

Disaggregated Speculative Decoding for Carbon-Efficient LLM Serving

Tianyao Shi , Yanran Wu , Sihang Liu , and Yi Ding , Member, IEEE

Abstract—Large language models (LLMs) are increasingly deployed in practice but incur significant computational costs and environmental impacts. Disaggregated serving techniques, particularly decoupling prefill and decoding (DPD) across GPUs, have been introduced to improve performance and reduce carbon emissions. However, DPD suffers from high bandwidth overhead due to frequent large KV cache transfers. To address this, we present disaggregated speculative decoding (DSD), which leverages speculative decoding by assigning draft models to older GPUs and target models to newer GPUs, requiring only token and probability distribution transfers. Building on this insight, we introduce GreenLLM, an SLO- and bandwidth-aware framework that unifies DPD and DSD, profiles workload characteristics, and dynamically selects the most carbon-efficient configuration. Across diverse benchmarks, GreenLLM reduces carbon emissions by up to 40.6% while meeting latency SLOs.

Index Terms—Large language models, disaggregated serving, speculative decoding, carbon emissions.

I. INTRODUCTION

LARGE language models (LLMs) are rapidly transforming various application domains. Serving LLMs involves two phases: the *prefill* phase, which processes the full prompt and is highly compute-intensive, and the *decoding* phase, which generates tokens autoregressively and is more memory-intensive. To reduce interference and improve performance, recent systems, such as Splitwise [1], disaggregated the two phases, offloading the less compute-bound decoding phase to older or less powerful GPUs [2], [3], a method we term **DPD**.

Meanwhile, LLM serving also raises sustainability concerns due to its significant carbon emissions [4]. There are two sources of carbon emissions. *Operational carbon* arises from energy consumption during LLM serving; for instance, a single ChatGPT prompt generates 4 grams of CO₂eq, over 20× that of a web search query [5]. *Embodied carbon* stems from GPU manufacturing process, with high-performance GPUs carrying especially heavy embodied emissions [6].

While recent DPD systems such as DynamoLLM [3] consider carbon emission reduction, they suffer from a critical limitation: the high bandwidth required to transfer large KV caches between GPUs. For example, disaggregating prefill and decoding of Llama-7B across A100 and T4 GPUs demands over 10 Gbps of bandwidth even at a modest request rate (Fig. 1). As model sizes grow, the transfer size of KV cache increase, and network bottlenecks can offset the performance gain and carbon savings that DPD provides.

Received 25 August 2025; revised 17 October 2025; accepted 28 October 2025. Date of publication 6 November 2025; date of current version 25 November 2025. This work was supported by NSF under Grant CCF-2413870. (Tianyao Shi and Yanran Wu contributed equally to this work.) (Corresponding author: Yi Ding.)

Tianyao Shi, Yanran Wu, and Yi Ding are with Purdue University, West Lafayette, IN 47907 USA (e-mail: yiding@purdue.edu).

Sihang Liu is with the University of Waterloo, Waterloo, ON N2L 3G1, Canada.

Digital Object Identifier 10.1109/LCA.2025.3630094

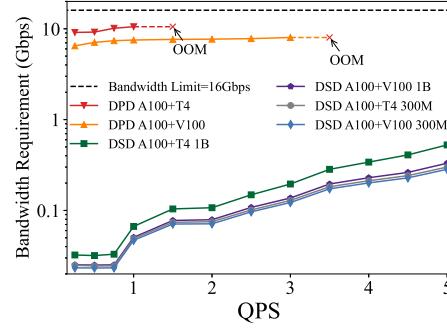


Fig. 1. Bandwidth requirement comparisons between DPD and DSD when connecting A100 and T4/V100 at different request rates (QPS, req/s). DPD runs the 7B model only, with the prefill phase running on A100 and the decoding phase on T4/V100. For DSD, the 7B target model runs on A100, while the 1B or 300 M draft model runs on T4/V100.

To overcome the bandwidth bottleneck, we introduce a new disaggregated serving method: disaggregated speculative decoding (**DSD**). DSD builds on speculative decoding [7], a recent technique that accelerates inference by running two models together: a smaller, faster draft model that generates candidate tokens, and a larger, more accurate model that verifies them. DSD assigns the large model to newer GPUs and the smaller model to older GPUs. Unlike DPD, DSD transfers only speculative tokens and their probability distributions between GPUs, drastically reducing bandwidth needs. To further enhance communication efficiency, we introduce asynchronous communication-computation overlap that exploits temporal dependencies between draft and target models.

