single- and multi-GPU systems. Central to our insights is the observation that while specialized AI engines account for the lion's hare of power consumption in generative AI, general purpose CPUs incur non-trivial embodied carbon overheads. Based on this observation, we further examine trade-offs for design space optimization during the ML lifecycle, especially for single- or multi-GPU systems used in generative inference.

We summarize the main contributions as follows:

- **Parameterized carbon models for AI platforms**: We develop a parameterized carbon model to estimate the operational and embodied carbon of AI systems. The model demonstrates that lifecycle optimization requires asymmetrically optimizing specialized AI engines (e.g., GPUs) for operational use and host processors (e.g., CPUs) for embodied carbon.
- **Carbon-efficient generative inference strategies**: Different batching, model sharding, and parallelization strategies open performance and carbon efficiency trade-offs, allowing application developers to optimize carbon-efficiency by up to 17%.

HotCarbon'24, July 9, 2024, Santa Cruz, CA

Furthermore, at iso-accuracy, we find sparse Mixture-of-Expert models to be more carbon efficient than dense Llama models by leveraging additional parallelism opportunities [19, 34, 46].

- **Asymmetric lifetime extension strategy:** By refreshing GPUs more frequently than CPUs, we can improve energy efficiency at an upfront embodied carbon cost.

2 MOTIVATION

Table 1: Examples of cloud GPU servers used in Figure 1. (Unit: Memory: GB, TDP: W)

Comp.	GPU	CPU	DRAM	SSD	CPU TDP
AWS	T4x1	Cascade Lake	16	125	205
AWS	T4x4	Cascade Lake	256	900	205
AWS	T4x8	Cascade Lake	384	1800	205
λGPU	A6000$^{\times\{1,2,4\}}$	AMD EPYC 7713	100$^{\times\{1,2,4\}}$	200$^{\times\{1,2,5\}}$	225
Azure	A100$^{\times\{1,2,4\}}$	AMD EPYC 7v12	220$^{\times\{1,2,4\}}$	960$^{\times\{1,2,4\}}$	225
λGPU	A100x8	AMD EPYC 7713	1800	21990	225

With the increasing prevalence of AI, optimizing energy and resource efficiency has become a more prominent research area [33, 51]. Most research has focused on efficient AI algorithms, such as dynamic batch sizes [33, 51] and sparsity [36, 44, 53], as well as system runtime optimization, including scheduling and dynamic voltage frequency scaling [15]. While reducing the computation and memory required for AI training or inference can save operational carbon, attention must also be given to embodied carbon for achieving sustainability.

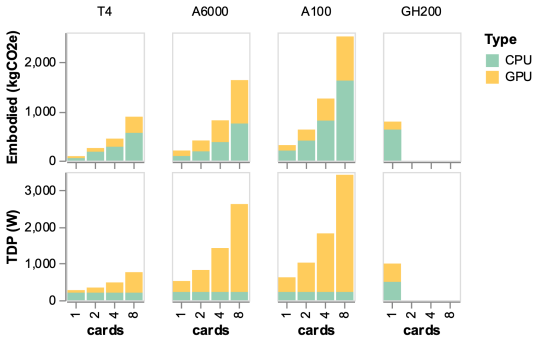

Figure 1: Embodied carbon (top) and Thermal design power (TDP) (bottom) with different cloud GPU offerings sampled from Azure (A100 SXM 80G with 1,2,4 cards), AWS (T4), and LambdaCloud (A6000, A100x8) [1, 3, 47]. [1]

According to Apple [13], the commodity hardware manufacturing (a major part of embodied carbon) accounts for 74% of the total carbon footprint, while the operational use of all its devices contributes 19%. Similarly, for data center AI hardware, it is essential to quantify the full-system and AI life-cycle carbon footprint [49]. However, there is a notable lack of fine-grained carbon models necessary to answer the questions and explore design trade-offs.

[1]T4 instances are usually used for inference, A6000 and A100 are both for inference and training. GH200 with 96 GB HBM3 is an academic training platform for reference. Instances with more GPUs are also equipped with more memory and SSD, increasing embodied carbon. The embodied carbon calculation is shown in the methodology sections.

Furthermore, host CPU processors, which significantly impact the embodied carbon footprint, have not received as much focus.

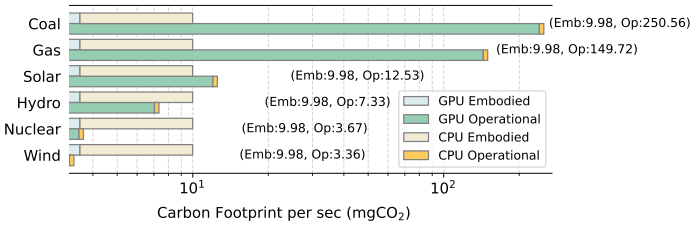

Figure 2: Carbon footprint of A100x4 GPU server per-second LLM inference application when powered by commonly used energy sources in US with different carbon intensity.

