Reimagining LLM Inference Infrastructure with Memory-Centric KV Cache Servers

Khyati Kiyawat
khyati@virginia.edu
University of Virginia

Kevin Skadron
skadron@virginia.edu
University of Virginia

Abstract

Data centers are building GPU warehouses to serve a memory problem. As generative AI workloads shift from batch training to interactive, long-context, and agentic inference, the dominant bottleneck has migrated from floating-point throughput to KV cache capacity and bandwidth. Production deployments report KV cache reuse rates of 50–90% [1], yet infrastructure investments remain anchored to compute-centric accelerators whose memory is expensive, and capacity-limited. We argue that a *KV cache server*— disaggregated, CXL-attached memory device(s) with optional near-memory compute—is the correct infrastructure primitive for the next generation of AI data centers. We propose a spectrum of CXL-attached memory devices—from CXL DDRx expanders, through CXL-PNM, to CXL-PIM+PNM archetypes—as the substrate for a KV cache server infrastructure. Our back-of-envelope analysis grounds this vision for cost- and energy-efficient inference. We advocate architecting AI data centers around *memory*, not FLOPs.

Keywords: KV Cache, Memory-centric AI, CXL, PIM, PNM

1 The Inference Infrastructure Mismatch

LLM inference is split into compute-bound *prefill* and memory-bound *decode*. As context lengths and model sizes grow, the bottleneck has shifted decisively to memory. The KV cache grows rapidly: at 128K context, LLaMA-3-70B requires 34 GB per request; combined with 140 GB model weights, the total far exceeds the 80 GB HBM of a single H100, requiring multi-GPU sharding before a token is generated (Figure 1).

Conventional memory scaling hits a physical wall. Adding memory to the compute die is blocked by die *beachfront*: the H100 integrates six HBM3 stacks; a seventh stack requires either a reticle-limited larger GPU die or taller 3D stacked HBM with prohibitive thermal penalties. CPU DRAM faces an analogous *pin-count wall*: DDR5 channels consume ~100 pads each, leaving no socket headroom for growth. CXL dissolves both constraints with a single ×16 link, scaling capacity (TBs) linearly by adding memory modules with no die-area or pin-count tax [2–4].

Decode length explosion. Azure 2023 traces show median output lengths of 100–200 tokens [5]; by 2025, reasoning models such as DeepSeek-R1 and OpenAI o1 routinely generate 8,000–32,000 tokens [6]—a **40–160× increase in KV cache pressure in under two years**. Each decode step is a memory-bound operation (OI ≈ 0.5 ops/byte, 600× below the H100 ridge point), so the GPU spends >70% of decode time waiting for memory [3, 7].

KV reuse is the dominant optimization lever. Production deployments report KV reuse rates of 50–90% [1], with 75–95% in multi-agent workloads [8]; cross-engine KV sharing alone delivers 15× throughput improvement [9]. In the majority of inference traffic, *the most valuable operation is not computing new KV vectors but retrieving existing ones*—making a fast, high-capacity KV store and transfer the appropriate infrastructure primitive.

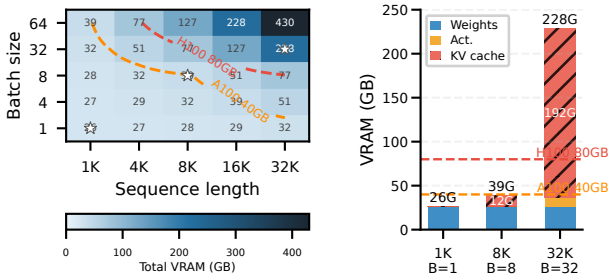

Figure 1. VRAM breakdown for DeepSeek-R1-Distill-Qwen-14B (GQA, BF16) across batch sizes and sequence lengths. KV cache grows linearly with both dimensions, rapidly dominating total memory and exceeding GPU VRAM capacity.

1.1 GPU Data Centers Miss The Memory Bottleneck

The consequence of deploying compute-optimized hardware for a memory-intensive workload is predictable: expensive underutilization. GPU compute utilization collapses to <30% for decode-dominant workloads [3, 7, 10], while HBM capacity is simultaneously exhausted by growing KV caches, causing out-of-memory failures that cap batch size and throughput [7, 11, 12]. The GPUs (or other xPUs such as TPUs, NPUs, and LPUs) are both memory bandwidth-bound and capacity-bound, yet neither constraint is addressable on-die.

