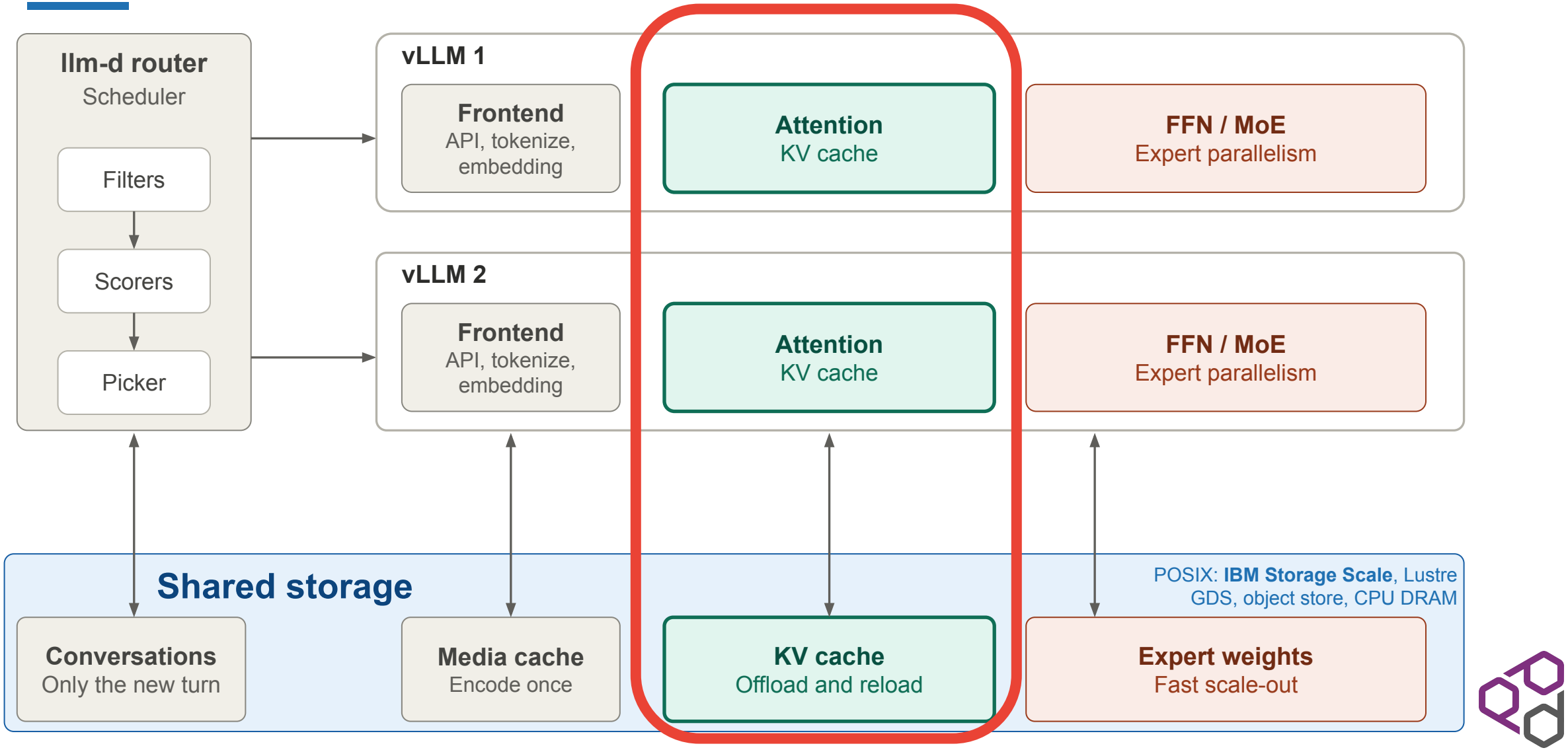
Your KV Cache Is Bigger Than Your GPU: Scaling Agentic Inference with Networked Storage

Kfir Toledo, IBM research
Nishtha Jain, Google Cloud

Cloud Native AI + Inference Day | Salt Lake City, Nov 9, 2026

Google Cloud | IBM | llm-d

Every part of the inference stack needs storage



Agentic Workloads Shift the Bottleneck from compute to memory and Caching

Inference has shifted from 8K context chat to **agentic workloads** with 100K+ contexts and rapid tool-use loops.