In Figure 1, within a typical cloud GPU server, we calculated the CPU and GPU carbon footprint sample from the configuration of the cluster servers and the cloud inference servers. We observed that **CPU dominates embodied carbon (top), whereas GPU dominates thermal design power (bottom)**, which can serve as a proxy for operational carbon assuming a similar utilization profile. The trend is likely to continue given the TDP trends between CPU and GPU in Figure 3 and the clean energy adoption in Figure 2.

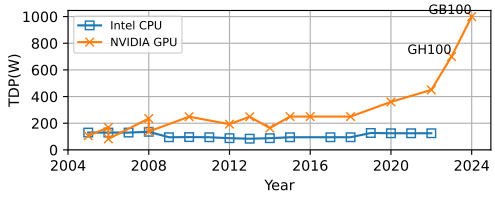

Figure 3: Thermal design power (TDP) trend for CPU and GPU averaged if multiple CPU options are present. CPU is almost constant across generations [45].

The *asymmetric carbon distribution between CPU and GPU* can be attributed to at least two factors:

1) GPUs have become a significant source of operational carbon emissions due to their rapidly increasing thermal design power (TDP) and high idle power consumption. 2) CPUs, along with associated components such as SSDs, main memory, and motherboards, represent a substantial source of embodied carbon emissions, particularly in larger GPU servers. Additionally, the bit density of CPU memory technology is not advancing at the same rate as GPU high-bandwidth memory (HBM). It is because, for cost reasons, old technology nodes have worse carbon efficiency per unit of storage.

These findings underscore the need for distinct optimization strategies for CPUs and GPUs: In regions with low carbon intensity (CI), it is crucial to extend the lifespan of CPUs to reduce the amortized embodied carbon footprint. Conversely, in high-CI regions, the focus should be on enhancing GPU utilization or decreasing the time and energy required to complete equivalent workloads. Furthermore, implementing different hardware refresh cycles for CPUs and GPUs is necessary. This approach, combined with more granular recycling practices for machine learning inference servers, can help minimize the overall carbon impact of these systems.

3 METHODOLOGY

In this section, we first discussed the limitations of the existing work and then proposed our carbon modeling methods.

3.1 Limitation of Past Modeling Techniques

Life Cycle Assessment (LCA) is a tool for global and multi-criteria evaluation of environmental impacts [8, 9]. This standardized method makes it possible to measure the quantifiable effects of products or services on the environment. However, for AI inference workloads, it is more important to understand the implications of different hardware (GPU, TPU, CPU) involved in carbon footprint (CF). There are many past efforts [23, 29] which try to come up with models and methodologies to address the lack of supply chain carbon footprint data, however, none are fine-grained enough to account for the different configurations under multi-GPU setup.

Besides, there is a void in fine-grained modeling of different memory technology nodes for GPUs, which significantly impacts the embodied carbon footprint. Last but not least, the peripheral components for cooling or power delivery are also not considered in the past models, which is essential for higher-end GPUs. For example, in ACT, GPU embodied carbon being modeled is only 78% of all the embodied carbon components [22, 29].

3.2 Embodied Carbon Modeling

$$CF^{emb,cpu} = \frac{T}{LT}(N_r K_r + \sum_{k}^{\text{DRAM, PWB, PDN, Fan, Storage, Die,Chasis}} CF_k)$$

$$CF^{emb,gpu} = \frac{T}{LT}(N_r K_r + \sum_{k}^{\text{SoC,PCB,VRAM,PDN,PCB,Con.,Cooling}} CF_k)$$

The embodied carbon footprint (CF) is attributed to a combination of components such as DRAM, PCB, SSD, and the CPU die itself. For GPU, it can be attributed to SoC, Power Delivery Network (PDN), Cooling (Heat sink), etc. We take the inference servers' specs in Table 1 and decompose the analysis into different components in the following section (Figure 4): Here $N_r K_r$ denotes packing carbon footprint, ad LT shows the lifetime, which we take as 3-5 years. T is the execution time of the task.

3.2.1 SSD. We first study the LCA report [10, 13, 17, 25]. We base our analysis on the Dell R740 Report. The Dell PowerEdge R740 is a server that integrates accelerator cards, storage and computational resources in a 2U, 2-socket platform (i.e. 2 rack units). In our calculation, we get the per GB SSD carbon footprint and scale it according to the platform specs:

$$CF_{SSD/GB} = \frac{3378.944}{8 * 3.84 * 1000} = 0.10999\ (kgCO_2e/GB)$$

$$\text{Total Storage CF} = 0.10999 SSD_{GB}$$

3.2.2 Mainboard Printed Circuit Board (PCB). The mainboard in the LCA report was estimated to be 1925 square centimeters in area, and we scaled the numbers accordingly. In our calculation, we used the breakdown but scaled the numbers according to different PCB sizes, etc. In SCARIF [29], the numbers are not scaled, so it results in a very large CPU carbon footprint.