In addition, we design GreenLLM, an SLO- and bandwidth-aware framework for carbon-efficient LLM serving. GreenLLM incorporates: (1) a disaggregated runtime supporting both DPD and DSD configurations; (2) a profiler that measures performance and energy under diverse workloads; and (3) a scheduler that dynamically selects serving configurations to minimize total (operational and embodied) carbon while meeting latency SLOs. We evaluate GreenLLM on chatbot, code generation, and document summarization workloads under various request rates. Compared to serving on new GPUs alone, GreenLLM reduces total carbon emissions by up to 40.6% while meeting latency SLOs.

II. BACKGROUND AND MOTIVATION

This section first introduces carbon accounting methods and then speculative decoding, followed by a motivational example of bandwidth conditions for disaggregated LLM serving.

TABLE I
SPECIFICATIONS OF GPUs IN THIS PAPER

GPU Model	T4	V100	A100
Memory Bandwidth (GB/s)	320	900	1555
Chip Area (mm ²)	545	815	826
Max Power (W)	70	300	400
Technology Node (nm)	12	12	7
FP16 TFLOPs	65	28.26	312
Year	2018	2017	2020
Embodied Carbon (kgCO₂)	10.3	20	26.34

A. Carbon Emissions Accounting

The total carbon emission includes embodied and operational carbon emissions.

Embodied carbon. The embodied carbon of a hardware device C_e is modeled as a function of processor chip area and memory capacity [6]. We evaluate three GPU types—T4, V100, and A100—summarized in Table 1. These GPUs are selected to reflect the heterogeneous and generational mix typical of academic (e.g., CloudLab [8]) and public clouds (e.g., GCP, AWS), where older systems remain operational alongside newer ones. The A100 offers the highest compute capability and memory capacity, resulting in the largest embodied carbon. The V100, with a larger chip area than the T4, also has a higher embodied emission despite same memory capacity. This comparison highlights how GPU generation and architecture affect embodied carbon in LLM serving workloads.

The embodied carbon emission of a request is amortized over time by the ratio of the total execution time to the hardware lifetime (LT) [6] and the average batch size (\bar{BS}_{req}) of the inference engine during serving. A typical GPU lifetime is 5–7 years. A request executed on a GPU for t_{req} time yields the embodied carbon emission of:

$$C_{req,e} = \frac{t_{req}}{\bar{BS}_{req} \cdot LT} \cdot C_e \quad (1)$$

Operational carbon. The operational carbon emission of a request $C_{req,o}$ is determined by the product of its energy consumption E_{req} and carbon intensity of the grid (CI). *Carbon intensity* is measured as grams of CO_{2eq} emitted per kWh of electricity generated or consumed [9]. The operational carbon emission of a request is:

$$C_{req,o} = E_{req} \cdot CI \quad (2)$$

a) Total carbon: For a request that executes for t_{req} time, the total carbon emission is:

$$C_{req} = C_{req,e} + C_{req,o} = \frac{t_{req}}{\bar{BS}_{req} \cdot LT} C_e + E_{req} \cdot CI \quad (3)$$

B. SPECULATIVE DECODING

Speculative decoding consists of three components: the *draft model*, the *target model*, and the *verifier* [7]. The draft model is a smaller, more efficient model that generates a sequence of speculative tokens, while the target model, which is larger and more capable, takes these speculative tokens and performs decoding to compute the probability distribution. Finally, the

verifier decides whether to accept the tokens proposed by the draft model by comparing the probabilities from both models, typically using rejection sampling. This draft-then-verify paradigm enables simultaneous decoding of multiple tokens per step, significantly accelerating inference compared to the traditional autoregressive decoding.

C. BANDWIDTH REQUIREMENT IMPACT

As noted in Section I, a key drawback of DPD is its high interconnect bandwidth demand from transferring large KV caches. Prior work [1], [2], [3] recognizes this issue but typically relies on high-end GPUs with NVLink or InfiniBand. The problem remains in settings without such connectivity and worsens when mixing heterogeneous GPUs, where older, lower-tier GPUs are paired with newer ones. In such cases, DPD’s performance and carbon efficiency can degrade due to network bottlenecks, synchronization delays, and underutilization of high-end GPUs.

