International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Memory-Efficient LLM Training and Inference: Balancing Capacity, Speed, and Environmental Impact

Smitha Shivashankaraiah

Independent researcher
smithabs@gmail.com

Abstract:
The memory requirements of large language models (LLMs) present a critical bottleneck for both training and inference. While numerous techniques exist to reduce memory usage, most surveys organize them by mechanism or by the specific bottleneck addressed. This paper takes a different approach. We argue that real-world LLM deployment must balance three fundamental constraints: memory capacity, memory speed, and environmental impact. Capacity-focused techniques such as CXL memory pooling prioritize fitting large models and long contexts. Speed-focused techniques such as near-memory compute prioritize low latency for real-time and agentic workloads. Environmental factors — including carbon, water, noise, vibration, and e-waste — impose social and regulatory constraints that can override technical advantages. We present a comparative analysis of these three factors, a decision matrix for different workloads, and recommendations for engineers designing LLM infrastructure. Our conclusion is that memory efficiency is not merely a technical problem but a systems problem requiring trade-offs across capacity, speed, and sustainability.

Keywords: Memory efficiency; Large language models; GPU memory; CXL; HBM; Environmental sustainability; AI infrastructure

Page 1: Introduction

Large language models (LLMs) have transformed AI, but their memory requirements remain a critical bottleneck [1]. During training, activations, optimizer states (Adam's momentum and variance), and model parameters compete for GPU memory. During inference, the key-value (KV) cache grows with sequence length, often dominating memory consumption for long-context applications.

The industry has developed multiple memory-efficiency techniques — gradient checkpointing, LoRA, 8-bit optimizers, FlashAttention, PagedAttention, and various offloading and parallelism strategies. Existing surveys typically categorize these techniques by mechanism or by the specific bottleneck they address [1].

This paper takes a different approach. Instead of focusing on individual techniques, we organize the discussion around three fundamental constraints that any real-world deployment must balance:

1. **Memory capacity** — How much memory is available? Techniques such as CXL memory pooling and tiered storage prioritize capacity [3][6].

2. **Memory speed** — How fast can data be accessed? Near-memory compute and SRAM-based architectures prioritize low latency [2].

3. **Environmental and social impact** — What is the cost in heat, noise, water, carbon, e-waste, and community disruption? [4][8]

No single factor dominates. The right choice depends on workload, deployment context, and increasingly, regulatory and social constraints that engineers cannot afford to ignore.

Page 2: Factor 1 — Memory Capacity

The most direct approach to memory efficiency is to increase available capacity. For LLMs, the pressure comes from three sources: activations (stored during forward pass for backpropagation), optimizer state (Adam stores both momentum and variance, typically twice the model size) [5], and the KV cache (grows with sequence length and batch size).

Capacity-Focused Techniques

Technique	How It Works	Trade-off
CXL memory pooling	GPUs share a common pool of DDR memory	Lower bandwidth than HBM, but cheaper per GB [3][7]
Tiered storage	Hot data in HBM, cold data in DDR or SSD	Capacity scales, but access latency varies [6]
CPU offloading (ZeRO-Offload)	Optimizer state moved to host memory	Reduces GPU memory at cost of PCIe bandwidth [1]

When Capacity Matters Most

Capacity-focused approaches excel in three scenarios:

1. **Training large models** — The optimizer state alone for a 175B parameter model exceeds 1TB [1]. Without capacity techniques, training on a single node is impossible.

2. **Batch inference** — Serving thousands of requests simultaneously requires storing many KV caches. Capacity, not speed, is the limiting factor [7].

3. **Long context windows** — As context grows to 128K or 1M tokens, the KV cache expands linearly. Capacity techniques become essential [2].

Limitations

More capacity does not solve latency problems. For agentic workloads where the model makes multiple sequential calls, waiting for data from slow memory tiers adds unacceptable delay [2]. Capacity and speed must be balanced.