$$\text{Mainboard CF} = 175.8\ (kgCO_2e)\ \text{PCB area} = 1925\ (cm^2)$$

$$CF_{PCB/cm^2} = 175.8 * 62\% * 0.85/1925 = 0.048\ (kgCO_2e/cm^2)$$

$$\text{Total PCB CF} = 0.048 \times Area_{PCB}$$

3.2.3 Die. We use the ACT and iMeC tool [20, 23] and the area of the chip and process node to approximate the CF_{Die}.

3.2.4 Peripheral Printed Wiring Board (PWB) Components. It is essential to quantify the peripheral PWB's carbon footprint (without RAM, including PCB, HDD controller, riser card, etc.). We calculate the CF numbers based on the area scaling to different components of the PWB. The detailed breakdown of selected components is shown below:

$$\text{Ethernet card CF} = 102.3 * 0.048 = 4.91\ (kgCO_2e)$$

$$\text{HDD Controller} = 107.0 * 0.048 = 5.136\ (kgCO_2e)$$

3.2.5 Memory. The methodology for calculating DRAM / VRAM CF is based on the different technology nodes' bit densities, and combined with the emission given a certain wafer area and its manufacturing technology (assuming constant yield) [30].

An example calculation for VRAM of GH200 (assuming yield as 0.9, Wafer CF for HBM3e as 700 from TechInsight Analysis [30]): To project the bit density for newer GPUs like the HBM3e used in GH200 not shown in the datasheet, we extrapolate its memory technology 1β as 1α / scaling factor = $0.315/0.88^2 = 0.4$.

$$\text{Wafer\_Mem} = \text{Size} \times \frac{\text{Bit Density}}{8} = 3675.62\ (GB)$$

$$CF_{HBM3e/GB} = \frac{\text{Wafer CF}}{\text{Yield} \times \text{Wafer\_Mem}} = 0.24\ (kgCO_2e/GB)$$

$$\text{Total RAM CF} = CF_{HBM3e/GB} \times \text{Mem}$$

Below, we give the formula for calculating general memory carbon footprints using bit density summarized from various company reports. We show the final memory carbon footprint per GB in Figure 5. The equation below shows some common numbers for GPU offerings with AMD ECPY CPUs: T4, A100, and GH200[2].

$$CF_{Mem} = \begin{cases} 0.24 \times \text{Mem} & \text{if using HBM3e,3,2e, } 1\beta \\ 0.28 \times \text{Mem} & \text{if using HBM2, } 1\alpha \\ 0.29 \times \text{Mem} & \text{if using DDR4/LPDDR5, } 1z \\ 0.62 \times \text{Mem} & \text{if using GDDR6, } 2y \end{cases}$$

3.2.6 Power Delivery Network (PDN). We scale the PDN carbon footprint from the LCA report with board-level TDP of the system of interest. For example, for A6000 GPU system,

$$CF_{PDN} = \frac{\text{Design\_TDP\_GPU}}{\text{LCA\_TDP}} \times \text{LCA\_PDN\_CF} = 22.89\ (kgCO_2e)$$

3.2.7 Fans / Cooling. We scale the cooling carbon footprint with chip-level TDP. For example, for A6000

$$CF_{Cooling} = \frac{\text{Design\_TDP\_GPU}}{\text{LCA\_TDP}} \times \text{LCA\_HeatSink\_CF} = 23.63\ (kgCO_2e)$$

[2]We focus on DDR4/LPDDR5 CPUs after 2016.

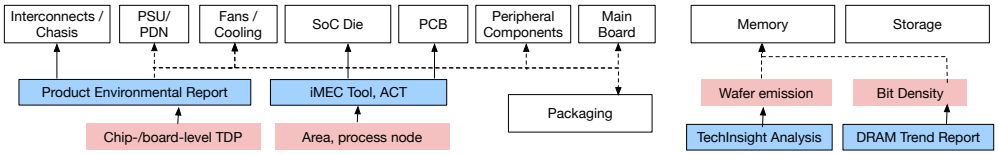

Figure 4: Proposed carbon modeling framework with more fine-grained embodied carbon estimation on memory, storage and power related components.

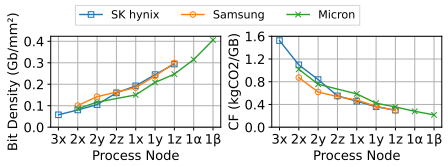

Figure 5: Left: Bit density trends for different DRAM technology nodes summarized from company reports. The 1β data is extrapolated. Right: Carbon footprint per GB trends.

3.3 Operational Carbon Modeling

There are several ways to quantify the operational carbon footprint, including using runtime energy measurements for different components and scaling them with the infrastructure's carbon intensity and PUE or making estimates based on hardware utilization counters accessible on the cloud.