Key Data Points

437K
P90 Input Sequence Tokens

258
P90 Session Turns

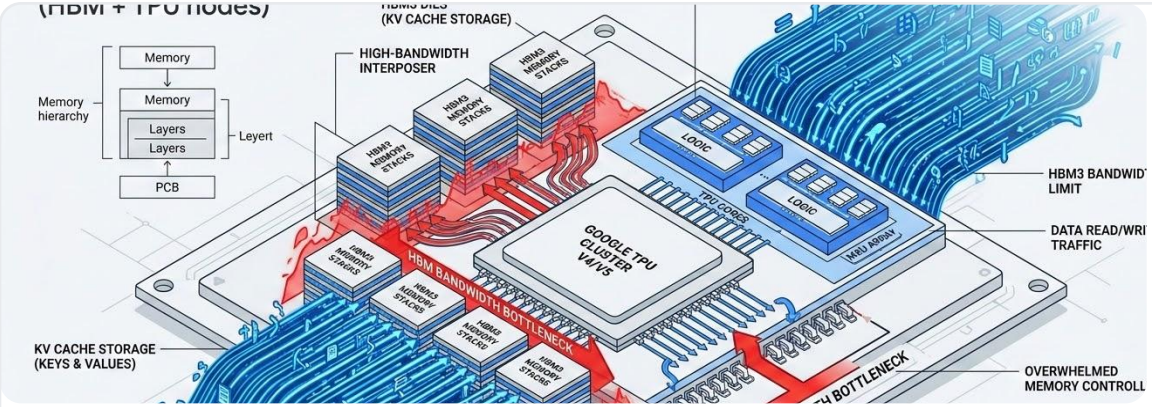
94.3%
Tokens are Cache Reads (436x Ratio)

The Scaling Challenge

Quadratic memory footprint of self-attention exhausts accelerator HBM.

Llama 3 70B (FP16): ~320GB needed for 1M tokens.

DeepSeek R1 (FP8): ~34GB needed for 1M tokens.



THE CORE BOTTLENECK • WHY GPU-ONLY CACHING FAILS

GPU-only KV caches transform evictions into wasted compute

Autoregressive LLM serving relies on the KV cache to avoid quadratic $O(N^2)$ recomputation on every generated token. Yet long-context RAG (50K+ tokens) and multi-turn agents rapidly saturate 80GB GPU HBM—forcing constant LRU evictions while terabytes of host RAM, shared storage, and peer memory sit idle.

The largest KV cache tier already exists in your cluster—in CPU DRAM, shared storage, and peer pods

ARCHITECTURAL PARADIGM SHIFT • WHY OFFLOADING MATTERS

Tiered KV cache offloading transforms isolated GPU bottlenecks into elastic fleet memory

POINT A • GPU-ONLY KV CACHE

HBM Eviction & Recompute Storms

Strict HBM Capacity Ceiling: Active KV working set cannot exceed local GPU VRAM after subtracting model weights.

Multi-Turn Thrashing: Think-act-observe agent pauses trigger LRU eviction, forcing full prefill recomputation.

Siloed Pod Islands: Round-robin load balancers send follow-up turns to cold pods, destroying prefix hit rates.

