Store or Recompute? Characterizing the Carbon Tradeoff of KV Cache Retention in LLM Inference

Abnash Bassi Jaylen Wang Fiodar Kazhamiaka† Daniel S. Berger† Akshitha Sriraman
Carnegie Mellon University †Microsoft
abnash@cmu.edu, jaylenw@andrew.cmu.edu, akshitha@cmu.edu

***Abstract*—The growing compute demand of Large Language Model (LLM) inference carries a significant environmental cost. LLM multi-turn conversations can grow to have large contexts, span multiple weeks, with sporadic session liveness driven by user habits. To reduce GPU load, KV caches can be saved in non-volatile storage servers and retrieved when the session becomes active. This setup presents a fundamental tradeoff between high power (and associated operational carbon emissions) of recomputing the KV cache and the high capacity requirements (and associated embodied carbon emissions) of storing the KV cache. We perform the first characterization of this tradeoff. We present a first-order analytical model to identify break-even points between this decision of storing vs. recomputing the KV cache as a function of storage duration, comparing the embodied carbon of storage hardware against the operational carbon of GPU-based recomputation. We further examine how response time requirements interact with carbon-saving decisions, revealing scenarios where carbon-efficient strategies may conflict with performance needs. Our results demonstrate that the break-even duration varies significantly with different system parameters (e.g., the carbon intensity of the energy source).**

***Index Terms*—KV cache, LLM, data center, carbon footprint, AI**

I. Introduction

Large Language Model (LLM) inference is being deployed at scale, driving a massive demand for computation [1]. This growing compute demand incurs significant carbon emissions [2]–[4].

Today, inference systems generate and store intermediate key-value (KV) caches [5], [6], to reduce redundant computation across queries. Upon processing a follow-up query, e.g., in a multi-turn conversation, the system can reuse this KV-cache rather than recomputing it from scratch. However, due to the KV cache's large memory footprint [6], [7] and the potentially long duration between reuse, inference systems face a retention decision: store the KV cache on persistent storage (e.g., an SSD) for future reuse, or discard it and recompute it when the next query arrives.

Between storing vs. recomputing the KV cache, we identify a critical carbon tradeoff. Storing KV caches on disk for long durations increases *embodied emissions* because provisioning sufficient storage capacity requires manufacturing additional storage hardware. Recomputing KV caches avoids these embodied emissions but increases *operational emissions* due to GPUs' high power draw during the recomputation step. The carbon-saving choice depends on the *reuse distance*—the time between when a KV cache is first computed and when a follow-up query requires it. Whereas prior work [8], [9] has examined how to manage KV cache storage capacity and hierarchy for performance, this KV cache retention decision's carbon impact remains largely under-characterized.

To this end, we perform the first holistic characterization of the carbon tradeoff between KV cache storage and recomputation. To enable this study, we present a first-order analytical model that identifies the break-even points between the two strategies as a function of realistic LLM inference system parameters: KV cache size, reuse distance, GPU device, storage technology, and grid carbon intensity. Our model compares the embodied carbon of provisioning storage hardware with the operational carbon of GPU-based recomputation.

Beyond carbon analysis, we examine how performance requirements interact with carbon-saving decisions. Specifically, we analyze the tension between minimizing response time and minimizing carbon. Our analysis reveals scenarios where carbon-efficient strategies conflict with meeting latency Service Level Objectives (SLOs). For example, we find that retrieving a KV cache from storage to save operational carbon may violate a real-time application's latency constraint. Thus, we reveal scenarios where looser latency constraints, e.g., a user being willing to wait longer for a response, can be leveraged for lower added carbon emissions.

Our results show that the break-even duration, and thus the storage-versus-recompute tradeoff, varies significantly based on the KV cache size. We also find that for low carbon intensities (e.g., during the day when more renewables are available), storing the cache is carbon-saving after 3-4 minutes, whereas for higher carbon intensities, recomputation remains superior for 7 minutes. Additionally, depending on the LLM and hardware configuration used, the ideal break-even duration determined from our model can be on the order of hours.

These findings suggest that static policies are insufficient, creating an opportunity for carbon-aware retention policies that dynamically select between storage and recomputation. Such a system would need to predict "reuse distance," i.e., the likelihood of a follow-up request and the duration until the request, and make a decision accordingly of whether to store or to recompute a KV cache.