The operational carbon comes from both the memory and computing of the GPU and CPU. Moreover, the power related to the memory controller, network device, and network components is nontrivial even if the device is in an idle state. Extensive research has been conducted on power modeling across various scales—from data centers and servers to virtual machines and applications. Notably, Leila Ismail's study provides a comparative evaluation of software-based power models for data center servers, exploring both linear and non-linear approaches that utilize mathematical or machine learning methods [28].

We take a hybrid method for estimating the operation carbon. For servers with direct RAPL [16] access, we use energy measurement tools such as Zeus [51]. Otherwise, we use STRESS-NG to add different CPU and memory (VM) loads to the system and use PSUTIL and INTEL-RAPL to measure the instant power and other hardware counters like utilization [4, 5]. Since there is no easy way to add a linearly scaled GPU power stressor, we focus on characterizing the power with respect to the different CPUs and try to extrapolate common principles.

One critical insight from these studies is that power models relying solely on CPU utilization to estimate server power consumption are prone to high error rates. More accurate models also incorporate memory usage. Additionally, incorporating a constant to account for other significant components, such as storage and network, could enhance model accuracy.

After we collect data with different stressors (as shown in Figure 6, for CPU stressor), we can fit the related constants in Equation 1. Here MFU is the model flops utilization.

$$
\begin{aligned}
CF^{op} &= E \times CI_{use} \quad \text{or } CI_{use} \times P \times t \\
&= CI_{use} t \times [(\alpha Util^{c} + \beta Util^{mem} + P^{c}_{idle}) \\
&\quad + (aUtil^{g}(\gamma + c\text{MFU}) + bUtil^{vram} + P^{g}_{idle})] \qquad (1)
\end{aligned}
$$

where we choose

$$
CI_{use} = \begin{cases} 35\ (gCO_2e/kWh) & \text{if renewable energy} \\ 380\ (gCO_2e/kWh) & \text{otherwise} \end{cases}
$$

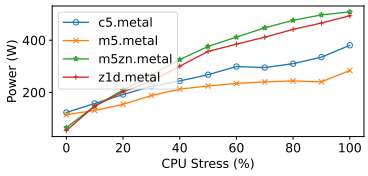

Figure 6: In order to fit the constants for the equation and estimate the power consumption of cloud instances, we not only measure the instance power with different CPU utilization [11] but also measure the power with different memory utilization.

4 CHARACTERIZATION AND ANALYSIS

In Figure 7, we visualize the CPU and GPU embodied carbon components (only manufacturing-related carbon) for the following two types of servers used for AI training and inference: Standard_NC96ads_A100_v4 with 2 A100 GPUs and 80 GB HBM2e and Standard_NC4as_T4_v3 T4 inference server with 16GB GDDR6.

For inference platforms like T4, much of the carbon footprint is on the CPU's mainboard (top row). This motivates us to recycle and reuse the CPU components better.

The percentage breakdown (Figure 7) shows that the storage and DRAM is a huge carbon sink for the A100 platforms' CPUs. A noticeable difference is in the storage for T4. Since the instance allocates a very small amount of storage, the carbon footprint is not a huge concern.

For the A100 GPU platform, the cooling part's carbon footprint is higher due to the higher TDP and higher vertical density of the 3D memory technology with HBM2e [6, 7]. This motivates us to think about design and runtime techniques for improving the power proportionality of the memory sub-system.

5 TOWARDS CARBON-EFFICIENT ML SERVING

In this section, we discuss and analyze several techniques used to optimize system throughput and efficiency for LM serving and

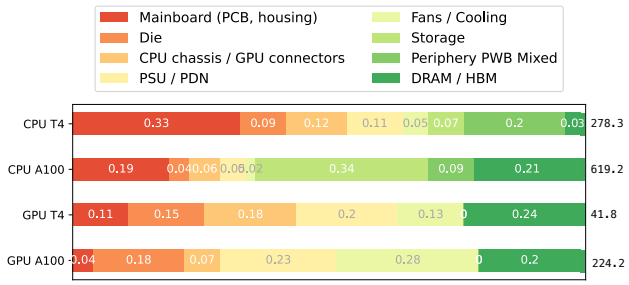

Figure 7: Azure T4 and A100 GPU carbon footprint breakdown (absolute values in kgCO2e for the total lifetime on the right. Cooling and PDN compose a big part for A100 GPU, and Mainboard and storage is a big part of CPU carbon.).