POINT B • LLM-D TIERED OFFLOADING

Persistent Context Across Memory Tiers

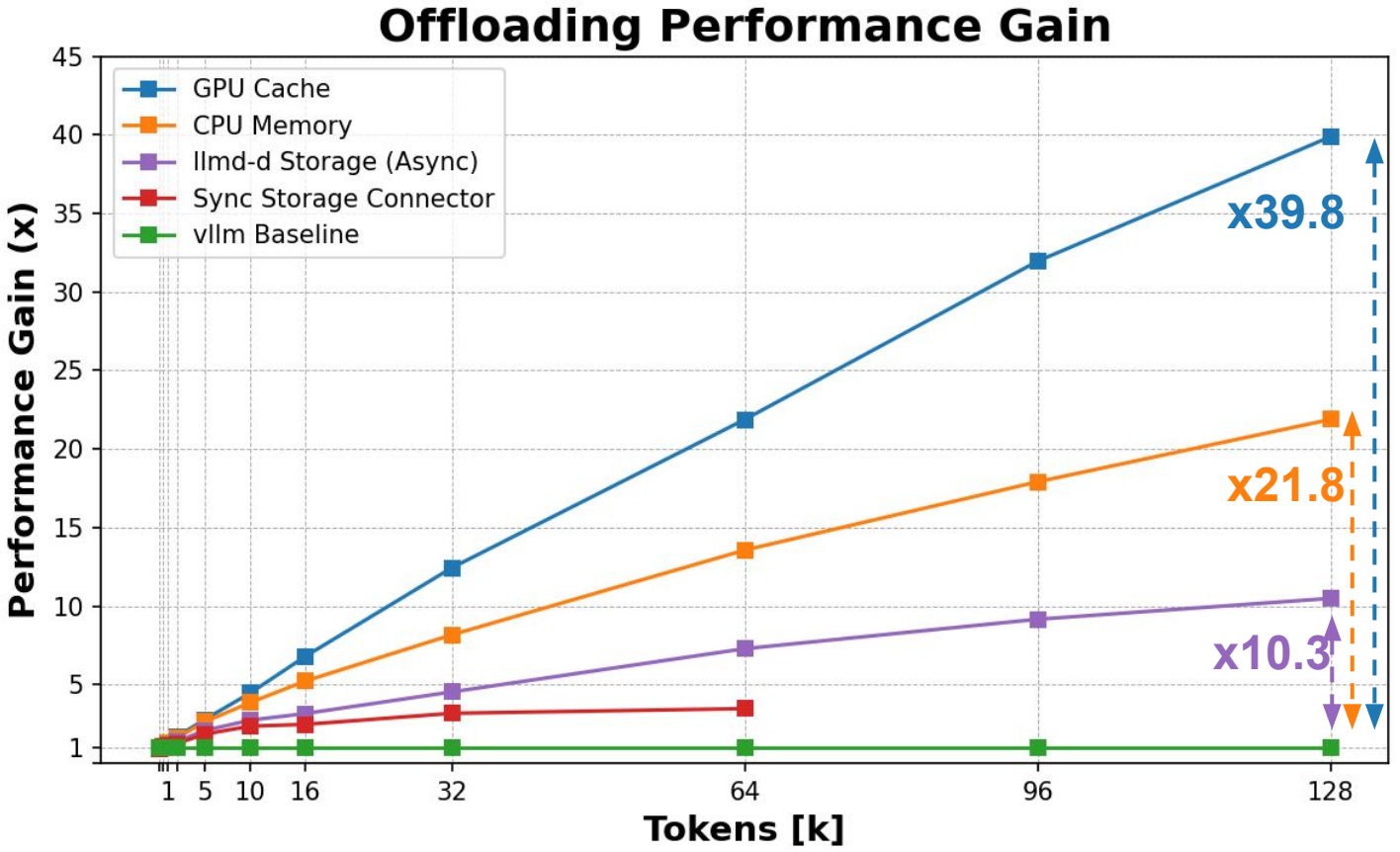
Prefill Bypass via DMA Read: Transferring KV blocks from CPU RAM or storage is orders of magnitude cheaper than recomputing attention.

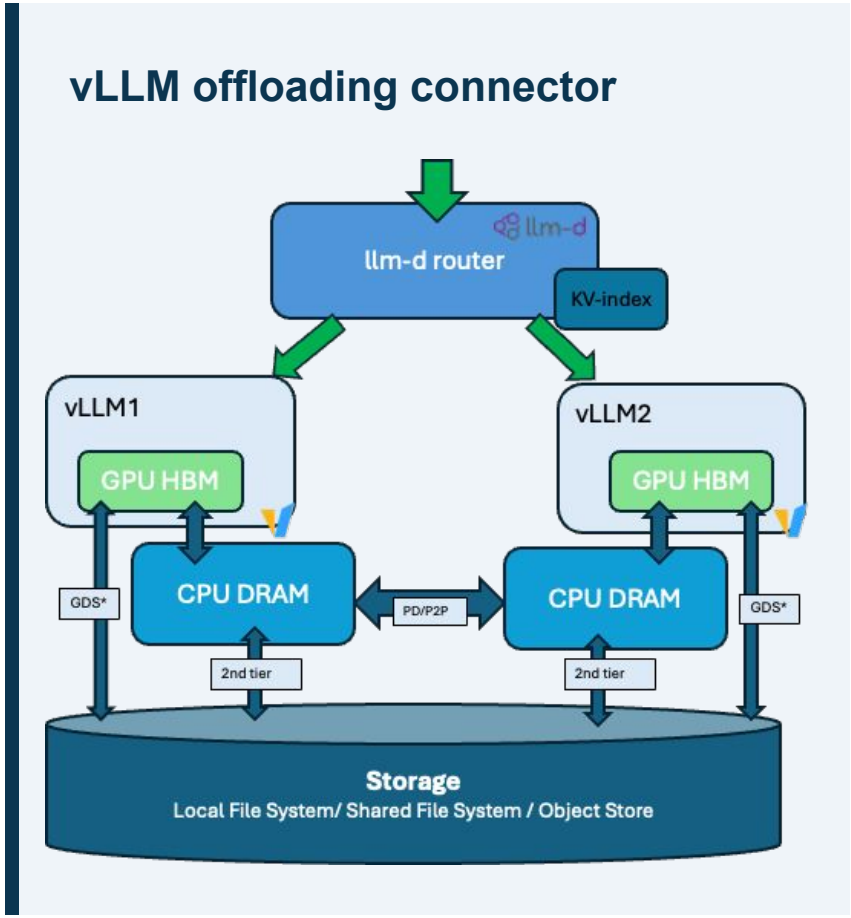
Decoupled Working Set: Petabyte-scale shared storage retains millions of system prompts, RAG chunks, and agent histories.