In summary, we contribute:

- A modeling methodology that quantifies the carbon footprint of generating LLM KV caches via (1) GPU recomputation and (2) retrieval from persistent storage, accounting for both operational and embodied emissions.

- The first characterization of how the recompute-versus-store carbon tradeoff changes with reuse distance and grid carbon intensity.
- An identification of break-even reuse distances across workload parameters.
- A first-order analysis of carbon impact and latency to surface carbon-performance tradeoffs.
- A case for designing inference systems that dynamically navigate this tradeoff space to enable sustainable LLM serving.

II. Motivation

This section provides background on KV caching in LLM inference (§II-A), the carbon emissions associated with AI infrastructure (§II-B), and the tradeoff between storing and recomputing KV caches (§II-C).

A. KV Caching in LLM Inference

LLMs process input tokens through a stack of attention layers, each of which computes key and value vectors for every token in the sequence [10]. During autoregressive generation, each new output token attends to the key and value vectors of all preceding tokens. Recomputing these vectors from scratch at each generation step wastes computation that grows linearly with the number of input tokens. To avoid this redundancy, inference systems store the key and value vectors in a *KV cache* that persists across generation steps within a single request [5].

In addition, the KV cache also enables reuse *across* requests. When multiple queries share a common prefix, such as a system prompt or a multi-turn conversation history, the system can compute the prefix's KV cache once and reuse it for all subsequent queries that share the prefix [9]. Such reuse eliminates redundant computation, reducing both latency and GPU utilization.

However, KV caches can be large; for modern models with tens or hundreds of billions of parameters, a single request's KV cache can consume gigabytes of GPU memory [6], [7]. As a result, GPU memory alone cannot retain KV caches for all potentially reusable prefixes; and storing the KV cache of even one request temporarily can be extremely costly [11].

To efficiently manage KV cache capacity, recent work [8], [12] has proposed hierarchical storage systems that manage where to store KV caches, i.e., from GPU memory to host DRAM and then to SSDs. These systems treat the storage hierarchy as a performance optimization: they evict caches to slower tiers when GPU memory is full and retrieve them when a matching prefix arrives. However, these systems do not consider how each tier of the hierarchy differs in environmental cost and under what conditions one tier is more environmentally costly than another. Storing a KV cache on an SSD avoids recomputation but requires provisioning storage hardware whose manufacturing carries an embodied carbon cost. Recomputing a KV cache avoids this embodied cost but draws GPU power. The environmental implications of the retention decision, i.e., which tier to store in, and for how long, remain unexamined.

B. Carbon Emissions of AI Infrastructure

The carbon footprint of computing infrastructure comprises two categories: operational emissions and embodied emissions [13]. Operational emissions arise from the electricity consumed by hardware, and their magnitude depends on both power and the *carbon intensity* of the electrical grid (i.e., grams of CO_2 emitted per kilowatt-hour). Embodied emissions are a one-time emissions cost from manufacturing, transporting, and disposing of hardware.

For AI systems, prior work [2], [13], [14] has shown that embodied emissions constitute a substantial and growing share of total emissions, particularly as operational efficiency improves and hardware refresh cycles accelerate. Prior work [15], [16] particularly shows that SSDs carry a high embodied carbon cost per unit of capacity due to the energy-intensive semiconductor fabrication process. As data centers deploy more SSDs to meet growing demand, such as from KV cache storage, the embodied carbon of storage becomes an increasingly relevant factor of AI systems' carbon emissions.

To reduce AI systems' emissions, one must consider both operational and embodied emissions, as decreasing one can lead to increases in the other, forming some fundamental tradeoffs [14]. For example, a decision that reduces operational emissions, such as caching a result to avoid recomputation, can increase embodied emissions if the caching requires additional storage hardware. This tension motivates a joint analysis of both emission categories, which we describe next.

C. The Store-versus-Recompute Tradeoff

Given the above context, the KV cache retention decision maps directly onto the operational-versus-embodied carbon tradeoff. Consider a concrete example: a chatbot serving multi-turn conversations where each message shares the conversation history as a common prefix. After completing a response, the system must decide whether to retain the just-computed KV cache on an SSD or discard it. If the user sends a follow-up message a few seconds later, retaining the cache avoids a GPU recomputation that would have consumed power, thus exacerbating operational emissions. However, if the user does not return for many hours, or not at all, the SSD capacity occupied by the cache contributed embodied emissions from the storage hardware provisioned to hold it—emissions that may exceed what a brief recomputation would have produced.