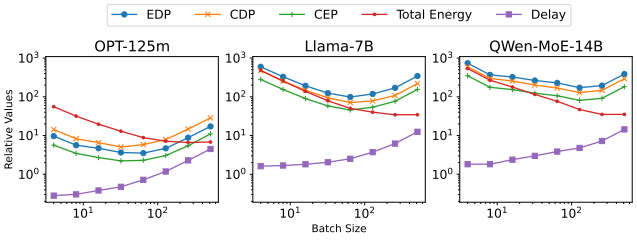

Figure 8: EDP (Energy-delay-product), CDP (Carbon-delay-product), CEP (Carbon-energy-product) and traditional metrics like total energy (per task) and delay vs. Batch (8-512): different models have different optimal batching points. For smaller models, optimizing for EDP, CDP gives a relatively small batch size, and the optimal batch size grows larger as the model becomes more complex.

demonstrate the tradeoffs and large design spaces when also optimizing for carbon efficiency.

5.1 Batching for Carbon-aware Offline Inference

For best-effort generative offline inference, batching techniques are often used to increase the utilization of the GPU system [14, 33, 52]. The larger the batch size, the higher the efficiency or utilization (until it is bandwidth-bounded by the roofline model [38]). We will start to examine the same design question from a holistic carbon perspective: *What is the optimal number of requests in a batch?*

We use the vLLM for inference and Zeus for energy measurement [2] and study the EDP (Energy-delay-product), CDP (Carbon-delay-product) and CEP (Carbon-energy-product) for each request (task) with different batching strategies for various models. The lower the number, the more carbon-friendly the system is. We fix the workload generation length and prompt length as the default in the vLLM offline inference benchmark and aggregate the results over three random seeds.

Observation: In Figure 8, the optimal CDP/CEP/EDP is achieved at different batch sizes (8-512) for different models[3]. This is because of the high idle power draw for GPUs; the larger the batch size, the better the compute and energy efficiency until it reaches the memory-bound region of the roofline model. Optimizing traditional metrics like EDP doesn't equate to optimizing for CEP. However, since embodied carbon is proportional to the delay, the curve is convex for CEP. **Implication:** It is important to select the right batch size to optimize for carbon efficiency, alongside latency and throughput goals.

5.2 Choice of Model Parallelism

There are at least three types of model parallelism: model tensor parallelism, expert parallelism, and model pipeline parallelism, in addition to data parallelism. Tensor parallelism (TP) is used for parallelizing computations within a tensor, while pipeline parallelism is used for computations between layers. Expert parallelism (EP) applies to Mixture-of-expert (MoE) models - breaking large feed-forward layers computation (FFN) into smaller ones. For large language models containing large-scale matrix multiplications, TP and EP can help reduce the memory footprint per GPU and reduce latency. However, it also introduces new challenges, such as synchronization and communication overhead. We ask: *Does the model or batch size play a role in the choice of parallelism strategy for carbon- or energy-efficient inference?*

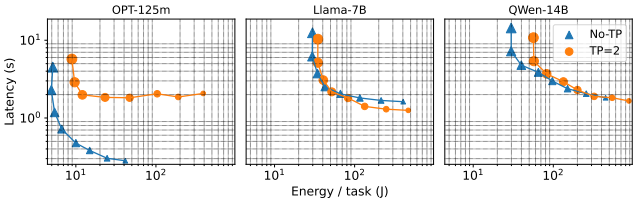

Figure 9: Latency vs Energy with different batch sizes under the same setup as in vLLM [33] for offline inference. The size of the circles/triangles is proportional to the log of batch size (4-512).

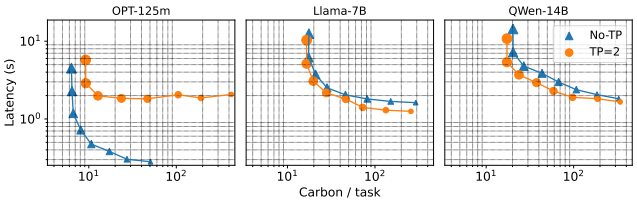

Figure 10: Latency vs Embodied Carbon. TP is better for MoE and Llama from a Carbon perspective because TP amortized the high CPU carbon footprint.

Observation 1: In Figure 9, we used both parallelization (TP=2) and a non-parallelized deployment (No-TP) strategy for various models. MoE architecture usually requires more model weight parameters but fewer activated parameters compared with iso-accuracy dense models [41]. It's worth noting that QWen-MoE [46]

[3]We scale the CEP by a common factor for all models and systems for easier representation.

has 2.7B active parameters during inference for batch size equaling 1; however, as the batch size grows, the number of activated experts also increases for a single batch, increasing the delay for communication due to load imbalance. QWen has a better model quality than Llama-7B but has a comparative energy and carbon profile, especially under larger batch sizes for offline inference. Hence, MoE architecture is more carbon-friendly under iso-accuracy model variants.