Fig. 1 shows the bandwidth requirements when connecting A100 and T4/V100 GPUs under a 16 Gbps cap, which is the actual bandwidth constraint in our evaluation setup on Google Cloud. Serving LLaMA-7B with DPD quickly runs into out-of-memory (OOM) errors as QPS increases, since older GPUs lack sufficient memory to buffer the full KV cache. OOM occurs before reaching the network limit, as older GPUs can store only weights and partial KV cache for parallel requests. In contrast, DSD avoids this issue: with LLaMA-7B as the target model and smaller (1B, 300 M) draft models, it reduces bandwidth requirement by a factor of 65–434 across diverse request rates because it transfers only tokens and probability distributions instead of large KV caches. Thus, DPD requires high-bandwidth interconnects, while DSD enables efficient disaggregation under limited bandwidth.

III. DSD AND GREENLLM

We present DSD, which assigns the draft model to an older GPU and the target model (along with the verifier) to a newer GPU. However, separating the draft and target models across GPUs inevitably introduces communication overhead.

To maximize carbon savings while minimizing this overhead, we analyze the temporal dependencies between the two models for communication optimization. We observe that the speculative token IDs and the draft model’s probabilities differ significantly in size, with the probabilities being V times larger, where V is the vocabulary size. Importantly, the verifier does not require the draft probabilities until after the target model has processed the speculative tokens.

Given this insight, we optimize communication as shown in Fig. 2. We first transfer only the speculative token IDs, which are relatively small in size. While the target model performs decoding using these token IDs, we overlap communication and computation by asynchronously transferring the larger draft probabilities in parallel. This approach minimizes communication overhead by leveraging the delayed dependency of probabilities, effectively hiding data transfer delays between the draft and target models.

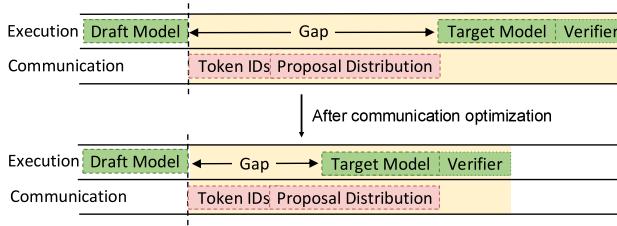


Fig. 2. Communication overlapping optimization for DSD. Green blocks indicate the execution of each component. Red blocks indicate data transfer. The yellow background indicates the time comparisons.

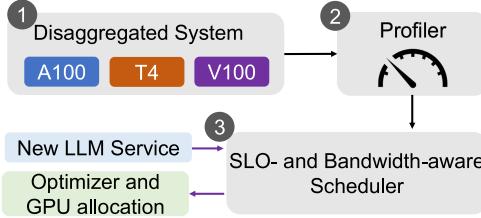


Fig. 3. Workflow of GreenLLM.

Combining DSD and DPD, we design GreenLLM, an SLO- and bandwidth-aware LLM serving framework that minimizes total carbon emissions by reusing older GPUs. Fig. 3 shows the workflow of GreenLLM, which consists of three components: ① a disaggregated system on heterogeneous GPUs, ② a profiler, and ③ an SLO- and bandwidth-aware scheduler. ① implements how we disaggregate two optimizers (DPD and DSD) on heterogeneous GPUs with minimal communication overhead. ② performs profiling to measure GPU performance and carbon emissions across request sizes and rates. ③ selects the optimal optimizer and GPU configuration to minimize carbon emissions using profiled data.

IV. IMPLEMENTATION

We implemented GreenLLM on top of the open-source LLM serving platform vLLM 0.5.4+cu122 [10], integrating the profiler and DSD method as plug-ins. For DPD, we leveraged community implementations of vLLM based on the design principles of Splitwise [1]. The profiler uses `pynvml` APIs to collect GPU power consumption data with minimal overhead. Communication and profiling data aggregation across multiple servers are managed via RESTful APIs, ensuring robustness and efficiency in distributed environments.

For DSD, we used Ray actors to create GPU workers for the draft model inference, assigning older GPUs in a heterogeneous cluster for running the draft model workers. Unlike the original vLLM design, we introduced two distinct parallelism groups on heterogeneous nodes to support parallel inference. One group was responsible for the draft model, while the other handled the primary serving instance, including the serving API, target model, and verifier. To facilitate communication between these groups, we used NCCL along with PyTorch's asynchronous communication primitives, `isend` and `irecv`, to avoid blocking GPU computations during data transmission.

V. EVALUATION

In this section, we first describe the evaluation methodology and then present carbon saving results.