The carbon-saving choice depends on the *reuse distance*: the time between when the system stores the KV cache and when a follow-up query requires it. Short reuse distances favor storage because the embodied emissions decrease with shorter distance, while the operational emissions of recomputation stay the same. The fixed recomputation cost thus leads to recomputation being favored with long reuse distances. At some reuse distance, the two costs are equal—this duration is the *break-even point*. We identify this break-even point using the analytical model we present in the next section.

TABLE I
SUMMARY OF NOTATION.

Symbol	Description
S_{KV}	KV cache size (GB)
S_{SSD}	SSD capacity (GB)
t_{store}	Duration the KV cache is stored on the SSD
t_{recomp}	Time to recompute the KV cache on the GPU
t_{life}	Hardware operational lifetime
P_{SSD}	SSD idle power draw (W)
P_{GPU}	GPU active power draw (W)
P_{other}	Non-GPU component power draw (W)
CI	Grid carbon intensity (gCO_2eq/kWh)
PUE	Power Usage Effectiveness
B	Batch size (concurrent requests)
c_{SSD}	Embodied carbon per GB of SSD (gCO_2eq/GB)
c_{GPU}	Embodied carbon of the GPU system (gCO_2eq)

III. METHODOLOGY

We model the *added* carbon footprint of two KV cache management strategies: (1) storing the cache on an SSD for later retrieval (§III-A), and (2) discarding the cache and recomputing it when a follow-up query arrives (§III-B). We also estimate the latency of each strategy to identify carbon-performance tradeoffs. Table I summarizes the notation used throughout this section.

To isolate the carbon emissions of the retention decision, we model only the *additional* emissions incurred beyond the baseline footprint of an LLM inference deployment. In the storage scenario, the additional emissions come from embodied emissions of the SSD hardware and its power draw. In the recompute scenario, the additional cost comes from the GPU computation and the associated hardware amortization. Each scenario accounts for both operational carbon (from electricity) and embodied carbon (from hardware manufacturing).

A. Storage Scenario

The storage scenario retains the KV cache on an SSD for a duration t_{store}, representing the reuse distance.

Operational carbon. The SSD consumes power while holding the cache. We compute the energy drawn during t_{store}, scaled to the fraction of SSD capacity the KV cache occupies:

$$E_{\text{store}} = P_{\text{SSD}} \cdot t_{\text{store}} \cdot \frac{S_{\text{KV}}}{S_{\text{SSD}}}. \tag{1}$$

Multiplying by the grid carbon intensity yields the operational carbon:

$$\Delta C_{\text{op}}^{\text{store}} = CI \cdot E_{\text{store}}. \tag{2}$$

Embodied carbon. As with prior work [14], [17], we amortize the SSD's embodied carbon over its operational lifetime. The per-GB embodied cost c_{SSD} (derived from prior work [15]; see §IV for values) is scaled by the cache size and the fraction of the SSD's lifetime the cache occupies:

$$\Delta C_{\text{emb}}^{\text{store}} = c_{\text{SSD}} \cdot S_{\text{KV}} \cdot \frac{t_{\text{store}}}{t_{\text{life}}}. \tag{3}$$

Total carbon. The added carbon footprint of storing the KV cache is:

$$\Delta C_{\text{total}}^{\text{store}} = \Delta C_{\text{op}}^{\text{store}} + \Delta C_{\text{emb}}^{\text{store}}. \tag{4}$$

B. Recompute Scenario

The recompute scenario discards the KV cache after the initial query and regenerates it from scratch when the follow-up query arrives. The added carbon is attributed to the GPU cluster that performs the recomputation.