Observation 2: From Figure 9 and 10 left, for smaller models, No-TP is preferred at all batch sizes for energy/carbon-efficiency and latency. This is because the compute intensity for GEMM operations is small, and the overhead of communication overshadows the benefit of parallelization. However, there is a trade-off for larger models like Llama and MoE (middle and right). TP is preferred for latency, although the win is small (under 8% most of the time); No-TP is preferred for energy, and the gain is large (55% at best). TP is also preferred for embodied carbon (17% at best). Because MoE architecture's load imbalance issue can be partly mitigated with the sub-expert partition, and the bigger the batch size, the more carbon efficiency gain or latency improvement TP can bring. We also ran JetMoE [43] to consider almost iso-accuracy with Llama, and the carbon efficiency gain is 19%.

Implication 1: It's more carbon-friendly to use MoE compared with dense models. It's also more embodied-carbon-friendly to use TP under larger models, although operational carbon or energy can be worse. **Implication 2**: For datacenter regions with cleaner energy sources, TP is preferred for larger models for carbon, whereas non-TP is preferred for regions with coal supply (Figure 2).

5.3 Optimize GPU Operational Carbon with Model Sharding for Online Serving

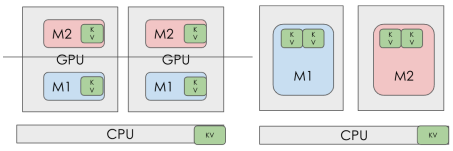

Figure 11: Illustration on model parallel decisions. Two models can either be sharded horizontally (A) or vertically (B) across two GPUs. Left is preferred for carbon.

Model sharding (or statistical multiplexing) is a common technique to improve the utilization or latency of multi-GPU serving systems [35]. For example, we can multiplex two models on two GPUs with the same CPU host (Figure 11). Since CPUs are mostly used for request scheduling, tokenization, and continuous batching, it can help reduce the embodied carbon when sharing CPUs for different models.

Table 2: Comparison for model sharding strategies

TP	Lat (ms)	P^C (W)	P^G (W)	Energy (J)	CF^{op}	CF^{emb}
No (B)	111.23	129.02	240.5	41102.53	4.206	1.11
Yes (A)	127.12	136.75	167.34	29907.643	3.057	1.14

In an online serving setting, we generate requests based on the ShareGPT dataset to two different models (M1, M2 as Mistral in Figure 11). The QPS is set to the same as 2, 4, and 8 requests/second at Poisson arrival. We study the different placement strategies for models sharing the same CPU (Table 2). The operational carbons are 3.057 and 4.206 (gCO2e), respectively, on average for full inference on 200 requests. It is worth noting that (A) has a longer TPOT (time per output token) (degradation by 14%), but improves the operational carbon by around 30%. We defer how the burstiness of the workloads and model multiplexing beyond TP=2 impact the optimal parallelism strategy for future work.

Observation: Tensor parallelism (A) improves energy efficiency at a slight throughput cost (due to extra communication overhead). **Implication:** We can optimize for energy by interleaving models together on different GPUs horizontally rather than vertically. The exact sweet spot would depend on hardware configuration (bandwidth, peak FLOPS) and model characteristics.

5.4 Amortize CPU Embodied Carbon

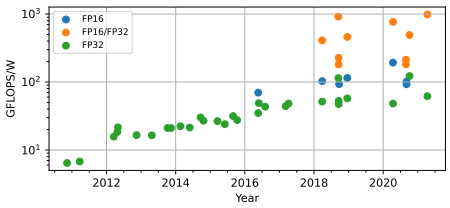

Figure 12: GFlops per Watt trend on different GPU and floating point formats. Energy efficiency doubles almost every 3 years [18].

For estimation of the AI model's operational carbon scaling with different hardware updates, we refer to the trend for GFLOPS/Watt of GPUs across generations (Figure 12). Analyzing a dataset of 470 GPUs from 2006–2021 provided by Sun et al, it can be estimated that FLOPS per watt doubles every 3–4 years on average. This is a slower rate than the 2-year doubling time of Moore's law [26, 45]. It means that for the same amount of work for AI inference, the operational carbon footprint will be reduced by half every 3-4 years if data center providers constantly refresh their GPU hardware.

Implication: Given the asymmetric improvement in GPU and CPU power efficiency and DRAM density, we can use CPUs with fewer cores, put CPUs to a lower frequency state, or extend the life of CPUs and SSDs. On the other hand, for GPUs, we should replace them based on the operation energy efficiency requirements and the embodied carbon footprint accounting models.

6 SUMMARY AND FUTURE DIRECTIONS

Our work offers a framework for quantifying carbon footprints in AI systems and hardware. By integrating operational and embodied emissions, we enable optimization of AI sustainability. This work lays the foundation for carbon-aware models, system and hardware co-design, and eco-conscious hardware refresh strategies. We call on the research community to build upon this framework, advancing the frontier of sustainable AI through holistic, quantitative approaches to environmental impact assessment and mitigation.

ACKNOWLEDGMENTS

We sincerely thank Leo Han, Nikita Lazarev, Michael Shen, Zhanqiu Hu, and the anonymous reviewers for their valuable feedback. We would also like to acknowledge the CloudLab (NSF 1419199) and NSF ACCESS Discovery (CIS230253) infrastructure support for providing resources to conduct these experiments. This work is also in part supported by NSF Grant CNS-2335795, and SRC ACE Jump Center.