A. Methodology

System setup. We evaluate GreenLLM using Google Cloud Platform (GCP) servers with one A100 GPU and one T4/V100 GPU. Servers are connected within the GCP network with a default bandwidth of 16 Gbps.

Models and configurations. We experiment with Llama models of 7B, 1B, and 300 M parameters. GreenLLM automatically selects among the following configurations to minimize carbon emissions while achieving *SLO attainment*, which requires 90% of requests to satisfy both TTFT and TPOT latency SLOs.

- *Standalone*: serve a 7B Llama model on a single A100 with chunked prefill enabled (using engine-default size=512) and prefix caching disabled. This is our *baseline* as it represents a non-disaggregated system that does not reuse old GPUs but only uses newer GPUs.
- *SpecDecode* (Single-GPU Speculative Decoding): serve target (7B) and draft (1B or 300 M) models on a single A100, fixing the number of speculative tokens to 3. The average acceptance rates are 35.3% for the 300 M draft and 45.7% for the 1B draft model.
- *DPD*: serve prefill and decoding phases of a 7B Llama model on different GPUs: prefill on newer GPU (A100) and decoding on older GPU (T4 or V100).
- *DSD*: use 1B and 300 M Llama models as draft models running on old GPUs (T4 or V100), and 7B Llama model as the target model on the newer GPU (A100), keeping all other speculative decoding settings identical to *SpecDecode*.

Workloads. We evaluated the same LLM serving datasets as in previous work [2]: ShareGPT, HumanEval, and LongBench. They represent three types of LLM applications: chatbot, code completion, and summarization. Table II summarizes the SLOs, and input and output distributions for each workload. **Accounting scope.** We focus on GPU-related carbon emissions, as GPUs are the primary executors of LLM workloads. We exclude contributions from CPUs, memory, and storage. For disaggregated serving configurations, we amortize both embodied and operational carbon emissions across the full execution period, accounting for idle energy and time on newer GPUs while they wait for older GPUs to complete decoding (DPD) or generate token proposals (DSD).

Measurement. For each workload, we sweep over the achievable QPS across all configurations and collect data using 2-minute profiling runs at each QPS, preceded by a 20-second warm-up stage. We measure execution time by inserting a timer in GreenLLM and recording token generation timestamps. During runtime, we measure GPU power consumption every 200 ms using the `pynvml` library. Carbon emissions are calculated using a carbon intensity of 261 gCO₂/kWh from the CISO grid by default.

Metrics. We use the following metrics: (1) Carbon Per Token: gCO₂ of carbon emission per token; (2) Time To First Token

TABLE II
DATASETS FOR EVALUATION AND SLO REQUIREMENTS

Dataset	Task	TTFT SLO	TPOT SLO	P25 Req. Size	P50 Req. Size	P75 Req. Size
ShareGPT	Chatbot	200ms	80ms	(24,24)	(160,140)	(510,357)
HumanEval	Code Generation	125ms	200ms	(108,31)	(136,55)	(182,88)
LongBench	Summarization	15s	150ms	(1134,201)	(1495,275)	(1817,352)

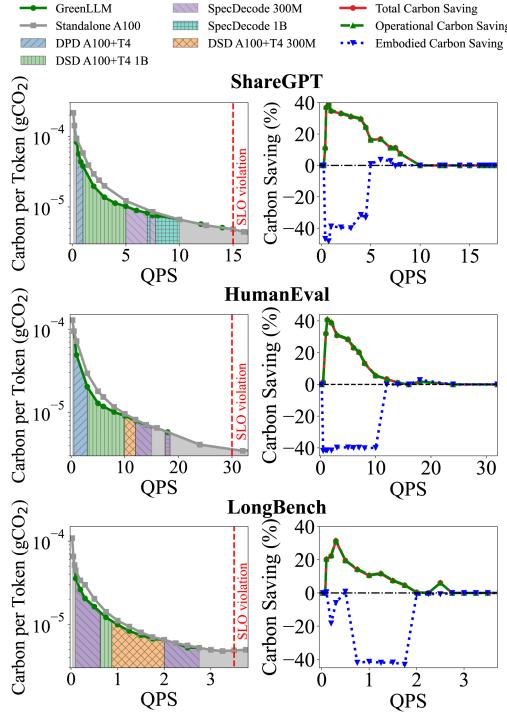


Fig. 4. Carbon emission per token (left) and carbon savings comparison between GreenLLM and the baseline Standalone A100 (right).