Operational carbon. During recomputation, the GPU operates near its maximum rated power [18] because KV cache generation (the prefill phase) is compute-bound. The total system power includes the GPU, non-GPU components (e.g., host CPU, DRAM, networking), and facility overhead captured by PUE [19]:

$$P_{\text{sys}} = (P_{\text{GPU}} + P_{\text{other}}) \cdot PUE. \tag{5}$$

The recompute time t_{recomp} depends on the number of input tokens, model architecture, and hardware. Based on internal profiling, we use a piecewise model: t_{recomp} scales linearly with token count for sequences $\leq$8k tokens and quadratically for sequences $>$8k tokens (see §IV for the fitted parameters). The recompute energy is then:

$$E_{\text{recomp}} = P_{\text{sys}} \cdot t_{\text{recomp}}. \tag{6}$$

We assume a batch of B requests recompute their KV caches in parallel across the GPU cluster. Each request's operational carbon is:

$$\Delta C_{\text{op}}^{\text{recomp}} = \frac{CI \cdot E_{\text{recomp}}}{B}. \tag{7}$$

Embodied carbon. We amortize the GPU system's total embodied carbon c_{GPU} (including GPU dies and HBM) over its operational lifetime, then attribute a share proportional to t_{recomp} and divide across the batch:

$$\Delta C_{\text{emb}}^{\text{recomp}} = c_{\text{GPU}} \cdot \frac{t_{\text{recomp}}}{t_{\text{life}}} \cdot \frac{1}{B}. \tag{8}$$

Total carbon. The added carbon footprint of recomputing the KV cache is:

$$\Delta C_{\text{total}}^{\text{recomp}} = \Delta C_{\text{op}}^{\text{recomp}} + \Delta C_{\text{emb}}^{\text{recomp}}. \tag{9}$$

C. Break-Even Reuse Distance

The storage scenario's carbon cost grows with t_{store} (Equations 2 and 3), while the recompute scenario's cost is independent of t_{store}. The *break-even reuse distance* t^*_{store} is the storage duration at which both strategies produce the same added carbon:

$$\Delta C_{\text{total}}^{\text{store}}(t^*_{\text{store}}) = \Delta C_{\text{total}}^{\text{recomp}}. \tag{10}$$

Substituting Equations 4 and 9 and solving for t^*_{store}:

$$t^*_{\text{store}} = \frac{\Delta C_{\text{op}}^{\text{recomp}} + \Delta C_{\text{emb}}^{\text{recomp}}}{CI \cdot P_{\text{SSD}} \cdot \frac{S_{\text{KV}}}{S_{\text{SSD}}} + c_{\text{SSD}} \cdot S_{\text{KV}} \cdot \frac{1}{t_{\text{life}}}}. \tag{11}$$

For reuse distances shorter than t^*_{store}, storing the KV cache produces fewer emissions than recomputing. For reuse distances longer than t^*_{store}, recomputing is carbon-saving. By varying parameters such as CI, S_{KV}, and GPU device, we examine how the break-even point shifts across different deployment configurations in §V.

IV. Implementation

We now detail the configuration parameters and assumptions used to evaluate the carbon-saving storage duration.

A. Hardware Configuration

Similar to prior work [20], we model an Nvidia HGX H100 system containing 8 GPUs, each with 80GB of HBM. We model our SSDs after the Seagate Nytro 3331 SSD with 960GB capacity [21].

B. Embodied Carbon Data

We use ACT to calculate the embodied emissions of the SSD [17]. We derive the embodied emissions of the 8-GPU system from manufacturer reports [22]. We assume a six-year lifetime [23].

C. Operational Carbon Data

The electricity grid's carbon intensity affects the operational carbon emissions of the storage and recompute scenarios. We use carbon intensities reported for Azure data center regions and for specific energy types, such as coal and natural gas, from prior work [14], [24]. We use the Power Usage Effectiveness (PUE) reported by Microsoft Azure [19]. Component power consumption is determined from manufacturer reports [21], [25].

D. Model Parameters

We use Meta's Llama-2-70b model for our evaluation [26]. We calculate the KV cache size as a function of the number of tokens and model parameters, similar to existing model size calculators [27]. For the compute time, we use a piecewise function mapping the number of input tokens to the time-to-first-token (TTFT). To calculate the TTFT, we assume a linear relationship between the number of tokens and TTFT for $\leq$8k tokens and use llama-2 data from prior work to construct a linear equation [20]. For the TTFT of a large number of input tokens (>8k tokens), we fit a quadratic function to prior internal data.

V. Results

Fig. 1 shows the break-even point (ideal storage duration) for the Llama-2-70b model based on a fixed number of tokens and the average carbon intensity from the U.S. electricity grid. The shaded region to the left of the break-even points indicates storing the KV cache is better from a carbon standpoint when the duration of storage is less than $\sim$ 3 minutes. Beyond this duration, recomputing the KV cache using the GPU would lower the carbon footprint compared to keeping it in storage. This means that for a user in an ongoing session with an LLM chatbot with 8000 tokens per query, if they step away from the conversation for durations longer than $\sim$ 3 minutes, it is advantageous to discard the KV cache and recompute later.