The following *inference regimes* expose this mismatch: (i) **decode-heavy reasoning** — long outputs, short inputs, GPU compute idles while HBM bandwidth saturates; (ii) **high KV reuse** (multi-turn, CAG [13], RAG [14]) — most KV is

The 3rd Workshop on Hot Topics in System Infrastructure (HotInfra'26), co-located with ISCA'26, June 28, 2026, Raleigh, NC, USA

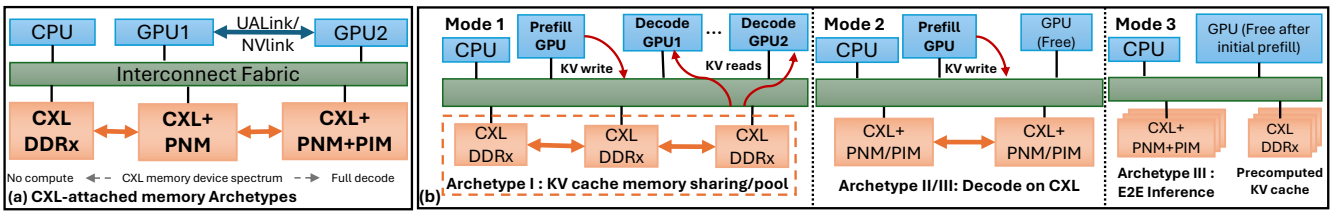

Figure 2. (a) KV cache server system integration and CXL-attached memory Archetypes. (b) Operating modes and archetype mix for different usecase scenarios.

already computed; (iii) **multi-user sharing, agentic clusters** — Multiple agents/users share a system prompt KV that a CXL pool stores once, while each agent/user maintains its own private KV on its dedicated memory node.

The energy and cost toll. Serving DeepSeek-R1-671B with 32K reasoning tokens requires 19 H100 GPUs at only ~15% compute utilization i.e., you pay for 18,791 TFLOPS to serve a bandwidth-limited workload. Section 3 quantifies this.

2 The KV Cache Server

We propose that the *KV cache server* should be a first-class data center infrastructure tier, positioned between GPU compute nodes and storage—just as object stores (S3 [15]) once decoupled storage from compute. We identify three *device archetypes* along the CXL-attached memory spectrum (Figure 2a); real deployments may instantiate any single archetype or blend characteristics of multiple archetypes[1]: (i) **CXL DDRx Expanders** – KV storage and multi-host pooling/sharing, no near-memory compute [2, 16]; (ii) **CXL+PNM** – offloads orchestration, token selection [17], or some memory-bound operations [18, 19]; (iii) **CXL+PNM+PIM** – enables end-to-end LLM inference, decoding or attention execution [3, 20, 21]. The operating mode is selected based on the archetype(s) deployed and the workload, enabling: inference with/without GPU [3, 20]; CXL 3.0 pooled KV accessible to multiple hosts via load/store without RDMA; and native multi-tenant KV routing for multi-agents/users. These archetypes can be connected through a futuristic memory-to-memory interconnect fabrics [22–26], like UAlink/NVlink connects accelerator nodes, enabling KV handoff between CXL devices without routing through a compute die.

Hierarchical orchestration. The CPU acts as *global coordinator*; for independent tasks, orchestration is offloaded to a *CXL Fabric Manager* (switch-level) or *local coordinators* in CXL-attached memory device controllers, resuming CPU involvement only at dependency frontiers. The device archetype determines how much it can absorb: a CXL DDRx expander defers all scheduling to the CPU; a CXL+PNM+PIM node handles full/partial operations locally, signaling only completion to the CPU.

[1]DDRx denotes DDR4, DDR5, and future standards (DDR6, LPDDR6); the architecture is generation-agnostic.

Usecase scenarios. Figure 2b illustrates three representative deployment scenarios; real systems may blend archetypes.

Scenario 1 (KV cache sharing with GPU decode): Targets prefill-decode disaggregated deployments where multiple users or agents share a common system prompt [27]. The prefill GPU computes the KV cache and writes it *once* to the Archetype I (CXL DDRx) pool; one or more decode GPUs then read the shared KV directly via CXL load/store—no per-user HBM replication, no RDMA copies. This is ideal for RAG, multi-turn chat, and shared system-prompt deployments where decode batch sizes remain sufficiently large to justify GPU-class compute.