REFERENCES

[1] AWS and NVIDIA — aws.amazon.com. https://aws.amazon.com/nvidia/. [Accessed 05-05-2024].
[2] GitHub - ml-energy/zeus: Deep Learning Energy Measurement and Optimization — github.com. https://github.com/ml-energy/zeus. [Accessed 08-05-2024].
[3] GPU Cloud - VMs for Deep Learning | Lambda — lambdalabs.com. https://lambdalabs.com/service/gpu-cloud. [Accessed 05-05-2024].
[4] Power management - ArchWiki — wiki.archlinux.org. https://wiki.archlinux.org/title/Power_management. [Accessed 07-05-2024].
[5] psutil documentation. https://psutil.readthedocs.io/. [Accessed 07-05-2024].
[6] SK hynix Newsroom — news.skhynix.com. https://news.skhynix.com/hbm2e-opens-the-era-of-ultra-speed-memory-semiconductors/. [Accessed 05-05-2024].
[7] The Ultimate Guide to HBM2E Implementation & Selection — rambus.com. https://www.rambus.com/blogs/hbm2e/#hbm2e_vs_gddr6. [Accessed 05-05-2024].
[8] Eio-lca. [Available Online] http://www.eiolca.net//, 2008.
[9] Product carbon footprints. [Available Online] https://www.hpe.com/psnow/doc/a00104415enw, 2019.
[10] Product carbon footprints. [Available Online] https://www.dell.com/en-us/dt/corporate/social-impact/advancing-sustainability/climate-action/product-carbon-footprints.htm#tab0=3, 2023.
[11] Amazon Web Services. Delivering progress every day : Amazon's 2021 sustainability report, 2021.
[12] Thomas Anderson, Adam Belay, Mosharaf Chowdhury, Asaf Cidon, and Irene Zhang. Treehouse: A case for carbon-aware datacenter software. *SIGENERGY Energy Inform. Rev.*, 3(3):64–70, oct 2023.
[13] Apple, Inc. Product environmental reports. [Available Online] https://www.apple.com/in/environment/, 2023.
[14] Lequn Chen, Weixin Deng, Anirudh Canumalla, Yu Xin, Danyang Zhuo, Matthai Philipose, and Arvind Krishnamurthy. Symphony: Optimized dnn model serving using deferred batch scheduling, 2023.
[15] Sangjin Choi, Inhoe Koo, Jeongseob Ahn, Myeongjae Jeon, and Youngjin Kwon. {EnvPipe}: Performance-preserving {DNN} training framework for saving energy. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, pages 851–864, 2023.
[16] Howard David, Eugene Gorbatov, Ulf R. Hanebutte, Rahul Khanna, and Christian Le. Rapl: memory power estimation and capping. In *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '10, page 189–194, New York, NY, USA, 2010. Association for Computing Machinery.
[17] Dell, Inc. Product environmental reports. [Available Online] https://www.dell.com/en-us/dt/corporate/social-impact/advancing-sustainability/climate-action/product-carbon-footprints.htm, 2023.
[18] Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. Trends in ai inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustainable Computing: Informatics and Systems*, 38:100857, April 2023.
[19] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.
[20] M. Garcia Bardon, P. Wuytens, L.-Å. Ragnarsson, G. Mirabelli, D. Jang, G. Willems, A. Mallik, A. Spessot, J. Ryckaert, and B. Parvais. Dtco including sustainability: Power-performance-area-cost-environmental score (ppace) analysis for logic technologies. In *2020 IEEE International Electron Devices Meeting (IEDM)*, pages 41.4.1–41.4.4, 2020.
[21] Google. Carbon free energy for google cloud regions, 2022.
[22] Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S. Lee, David Brooks, and Carole-Jean Wu. ACT: designing sustainable computer systems with an architectural carbon modeling tool. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, pages 784–799, New York, NY, USA, June 2022. Association for Computing Machinery.
[23] Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S. Lee, David Brooks, and Carole-Jean Wu. Act: Designing sustainable computer systems with an architectural carbon modeling tool. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, page 784–799. Association for Computing Machinery, 2022.
[24] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S. Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. Chasing Carbon: The Elusive Environmental Footprint of Computing. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 854–867, February 2021. ISSN: 2378-203X.
[25] Hewlett Packard Enterprise (HPE), Inc. Product environmental reports. [Available Online] https://www.hpe.com/psnow/doc/a50002945enw.pdf?jumpid=in_pdfviewer-psnow, 2023.
[26] Marius Hobbhahn. Trends in GPU Price-Performance — epochai.org. https://epochai.org/blog/trends-in-gpu-price-performance. [Accessed 07-05-2024].
[27] International Telecommunication Union. Greenhouse gas emissions trajectories for the information and communication technology sector compatible with the UNFCCC Paris agreement: L. 1470, 2020.