(TTFT): the latency between prompt reception and the generation of the first token; and (3) Time Per Output Token (TPOT): the average latency to generate each token during decoding.

B. Carbon Emission Savings

Fig. 4 shows carbon emission results: the left column shows carbon per token across QPS values, and the right column shows the corresponding carbon emission savings per token compared to the standalone A100 baseline. The input and output sizes of requests follow the median value of each dataset. GreenLLM dynamically selects the optimal serving configuration (Standalone, DPD, DSD, or SpecDecode, along with GPU pairing and draft model size) to minimize total carbon emissions under SLO constraints. Overall, GreenLLM achieves up to 31.3–40.6% carbon savings compared to Standalone when 90% of requests satisfy both TTFT and TPOT SLOs, i.e., an SLO attainment of 90%. Carbon savings primarily occur at lower QPS ranges, though the optimal range varies by workload. LongBench, with highest input/output token numbers, benefits from older GPUs only at lower QPS range [0.75,2]. HumanEval, with lower input/output tokens, benefits across a wider QPS range [0.5,11].

The right column in Fig. 4 shows the breakdown of carbon savings into operational and embodied emissions. At lower QPS,

most carbon savings come from reduced operational carbon in disaggregated modes. Using older GPUs in disaggregated modes can increase embodied carbon compared to using standalone A100, because it introduces extra hardware but does not improve latency. As QPS increases, GreenLLM shifts towards SpecDecode method on a standalone A100, which substantially reduces embodied carbon. This occurs because at higher QPS, smaller draft models accelerate token generation, improving TPOT and reducing end-to-end latency. As indicated by (1), lower latency directly decreases embodied carbon. Overall, these results demonstrate that GreenLLM dynamically reduces both operational and embodied carbon emissions by selecting the most carbon-efficient configuration across varying QPS values.

VI. CONCLUSION

DSD offers a complementary disaggregation path to DPD tailored for bandwidth-limited environments. We design GreenLLM, an SLO- and bandwidth-aware framework that leverages older GPUs to reduce total carbon emissions. Results demonstrate GreenLLM's ability to adaptively choose the most carbon-efficient serving configuration. Future work includes extending GreenLLM to heterogeneous multi-GPU and hybrid parallel settings to further explore scalability and efficiency in sustainable LLM serving.

REFERENCES

- [1] P. Patel et al., “Splitwise: Efficient generative LLM inference using phase splitting,” in *Proc. ACM/IEEE Annu. Int. Symp. Comput. Architecture*, 2024, pp. 118–132.
- [2] Y. Zhong et al., “DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving,” in *Proc. USENIX Conf. Operating Syst. Des. Implementation*, 2024, pp. 193–210.
- [3] J. Stojkovic, C. Zhang, I. Goiri, J. Torrellas, and E. Choukse, “DynamoLLM: Designing LLM inference clusters for performance and energy efficiency,” in *Proc. 2025 IEEE Int. Symp. High Perform. Comput. Architecture*, Mar. 2025, pp. 1348–1362.
- [4] Y. Wu, I. Hua, and Y. Ding, “Unveiling environmental impacts of large language model serving: A functional unit view,” in *Proc. Conf. Assoc. Comput. Linguistics*, 2025, pp. 10560–10576.
- [5] S. Griffiths, “Why your internet habits are not as clean as you think,” 2020. [Online]. Available: <https://www.bbc.com/future/article/20200305-why-your-internet-habits-are-not-as-clean-as-you-think>
- [6] U. Gupta et al., “ACT: Designing sustainable computer systems with an architectural carbon modeling tool,” in *Proc. ACM/IEEE Annu. Int. Symp. Comput. Architecture*, 2022, pp. 784–799.
- [7] Y. Leviathan, M. Kalman, and Y. Matias, “Fast inference from transformers via speculative decoding,” in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 19274–19286.
- [8] D. Duplyakin et al., “The design and operation of CloudLab,” in *Proc. USENIX Annu. Tech. Conf.*, 2019, pp. 1–14.
- [9] A. Li, S. Liu, and Y. Ding, “Uncertainty-aware decarbonization for datacenters,” *SIGENERGY Energy Inform. Rev.*, vol. 4, no. 5, pp. 141–147, 2025, doi: [10.1145/3727200.3727221](https://doi.org/10.1145/3727200.3727221).
- [10] W. Kwon et al., “Efficient memory management for large language model serving with pagedattention,” in *Proc. Symp. Operating Syst. Princ.*, 2023, pp. 611–626.