Scenario 2 (CXL-PNM/PIM decode): Targets decode-heavy workloads with small batch sizes where GPU compute is underutilized during decode. The GPU performs prefill and writes the KV cache to the Archetype I pool; Archetype II/III (CXL+PNM or CXL+PNM+PIM) devices execute all subsequent decoding via GEMV/flat-GEMM on their PIM units [3, 20], freeing the GPU for other workloads. Shared prompt KV can be transferred from Archetype I to Archetype II/III devices directly via custom memory-to-memory interconnects [22, 24, 25].

Scenario 3 (End-to-end on CXL): Targets near-zero-prefill workloads or maximum KV prefix reuse, where the GPU executes prefill *once*, writes the KV cache to the Archetype I pool, and is then fully released for other workloads. An Archetype III (CXL+PNM+PIM) device executes end-to-end inference—including any incremental prefill when needed—entirely within the memory node. Sangam [3] and CENT [20] demonstrate order-of-magnitude energy reduction versus GPU for this scenario.

Agentic AI with the KV cache server. Agentic deployments can be viewed as a composition of all three scenarios. Consider multiple concurrent agents: Models reside either on GPUs or on Archetype II/III devices based on the prefill-to-decode ratio and model size. The CPU acts as orchestrator, decomposing the tasks and assigning sub-tasks. The shared system prompt KV is written *once* to the Archetype I pool and read by all agents; each agent's private KV cache resides on its local memory node. CXL devices execute independently and in parallel until a *dependency frontier* is reached, at which point agents either coordinate through the CPU or share KV data directly via memory-to-memory interconnects [22, 23].

Table 1. 32K tokens generation with DeepSeek-R1-671B

	GPU cluster[2]	KV cache server
Devices	19× H100 SXM5 (80 GB HBM each)	CPU, 2× CXL switches, 23× 64 GB PIM-DIMMs
Total memory	1,520 GB	1,472 GB
Aggregate Bandwidth	63.7 TB/s	150.7 TB/s (2.4×↑)
Throughput	679 tok/s	1,607 tok/s (2.4×↑)
CapEx (purchase)	$570,000	$27,664 (20.6×↓)
OpEx (cloud rental+elec.)	$59.23/hr	$3.53/hr (16.8×↓)
Power (w/ PUE)	18.6 kW	1.49 kW (12.5×↓)
Tokens/Watt	0.051	**1.507 (29.5×↑)**
Cost/Mtokens	24.24 $	**0.61 $ (39.7×↓)**

KV cache server's CXL switches have 16 channels on SW1 and 7 on SW2.
SW2 has 9 free channels (576 GB) for other hosts. Mtokens: million tokens.

3 DeepSeek-R1-671B Inference Study

We evaluate a typical reasoning token generation case (32K output tokens) for a large DeepSeek model on a GPU cluster and a CXL+PIM+PNM archetype KV cache server. DeepSeek-R1-671B uses Mixture-of-Experts (MoE) with 37B active parameters per token [6] and Multi-head Latent Attention (MLA), which compresses the KV cache to a latent dimension d_{latent}=512 per layer—approximately 64× smaller than standard Multi-head Attention (MHA). Memory traffic per token at average sequence position $\bar{S} = S_{\max}/2$ is:

$$B_{\text{tok}} = \underbrace{W_{\text{active}} \cdot 2^{30}}_{\text{37B active weights: 74 GB}} + \underbrace{L \cdot d_{\text{latent}} \cdot \bar{S} \cdot \text{BPE}}_{\text{MLA KV: 1.02 GB}} = 75\ \text{GB/token} \tag{1}$$

where BPE= 2 (BF16), L=61 layers, $\bar{S}$=16K tokens. Decode throughput, cost-per-token, and tokens-per-watt follow as:

$$\tau = \frac{\eta \cdot BW_{\text{agg}}}{B_{\text{tok}}}, \quad C = \frac{\text{OpEx}}{\tau \cdot 3600}, \quad \text{tok/W} = \frac{\tau}{P} \tag{2}$$

where η=0.80 bandwidth utilization and P is total power. Aggregate internal PIM bandwidth across N_D DIMMs is:

$$BW_{\text{PIM}} = \frac{N_D \cdot N_{\text{banks}} \cdot (\text{GDL}_{\text{bits}}/8) \cdot N_{\text{chips}} \cdot N_{\text{ranks}}}{t_{\text{CCD\_l}}} \tag{3}$$

with N_{banks}=32, GDL_{bits}=128 b (16 B), N_{chips}=16, N_{ranks}=2, $t_{\text{CCD\_l}}$=2.5 ns (DDR5-4800), giving 6.55 TB/s per DIMM.