At higher carbon intensities, it is beneficial to store the KV cache in storage for longer rather than recompute. Fig. 2 shows how the carbon-saving duration of storage increases as the carbon intensity of the electricity grid powering the LLM inference hardware increases. This is because the carbon emissions of recomputing are primarily due to operational carbon emissions, which increase with the carbon intensity, resulting in higher overall carbon emissions for recomputing.

Due to our piecewise model for TTFT in the recompute scenario, for $\leq$8k tokens, both the recompute TTFT and the KV cache size are scaled by the number of tokens, leading to a net zero change in the break-even point as the number of tokens increases up to 8k. However, past 8k tokens, the

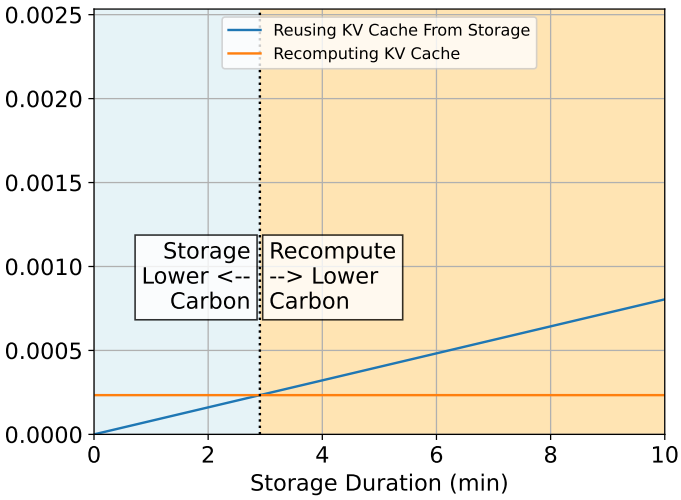

Fig. 1. Additional carbon emissions versus storage duration when reusing the KV cache from storage versus recomputing the KV cache, using a batch size of 512 when recomputing on the GPUs. Added carbon is additional carbon emissions relative to a baseline LLM inference system, due to either SSD (storage) or GPU usage (recompute); we focus on this additional carbon to determine the break-even point. Results shown are for Llama-2-70b with 8000 tokens, KV cache size of 2.44GB, prefill run time of 0.51s, and CI of 138 g/kWh (Azure US East).

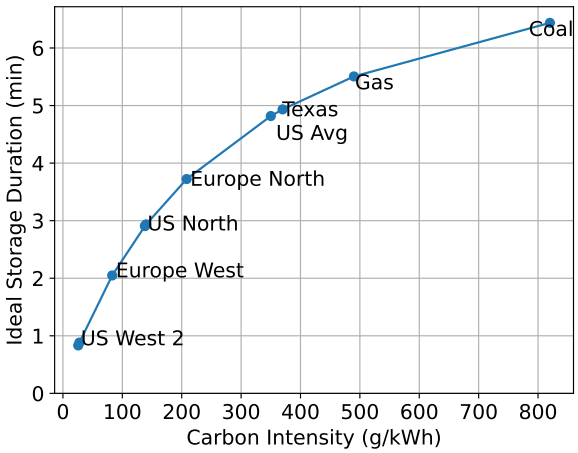

Fig. 2. Ideal storage duration for varying electricity grid carbon intensity. Results shown are for Llama-2-70b with 8000 tokens, KV cache size of 2.44GB, and prefill run time of 0.51s

TABLE II
VARIOUS SSD EMBODIED CARBON ESTIMATES (TOKENS = 8000, KV CACHE: 2.4414 GB), CI = 138 G/KWH

Approach	SSD Embodied Carbon (g/GB)	Ideal Storage Duration (min)
Seagate Manufacturer Report [21]	5.61	12.77
ACT [17]	69.94	4.87
Dell R740 LCA [28]	878.39	0.55

TABLE III
LATENCY PER SCENARIO (TOKENS = 8000, KV CACHE: 2.4414 GB)

Scenario	TTFT (s)
Storing	2.2195
Recomputing	0.5067

recompute TTFT grows quadratically and scales the recompute carbon footprint accordingly, whereas the storage carbon footprint still only scales linearly. As a result, it is favorable to store the KV cache for a longer duration as the number of tokens increases.