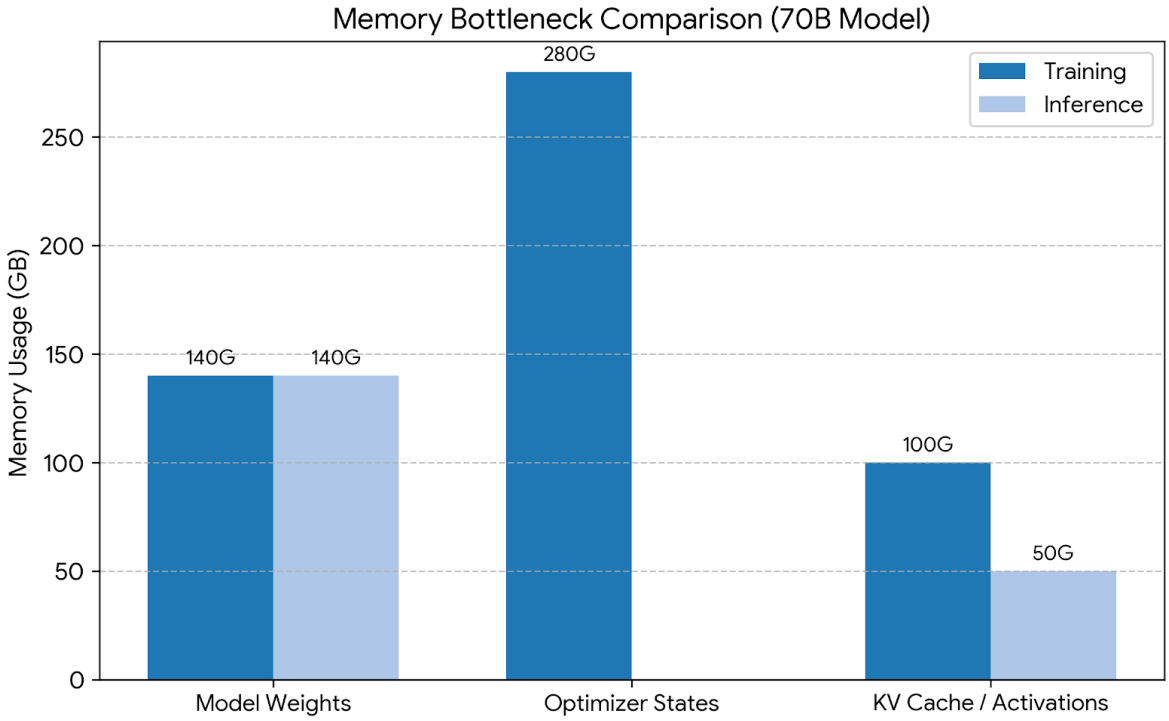

Figure 1: Memory Bottleneck Comparison. Model weights remain constant, but training is dominated by optimizer states while inference is constrained by KV cache capacity [1], [5]

Page 3: Factor 2 — Memory Speed

For real-time and agentic applications, access latency matters more than total capacity. Even with abundant memory, if data cannot be retrieved quickly, user experience degrades and agent loops become impractical [2].

Speed-Focused Techniques

Technique	How It Works	Trade-off
SRAM / near-memory compute	Memory placed close to compute logic (e.g., Groq architecture)	Extremely low latency, but lower density than DRAM [2]
HBM (High Bandwidth Memory)	Stacked memory with wide interface	High bandwidth, but expensive and power-hungry [4]
PagedAttention (vLLM)	Efficient KV cache management with virtual memory	Reduces fragmentation, improves effective speed [2]

When Speed Matters Most

Speed-focused approaches are critical for:

1. **Agentic AI** — Agents make sequential calls (plan → execute → observe → loop). Each call's latency adds directly to total response time. At 100ms per call, a 10-step agent takes a full second [2].

2. **Real-time inference** — Voice assistants, autonomous systems, and interactive applications require sub-second responses. Memory latency becomes the bottleneck [4].

3. **High-frequency serving** — At thousands of requests per second, per-request latency compounds. Faster memory increases throughput without adding GPUs [7].

Limitations

Speed is expensive. HBM costs significantly more than DDR per GB [4]. SRAM-based solutions offer low latency but cannot store large models or long contexts without frequent offloading [2]. For training and batch processing, raw speed matters less than total capacity [1].