Table 1 presents the hardware configuration and results. *Assumptions:* (i) PIM bandwidth is internal bank-level BW for in-situ compute, not CXL I/O bandwidth [3]; (ii) PIM-DIMM cost ($12/GB) at 2-4× commodity DDR5; (iii) KV cache server CapEx and Power includes 23 PIM-DIMMs ($768, 30W each), 2 CXL switches ($3000, 13W each), and a CPU ($4000, 350W); (iv) operational power includes electricity at $0.12/kWh with PUE (Power Usage Effectiveness) = 1.4; (v) GPU cluster cloud rental uses H100 on-demand pricing at $3.00/hr [28], while KV cache server rental is scaled proportionally to its 20.6× lower CapEx. This analysis models a steady-state, decode-dominant regime where one-off prefill overheads of the GPU are amortized over long sequences.

4 Related Works

KV cache serving systems. vLLM [11], SGLang [8], LM-Cache [9], and Mooncake [1] manage KV cache in software, delivering significant throughput improvements. These software techniques can naturally extend to manage allocation, eviction, and cost-effective CPU/SSD offloading fallbacks across our tiered CXL memory archetypes. Future work must investigate the optimal placement of these archetypes within a tiered KV Cache storage hierarchy to maximize efficiency.
CXL memory for KV offload. TraCT [2] and Beluga [16] use CXL as a passive KV transfer substrate but rely on the GPU for all compute. CXL-NDP [19] adds controller-level bandwidth amplification. Kim et al. [17] demonstrate CXL-PNM token page selection at 1M-token contexts (21.9× throughput, 60× lower energy), exemplifying the CXL+PNM archetype. None provide flat-GEMM capability or native multi-tenant KV sharing.
PIM and near-memory compute. L3 [21], AttAcc [12], and NeuPIMs [7] offload decode attention to DIMM-PIM or HBM-PIM but remain capacity-constrained and GEMV-only. CENT [20] demonstrates GPU-free CXL-PIM inference; Sangam [3] extends this with chiplet DDR5-PIM and systolic arrays, serving as our reference architecture (Section 3).

No prior work frames the KV cache server as a first-class infrastructure tier with a substrate-agnostic CXL archetype spectrum, mode-adaptive operation, and hierarchical orchestration for different usecase scenarios. A comprehensive HW/SW stack study is required to characterize its coordination with existing GPU-HBM serving systems.

5 Conclusions

Deploying TFLOP-optimized GPU infrastructure for memory-bound LLM decode leads to massive underutilization and high operational costs. 23 CXL PIM-DIMMs serving DeepSeek-R1-671B at 32K tokens deliver 20.6× lower CapEx, 12.5× lower power, 2.4× more bandwidth, and 39.7× lower cost/Mtokens than 19 H100s doing the same job.

We call on the community to: 1) Treat the KV cache as a first-class, shared data structure and build the infrastructure with dedicated hardware. 2) Build memory-native fabric by extending accelerator fabrics (NVLink, UALink) to connect memory nodes directly creating *distributed KV cache fabrics*.

The KV cache server is not a niche accelerator. It is the correct answer to: *what is the minimum hardware needed to generate the next token?* For the dominant fraction of production inference traffic, the answer is a memory device with modest near-memory compute—not a $30,000 GPU.

Acknowledgments

This work was supported in part by PRISM, one of seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

[2]Cost and Power of CPU and NV Switches are not included.

References

[1] Ruoyu Qin, Zheming Li, Weiran He, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: A KVCache-centric disaggregated architecture for LLM serving. In *Proc. USENIX FAST*, 2025. arXiv:2407.00079.

[2] Dongha Yoon et al. TraCT: Disaggregated LLM serving with CXL shared memory KV cache at rack-scale. *arXiv preprint arXiv:2512.18194*, 2025.

[3] Khyati Kiyawat, Zhenxing Fan, Yasas Seneviratne, Morteza Baradaran, Akhil Shekar, Zihan Xia, Mingu Kang, and Kevin Skadron. Sangam: Chiplet-based DRAM-PIM accelerator with CXL integration for LLM inferencing. *arXiv preprint arXiv:2511.12286*, 2025.

[4] CXL Consortium. Compute express link (CXL) 4.0 specification. https://computeexpresslink.org, 2025. Released November 18, 2025.

[5] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. In *Proc. ACM/IEEE ISCA*, 2024.