Our model is dependent on the underlying values selected, such as the baseline SSD embodied carbon. Table II shows how varying estimates of the embodied carbon in gCO2e per GB affects the ideal storage duration. We choose the ACT embodied carbon as a middle-of-the-pack estimate for our model.

We also present a first-order estimate of the latency to read the stored KV cache from SSD in Table III. We exclude network delay. Although the TTFT is higher for the storage scenario and may make it challenging to meet SLOs, this could be reduced in practice by techniques such as striping and KV cache compression. If the LLM inference system includes striping, we could read the KV cache in parallel chunks from N number of SSDs. With striping, the SSD read latency would decrease by $\times N$. KV cache compression techniques should further reduce the latency of storing the KV cache, assuming relatively small delays for compression/decompression.

VI. CONCLUSION

This work characterizes the environmental tradeoff between storing LLM KV caches on persistent hardware and recomputing them on GPUs. The analytical model identifies specific break-even reuse distances where the embodied emissions of storage hardware equals the operational emissions of GPU-based recomputation. Results demonstrate that these break-even points shift significantly based on grid carbon intensity, model scale, and hardware specifications.

The analysis also reveals a fundamental tension between carbon-saving decisions and latency constraints. In several scenarios, the retrieval of a stored cache minimizes emissions but may violate strict performance Service Level Objectives (SLOs). Conversely, recomputation provides faster response times at a higher environmental cost.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their feedback. This work was partially supported by an NSF CAREER Award CCF-2340042.

REFERENCES

[1] B. Li, Y. Jiang, V. Gadepally, and D. Tiwari, "LLM Inference Serving: Survey of Recent Advances and Opportunities," in *High Performance Extreme Computing Conference*, 2024.

[2] C.-J. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. Aga, J. Huang, C. Bai, M. Gschwind, A. Gupta, M. Ott, A. Melnikov, S. Candido, D. Brooks, G. Chauhan, B. Lee, H.-H. Lee, B. Akyildiz, M. Balandat, J. Spisak, R. Jain, M. Rabbat, and K. Hazelwood, "Sustainable AI: Environmental Implications, Challenges and Opportunities," *Proceedings of Machine Learning and Systems*, 2022.

[3] A. A. Chien, L. Lin, H. Nguyen, V. Rao, T. Sharma, and R. Wijayawardana, "Reducing the Carbon Impact of Generative AI Inference (Today and in 2035)," in *Workshop on Sustainable Computer Systems*, 2023.

[4] P. Li, J. Yang, M. A. Islam, and S. Ren, "Making AI Less 'Thirsty'," *Communications of the ACM*, vol. 68, 2025.

[5] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, J. Heek, K. Xiao, S. Agrawal, and J. Dean, "Efficiently Scaling Transformer Inference," *Proceedings of Machine Learning and Systems*, 2023.

[6] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient Memory Management for Large Language Model Serving with PagedAttention," in *Symposium on Operating Systems Principles*, 2023.

[7] C. Hooper, S. Kim, H. Mohammadzadeh, M. W. Mahoney, Y. S. Shao, K. Keutzer, and A. Gholami, "KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization," *Advances in Neural Information Processing Systems*, 2024.

[8] B. Gao, Z. He, P. Sharma, Q. Kang, D. Jevdjic, J. Deng, X. Yang, Z. Yu, and P. Zuo, "Cost-Efficient Large Language Model Serving for Multi-Turn Conversations with CachedAttention," in *Annual Technical Conference*, 2024.

[9] L. Zheng, L. Yin, Z. Xie, C. L. Sun, J. Huang, C. H. Yu, S. Cao, C. Kozyrakis, I. Stoica, J. E. Gonzalez, C. Barrett, and Y. Sheng, "SGLang: Efficient Execution of Structured Language Model Programs," *Advances in Neural Information Processing Systems*, 2024.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All You Need," *Advances in Neural Information Processing Systems*, 2017.

[11] Y. Sheng, L. Zheng, B. Yuan, Z. Li, M. Ryabinin, B. Chen, P. Liang, C. Ré, I. Stoica, and C. Zhang, "FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU," in *International Conference on Machine Learning*, 2023.