[28] Leila Ismail and Huned Materwala. Computing server power modeling in a data center: Survey, taxonomy, and performance evaluation. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.
[29] Shixin Ji, Zhuoping Yang, Xingzhen Chen, Jingtong Hu, Yiyu Shi, Alex K. Jones, and Peipei Zhou. Towards Data-center Level Carbon Modeling and Optimization for Deep Learning Inference, March 2024. arXiv:2403.04976 [cs].
[30] Scotten W. Jones. Modeling 300mm wafer fab carbon emissions. In *2023 International Electron Devices Meeting (IEDM)*, pages 1–4, 2023.
[31] Donald Kline, Nikolas Parshook, Xiaoyu Ge, Erik Brunvand, Rami Melhem, Panos K. Chrysanthis, and Alex K. Jones. GreenChip: A tool for evaluating holistic sustainability of modern computing systems. *Sustainable Computing: Informatics and Systems*, 22:322–332, June 2019.
[32] Donald Kline, Nikolas Parshook, Xiaoyu Ge, Erik Brunvand, Rami Melhem, Panos K. Chrysanthis, and Alex K. Jones. Greenchip: A tool for evaluating holistic sustainability of modern computing systems. *Sustainable Computing: Informatics and Systems*, 22:322–332, 2019.
[33] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023.
[34] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.
[35] Zhuohan Li, Lianmin Zheng, Yinmin Zhong, Vincent Liu, Ying Sheng, Xin Jin, Yanping Huang, Zhifeng Chen, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. AlpaServe: Statistical multiplexing with model parallelism for deep learning serving. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*, pages 663–679, Boston, MA, July 2023. USENIX Association.
[36] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and Beidi Chen. Deja vu: Contextual sparsity for efficient LLMs at inference time. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 22137–22176. PMLR, 23–29 Jul 2023.
[37] Jialun Lyu, Jaylen Wang, Kali Frost, Chaojie Zhang, Celine Irvene, Esha Choukse, Rodrigo Fonseca, Ricardo Bianchini, Fiodar Kazhamiaka, and Daniel S. Berger. Myths and misconceptions around reducing carbon embedded in cloud platforms. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems*, HotCarbon '23, New York, NY, USA, 2023. Association for Computing Machinery.
[38] NERSC. Roofline Performance Model - NERSC Documentation — docs.nersc.gov. https://docs.nersc.gov/tools/performance/roofline/. [Accessed 08-05-2024].
[39] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. The carbon footprint of machine learning training will plateau, then shrink. *arXiv preprint arXiv:2204.05149*, 2022.
[40] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.
[41] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale, 2022.
[42] Victor Schmidt, Kamal Goyal, Aditya Joshi, Boris Feld, Liam Conell, Nikolas Laskaris, Doug Blank, Jonathan Wilson, Sorelle Friedler, and Sasha Luccioni. Codecarbon: Estimate and track carbon emissions from machine learning computing, 2021.
[43] Yikang Shen, Zhen Guo, Tianle Cai, and Zengyi Qin. Jetmoe: Reaching llama2 performance with 0.1 m dollars. *arXiv preprint arXiv:2404.07413*, 2024.
[44] Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. Powerinfer: Fast large language model serving with a consumer-grade gpu. *arXiv preprint arXiv:2312.12456*, 2023.
[45] Yifan Sun, Nicolas Bohm Agostini, Shi Dong, and David Kaeli. Summarizing cpu and gpu design trends with product data. *arXiv preprint arXiv:1911.11313*, 2019.

[46] Qwen Team. Qwen1.5-MoE: Matching 7B Model Performance with 1/3 Activated Parameters — qwenlm.github.io. https://qwenlm.github.io/blog/qwen-moe/. [Accessed 08-05-2024].

[47] vikancha MSFT. Azure VM sizes - GPU - Azure Virtual Machines — learn.microsoft.com. https://learn.microsoft.com/en-us/azure/virtual-machines/sizes-gpu. [Accessed 05-05-2024].

[48] Jaylen Wang, Udit Gupta, and Akshitha Sriraman. Peeling back the carbon curtain: Carbon optimization challenges in cloud computing. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems*, HotCarbon '23, 2023.

[49] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems*, 4:795–813, 2022.

[50] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga Behram, James Huang, Charles Bai, et al. Sustainable AI: Environmental implications, challenges and opportunities. *arXiv preprint arXiv:2111.00364*, 2021.

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

[52] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for Transformer-Based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 521–538, Carlsbad, CA, July 2022. USENIX Association.

[53] Youpeng Zhao, Di Wu, and Jun Wang. Alisa: Accelerating large language model inference via sparsity-aware kv caching. *arXiv preprint arXiv:2403.17312*, 2024.