[6] DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[7] Guseul Heo, Sangjun Lee, Jaehong Cho, Hyunjoon Choi, Sanghyeon Lee, Hyungkyu Ham, Gwangsun Kim, Divya Mahajan, and Jongse Park. NeuPIMs: NPU-PIM heterogeneous acceleration for batched LLM inferencing. In *Proc. ACM ASPLOS*, 2024.

[8] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[9] Yihua Cheng, Yuhan Liu, Jiayi Yao, Yuwei An, Xiaokun Chen, Shaoting Feng, Yuyang Huang, Samuel Shen, Kuntai Du, and Junchen Jiang. LMCache: An efficient KV cache layer for enterprise-scale LLM inference. *arXiv preprint arXiv:2510.09665*, 2025.

[10] Yunho Jin, Chun-Feng Wu, David Brooks, and Gu-Yeon Wei. S3: Increasing GPU utilization during generative inference for higher throughput. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 18015–18027, 2023.

[11] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proc. ACM SOSP*, 2023.

[12] Jaehyun Park, Jaehee Choi, Kyuyoung Kim, Michael Jaemin Kim, Yongsuk Kwon, Nam Sung Kim, and Jung Ho Ahn. AttAcc! unleashing the power of PIM for batched transformer-based generative model inference. In *Proc. ACM ASPLOS*, 2024.

[13] Brian J Chan, Chao-Ting Chen, Jui-Hung Cheng, and Hen-Hsen Huang. Don't do rag: When cache-augmented generation is all you need for knowledge tasks. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 893–897, 2025.

[14] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[15] Mayur R Palankar, Adriana Iamnitchi, Matei Ripeanu, and Simson Garfinkel. Amazon s3 for science grids: a viable solution? In *Proceedings of the 2008 international workshop on Data-aware distributed computing*, pages 55–64, 2008.

[16] others. Beluga: A CXL-based memory architecture for scalable and efficient LLM KVCache management. *arXiv preprint arXiv:2511.20172*, 2025.

[17] Dowon Kim, Junsoo Kim, Taehyun Kim, Hyungyo Kim, Youngsok Kim, and Jungwook Choi. Scalable processing-near-memory for 1M-token LLM inference: CXL-enabled KV-cache management beyond GPU limits. In *Proc. IEEE International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2025. arXiv:2511.00321.

[18] Lian Liu, Shixin Zhao, Bing Li, Haimeng Ren, Zhaohui Xu, Mengdi Wang, Xiaowei Li, Yinhe Han, and Ying Wang. Make llm inference affordable to everyone: Augmenting gpu memory with ndp-dimm. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 1751–1765. IEEE, 2025.

[19] others. Amplifying effective CXL memory bandwidth for LLM inference via transparent near-data processing. *arXiv preprint arXiv:2509.03377*, 2025.

[20] Yufeng Gu et al. PIM is all you need: A CXL-enabled GPU-free system for LLM inference. In *Proc. ACM ASPLOS*, 2025.

[21] others. L3: DIMM-PIM integrated architecture and coordination for scalable long-context LLM inference. *arXiv preprint arXiv:2504.17584*, 2025.

[22] Zhe Zhou, Cong Li, Fan Yang, and Guangyu Sun. DIMM-Link: Enabling efficient inter-DIMM communication for near-memory processing. In *Proc. IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 302–316. IEEE, 2023.

[23] Hyeonuk Kim, Yongju Lee, Minseok Seo, Jiwon Seo, and Hanjun Kim. Application-transparent near-memory processing architecture with memory channel network. In *Proc. IEEE/ACM MICRO*, 2021.

[24] Weiyi Sun, Zhaoshi Li, Shouyi Yin, Shaojun Wei, and Leibo Liu. ABC-DIMM: Alleviating the bottleneck of communication in DIMM-based near-memory processing with inter-DIMM broadcast. In *Proc. ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 237–250. IEEE, 2021.

[25] Youngeun Kwon, Yunjae Lee, and Minsoo Rhu. TensorDIMM: A practical near-memory processing architecture for embeddings and tensor operations in deep learning. In *Proc. 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 740–753, 2019.

[26] Daichi Fujiki et al. AIM: Accelerating computational genomics through scalable and noninvasive accelerator-interposed memory. In *Proc. ACM ASPLOS*, 2021.

[27] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *Proc. USENIX OSDI*, 2024.

[28] Tim Bradshaw. How inference is driving competition to NVIDIA's AI chip dominance. https://www.ft.com/content/d5c638ad-8d34-4884-a08c-a551588a9a28, 2025. Financial Times, March 2025.