Fleet-Wide Cache Sharing: Any pod can hydrate warm KV blocks from shared storage or peer memory on demand.

SINGLE REQUEST

Even the storage tier is 10x faster than recomputing

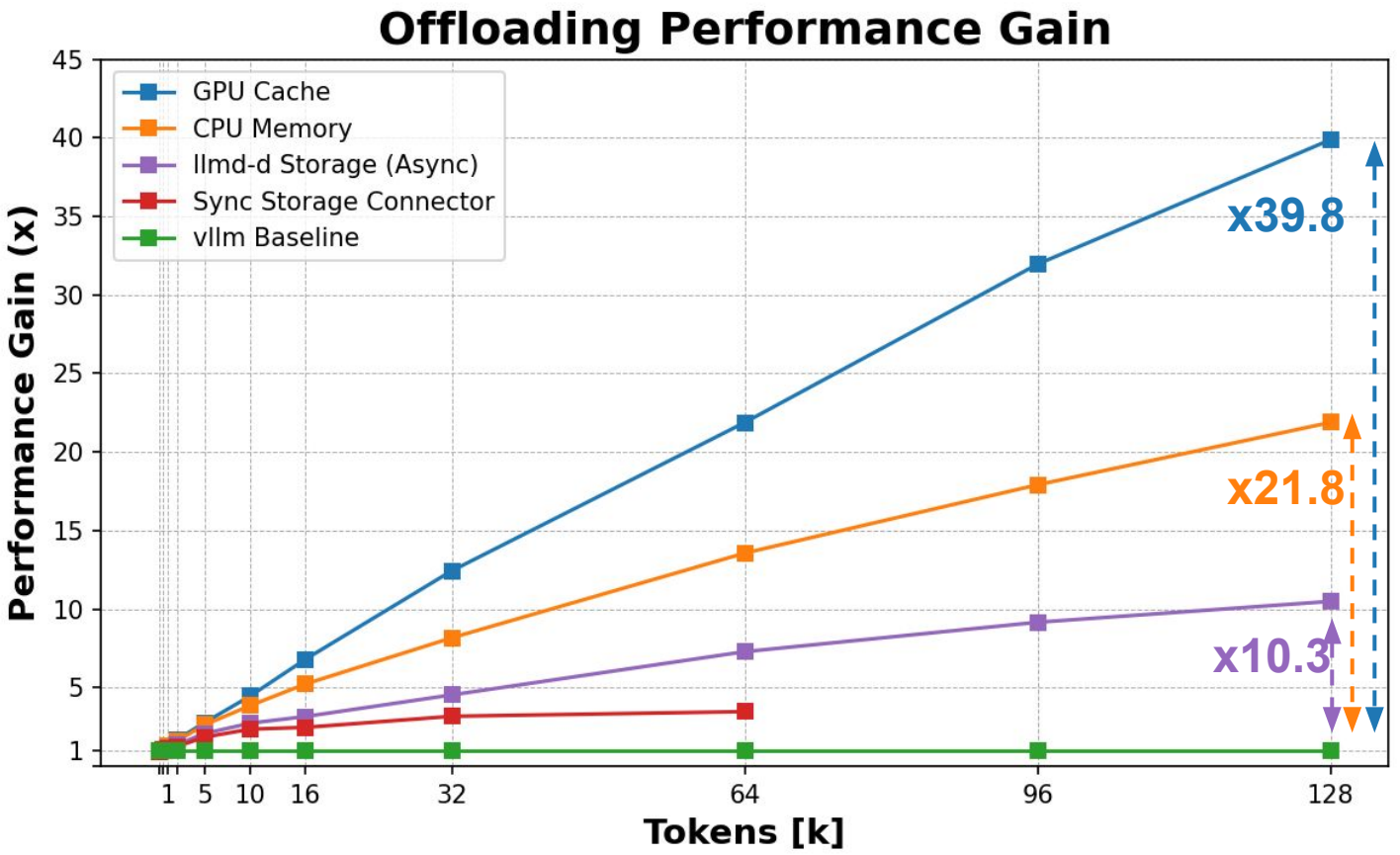




Llama-3.1-70B on NVIDIA H100, single request, **IBM Storage Scale** as the storage tier.
Speedup vs. vLLM recompute baseline, KV cache reused from each tier.