[12] Z. Xie, Z. Xu, M. Zhao, Y. An, V. S. Mailthody, S. Mahlke, M. Garland, and C. Kozyrakis, "Strata: Hierarchical Context Caching for Long Context Language Model Serving," *arXiv preprint arXiv:2508.18572*, 2025.

[13] U. Gupta, Y. G. Kim, S. Lee, J. Tse, H.-H. S. Lee, G.-Y. Wei, D. Brooks, and C.-J. Wu, "Chasing Carbon: The Elusive Environmental Footprint of Computing," in *International Symposium on High-Performance Computer Architecture*, 2021.

[14] J. Wang, D. S. Berger, F. Kazhamiaka, C. Irvene, C. Zhang, E. Choukse, K. Frost, R. Fonseca, B. Warrier, C. Bansal, J. Stern, R. Bianchini, and A. Sriraman, "Designing Cloud Servers for Lower Carbon," in *International Symposium on Computer Architecture*, 2024.

[15] S. Tannu and P. J. Nair, "The Dirty Secret of SSDs: Embodied Carbon," *SIGENERGY Energy Informatics Review*, vol. 3, 2023.

[16] S. Mcallister, F. Kazhamiaka, D. S. Berger, R. Fonseca, K. Frost, A. Ogus, M. Sah, R. Bianchini, G. Amvrosiadis, N. Beckmann, and G. R. Ganger, "A Call for Research on Storage Emissions," *ACM SIGENERGY Energy Informatics Review*, vol. 4, 2024.

[17] U. Gupta, M. Elgamal, G. Hills, G.-Y. Wei, H.-H. S. Lee, D. Brooks, and C.-J. Wu, "ACT: Designing Sustainable Computer Systems With an Architectural Carbon Modeling Tool," in *Annual International Symposium on Computer Architecture*, 2022.

[18] A. K. Kamath, R. Prabhu, J. Mohan, S. Peter, R. Ramjee, and A. Panwar, "Pod-attention: Unlocking full prefill-decode overlap for faster llm inference," in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 2025, pp. 897–912.

[19] Microsoft, "Microsoft 2024 environmental sustainability report," https://cdn-dynmedia-1.microsoft.com/is/content/microsoftcorp/microsoft/msc/documents/presentations/CSR/Microsoft-2024-Environmental-Sustainability-Report.pdf, 2024.

[20] P. Patel, E. Choukse, C. Zhang, A. Shah, Goiri, S. Maleki, and R. Bianchini, "Splitwise: Efficient generative llm inference using phase splitting," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024, pp. 118–132.

[21] Seagate Technology, "Nytro 3331 sustainability report," 2020, product Sustainability Report. [Online]. Available: https://www.seagate.com/files/www-content/global-citizenship/_shared/product-sustainability/nytro-3331-sustainability-report/_shared/pdf/nytro-3331-sustainability-report-en-ca.pdf

[22] Nvidia, "Product carbon footprint (pcf) summary for nvidia hgx h100," https://images.nvidia.com/aem-dam/Solutions/documents/HGX-H100-PCF-Summary.pdf, 2025.

[23] J. Lyu, J. Wang, K. Frost, C. Zhang, C. Irvene, E. Choukse, R. Fonseca, R. Bianchini, F. Kazhamiaka, and D. S. Berger, "Myths and misconceptions around reducing carbon embedded in cloud platforms," in *Proceedings of the 2nd Workshop on Sustainable Computer Systems*, ser. HotCarbon '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3604930.3605717

[24] S. Schlömer, T. Bruckner, L. Fulton, E. Hertwich, A. McKinnon, D. Perczyk, J. Roy, R. Schaeffer, R. Sims, P. Smith *et al.*, "Annex iii: Technology-specific cost and performance parameters," in *Climate change 2014: Mitigation of climate change: Contribution of working group III to the fifth assessment report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, 2014, pp. 1329–1356.

[25] Nvidia, "NVIDIA H100 GPU Datasheet — resources.nvidia.com," https://resources.nvidia.com/en-us-hopper-architecture/nvidia-tensor-core-gpu-datasheet?ncid=no-ncid, 2024.

[26] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[27] LMCache, "KV Cache Size Calculator — lmcache.ai," https://lmcache.ai/kv_cache_calculator.html, 2025.

[28] Dell, "Lca dell r740," https://www.delltechnologies.com/asset/en-us/products/servers/technical-support/{F}ull_{L}{C}{A}_{D}ell_{R}740.pdf, 2019.