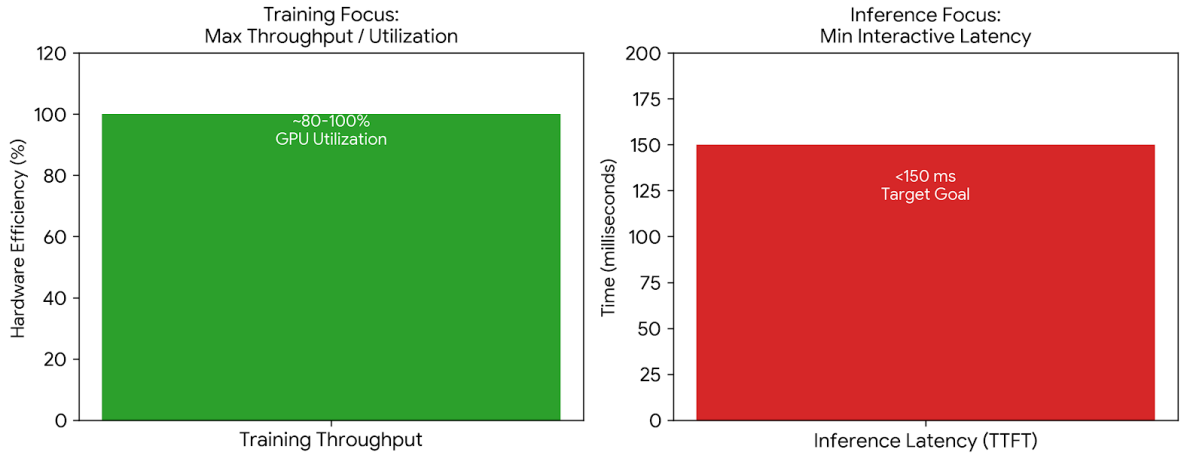

Figure 2: Speed Bottleneck (Throughput vs. Latency). Training focuses on maximizing hardware throughput and utilization, whereas inference prioritizes minimizing user-facing latency (TTFT} < 150ms)[1], [2].

Page 4: Factor 3 — Environmental and Social Impact

Datacenters do not operate in isolation. They consume water, generate heat, produce noise, create e-waste, and affect neighboring communities [4][8]. Technically optimal solutions that ignore these factors risk regulatory pushback, community protests, and eventual shutdown.

Environmental Costs of Memory Choices

Factor	Impact	Memory-Specific Concern
Carbon	Manufacturing HBM has high embodied carbon (3x per GB vs DDR)	Capacity-focused techniques (more DDR) have lower carbon per GB [4][8]
Water	Cooling data centers consumes millions of gallons annually	Denser memory (HBM) requires more aggressive cooling [4]
Noise	Fans and cooling systems generate 60-75dB continuous noise	Community complaints have shut down datacenters [8]
E-waste	Rapid hardware refresh cycles (2-3 years for GPUs)	Memory modules contribute to toxic waste [8]
Vibration	Constant fan vibration loosens connections, causes failures	Military-grade (MIL-STD-810) hardware exists but is rarely used [8]

Social and Regulatory Constraints

Communities near datacenters have protested noise, water usage, and land use [8]. In drought-prone regions, evaporative cooling faces restrictions. European regulations (EU Green Deal) and corporate carbon reporting (SEC, CSRD) now require sustainability metrics [4][8]. A memory architecture that ignores these factors may be technically optimal but operationally impossible.

Implications for Memory Design

Engineers must model not just total cost of ownership (TCO), but social license to operate [8]. This means:

- Choosing greener memory options (DDR over HBM) where performance allows [4]
- Locating capacity-heavy workloads in regions with renewable energy and abundant water [4]
- Investing in noise reduction and vibration isolation for dense clusters [8]
- Planning for hardware reuse and recycling to reduce e-waste [8]