SINGLE REQUEST

Even the storage tier is 10x faster than recomputing



Gains grow with prompt length
KV reuse speedup scales almost linearly with tokens

IBM Storage Scale: 10.3x at 128K
Below CPU (21.8x) and GPU (39.8x), far above recompute

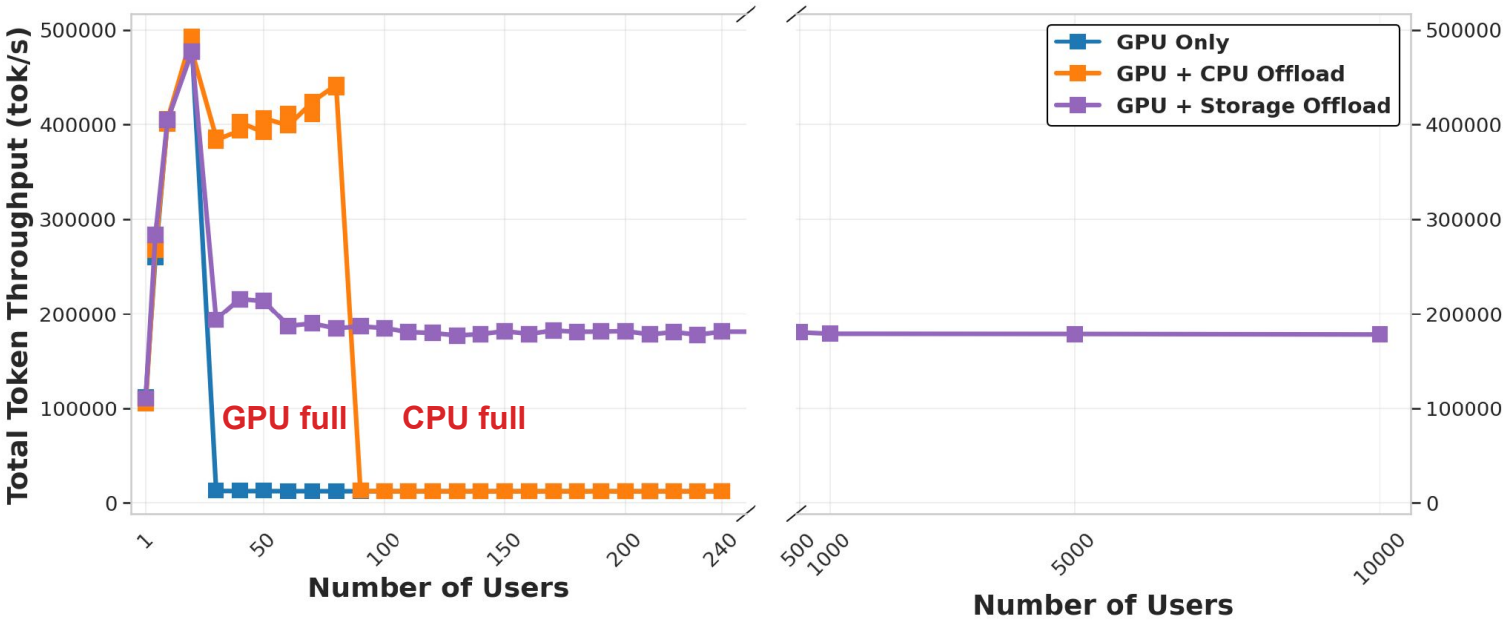
Each hop costs about half
Every tier further from the GPU gives up roughly half the gain

Llama-3.1-70B on NVIDIA H100, single request, **IBM Storage Scale** as the storage tier.
Speedup vs. vLLM recompute baseline, KV cache reused from each tier.

SCALE UP

Storage keeps throughput flat after GPU and CPU saturate

Prefill requests at 100% load: the upper bound of what offloading can deliver



GPU and CPU hit a cliff
Throughput collapses once their KV capacity is full

IBM Storage Scale holds steady
About 180K tok/s, flat out to 10,000 users

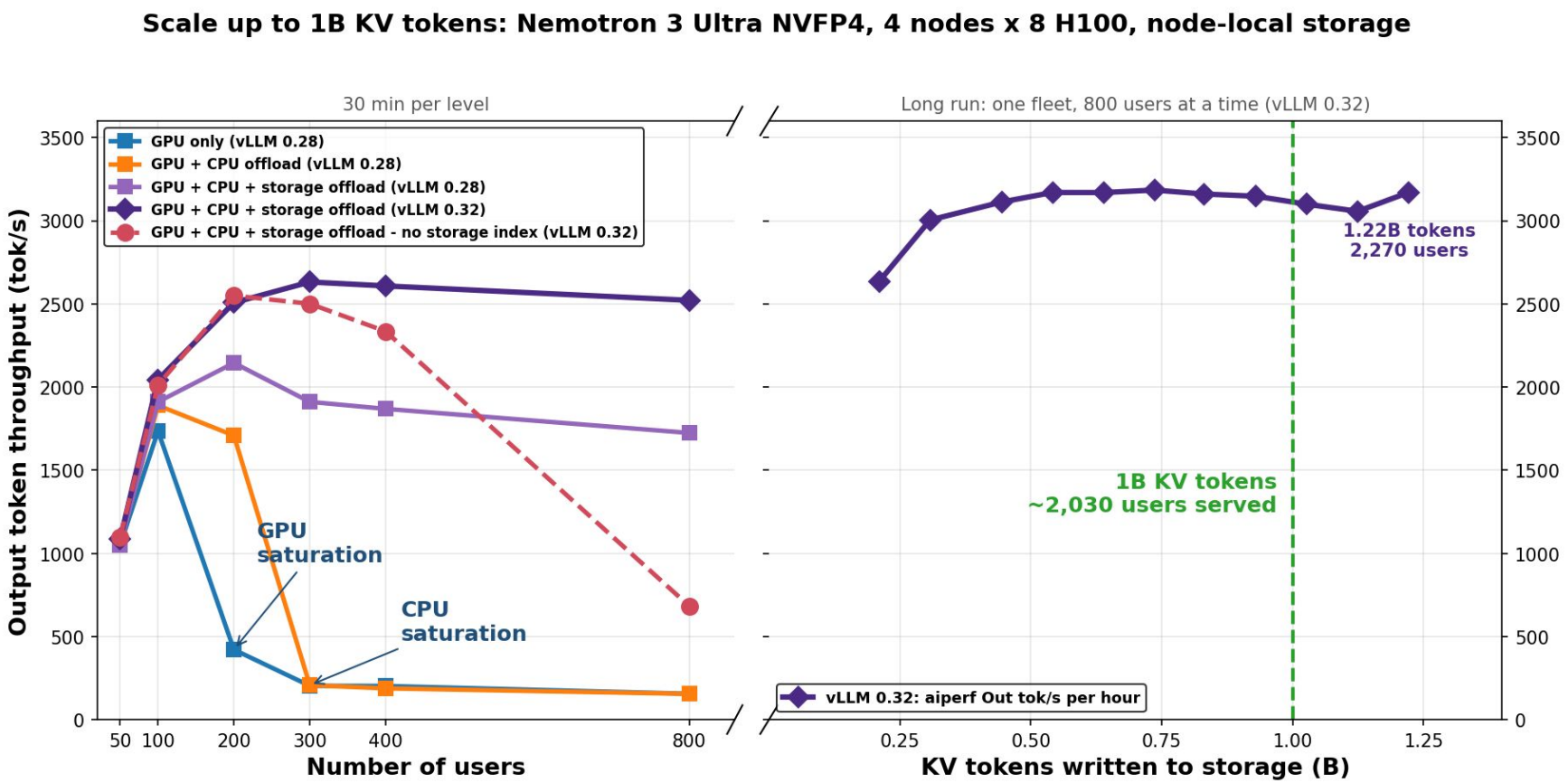
Storage wins at scale
Once GPU and CPU caches are full, only the storage tier keeps serving

Llama-3.1-70B on 4x NVIDIA H100, 100 GB host memory per GPU, I**BM Storage Scale**.
Prefill-only 16K-token requests at 100% load, users added each round.

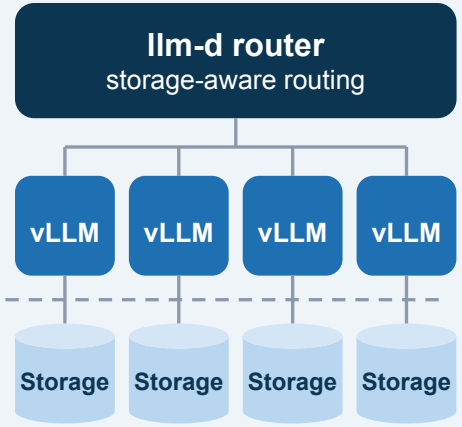
SCALE OUT

Storage keeps throughput flat past 1B KV tokens

Workload: replayed Claude Code agentic traces (Weka trace); pauses between turns (tool runs, user idle) capped at 10 min