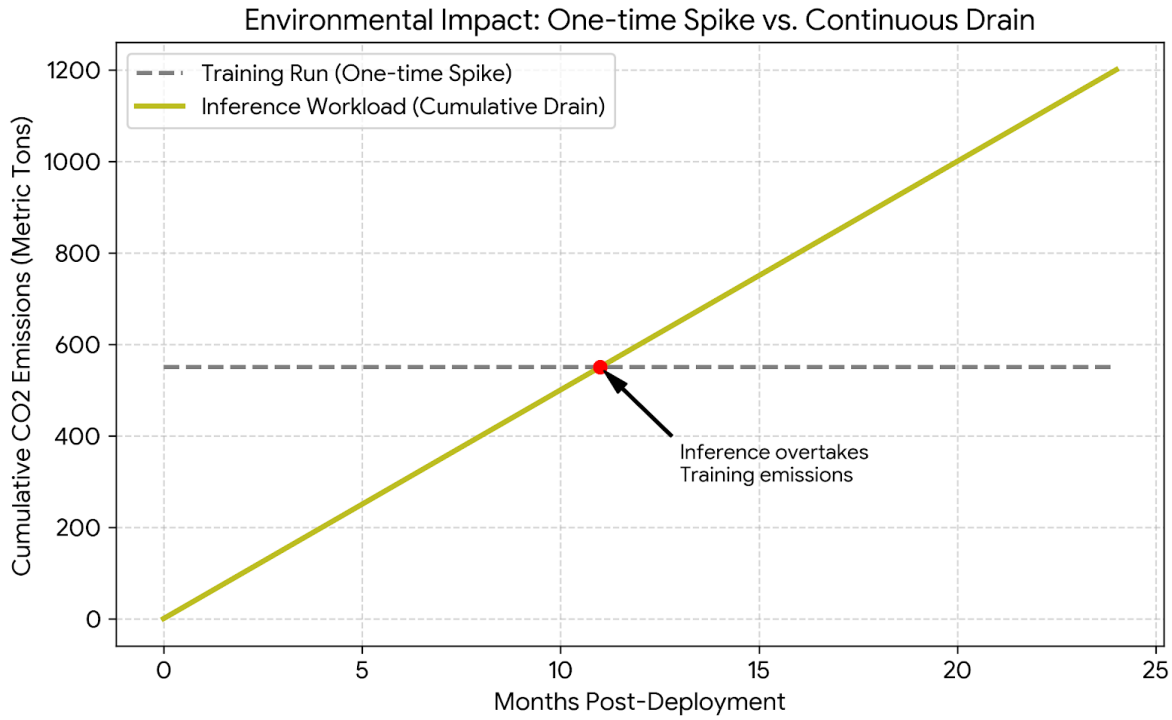

Figure 3: Environmental Impact Over Time. A single frontier training run creates a sharp carbon spike (550 tons), but continuous, high-scale daily user inference creates a compounding infrastructure footprint that quickly overtakes it over time [4]

Page 5: Conclusion and Recommendations

No single memory strategy wins for all workloads. Capacity, speed, and environmental impact must be balanced based on use case, deployment context, and regulatory environment [1][4][8].

Decision Matrix

Workload	Priority	Recommended Focus
LLM Training	Capacity	CXL pooling, tiered memory, CPU offloading [1][3]
Batch Inference	Capacity	KV cache optimization, memory pooling [7]
Real-time Inference	Speed	SRAM/near-memory, HBM, PagedAttention [2]
Agentic AI	Speed	Low-latency memory, efficient KV cache [2]
Sustainable / Green	Environment	DDR over HBM, renewable cooling, e-waste planning [4][8]
Edge Deployment	Hybrid	Balance all three with hardware constraints [2][8]

Final Recommendations

1. **For training and batch workloads** — Prioritize capacity. Use tiered memory and offloading. Accept higher latency in exchange for fitting the model [1][3].

2. **For real-time and agent workloads** — Prioritize speed. Invest in HBM or near-memory compute. Accept lower density and higher cost for lower latency [2].

3. **For all deployments** — Model environmental impact. Datacenters that ignore community and regulatory constraints risk shutdown, regardless of technical performance [4][8].

Conclusion

Memory efficiency for LLMs is not just a technical problem. It is a systems problem involving capacity, speed, and environmental sustainability [1][4][8]. Engineers who consider all three factors will build systems that perform well, operate continuously, and earn social license to scale.

REFERENCES:

[1] K. Tian, L. Qiao, B. Liu, G. Jiang, and D. Li, "A Survey on Memory-Efficient Large-Scale Model Training in AI for Science," *arXiv:2501.11847*, 2025.
[2] K. Shao et al., "DIRC-RAG: Accelerating Edge RAG with Robust High-Density and High-Loading-Bandwidth Digital In-ReRAM Computation," *arXiv:2510.25278*, 2025.
[3] M. Jung et al., "ScalePool: Hybrid XLink-CXL Fabric for Composable Resource Disaggregation in Unified Scale-up Domains," *arXiv:2510.14580*, 2025.
[4] N. Jegham, M. Abdelatti, C. Y. Koh, L. Elmoubarki, and A. Hendawi, "How Hungry is AI? Benchmarking Energy, Water, and Carbon Footprint of LLM Inference," *arXiv:2505.09598*, 2025.
[5] A. Ranganath, "Navigating LLM Valley: From AdamW to Memory-Efficient and Matrix-Based Optimizers," *arXiv:2605.09176*, 2026.
[6] K. Kiyawat et al., "Sangam: Chiplet-Based DRAM-PIM Accelerator with CXL Integration for LLM Inferencing," *arXiv:2511.12286*, 2025.
[7] "Evaluating CXL Memory Pooling for Scalable LLM Inference," *2025 IEEE 32nd International Conference on High Performance Computing, Data and Analytics Workshop (HiPCW)*, pp. 75-79, 2025.
[8] "A Deployment-Aware Framework for Carbon- and Water-Efficient LLM Serving," *Sustainability*, vol. 17, no. 23, p. 10473, 2025.