llm-d system



local storage on each node

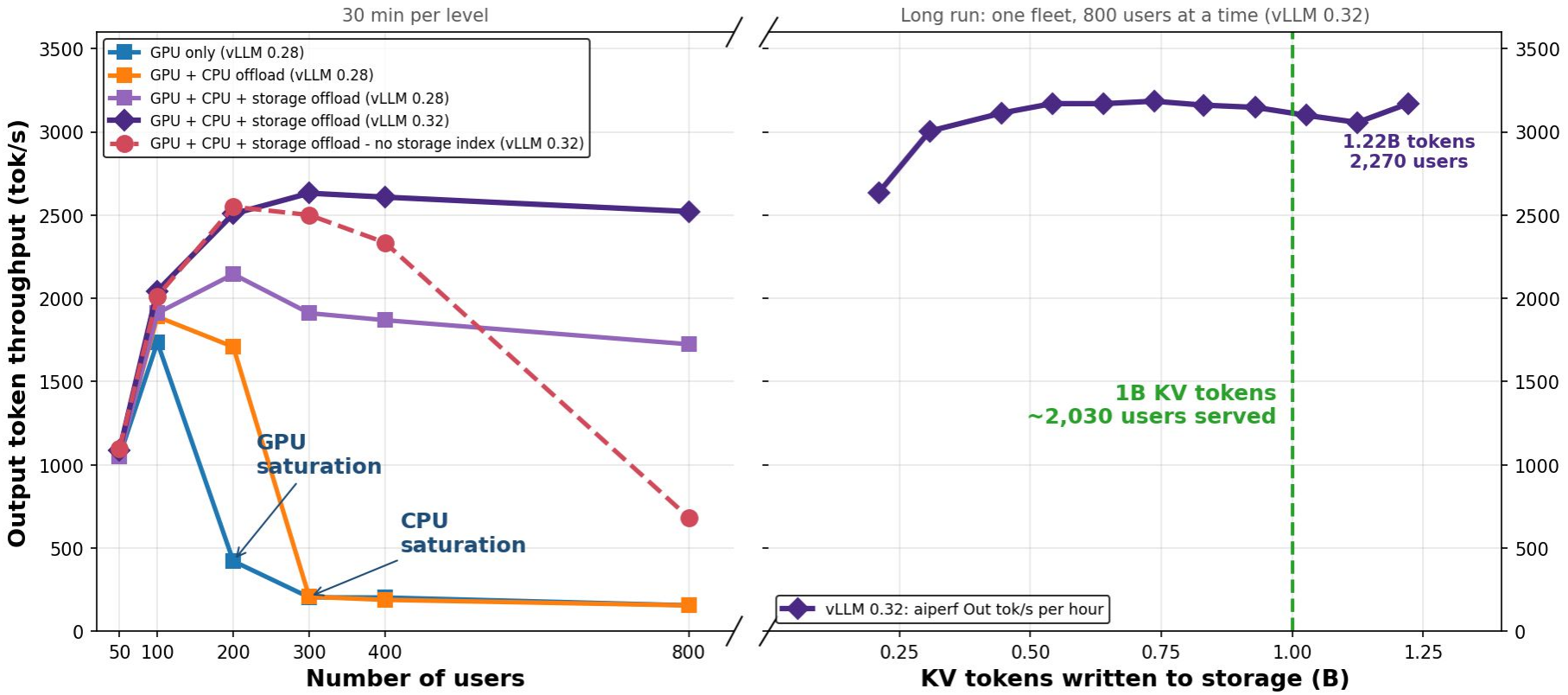
The router sends each turn to the node that already holds its KV

NVIDIA-Nemotron-3-Ultra-550B (NVFP4), 4 nodes x 8 H100, node-local storage.
Left: 30 min per load level. Right: one long run, 800 users at a time.

SCALE OUT

Storage keeps throughput flat past 1B KV tokens

Workload: replayed Claude Code agentic traces (Weka trace); pauses between turns (tool runs, user idle) capped at 10 min



Storage lets you scale
Throughput stays flat past 1B KV tokens

Know your workload
Agentic pauses decide how long KV must live

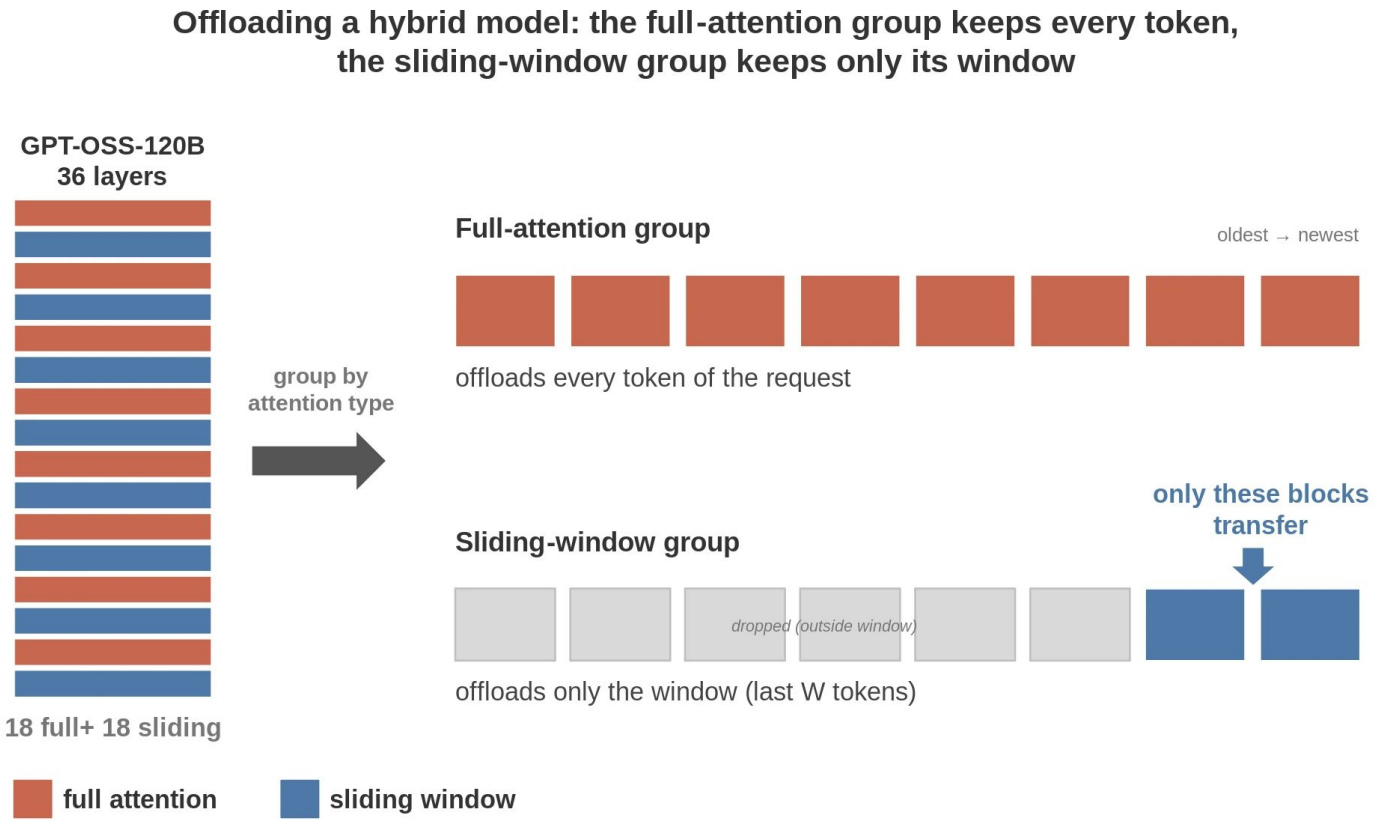
The router matters
No storage index: ~2,500 to ~700 tok/s at 800 users

HMA support
~2,500 vs ~1,700 tok/s at 800 users

NVIDIA-Nemotron-3-Ultra-550B (NVFP4), 4 nodes x 8 H100, node-local storage. Left: 30 min per load level. Right: one long run, 800 users at a time.

HYBRID MODELS

Hybrid models: offload only what each layer needs



Two cache groups
gpt-oss-120b: full attention keeps every token, sliding window keeps the last 128

~2x more KV space
Sliding-window layers store only 128 tokens, so each tier fits about twice the users

Loads move less data
Only in-window blocks transfer for sliding-window layers

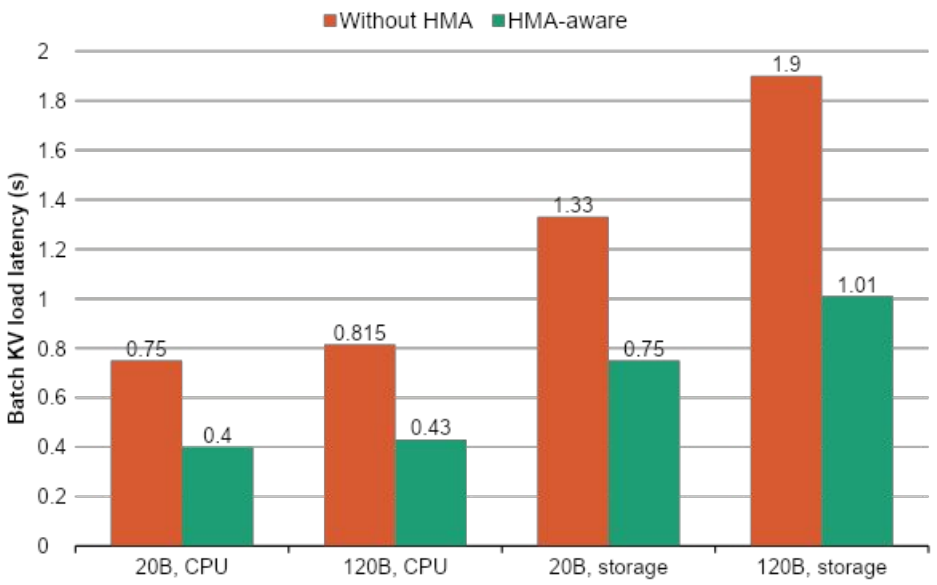
Old connectors turn HMA off
Offload works, but the memory savings are lost

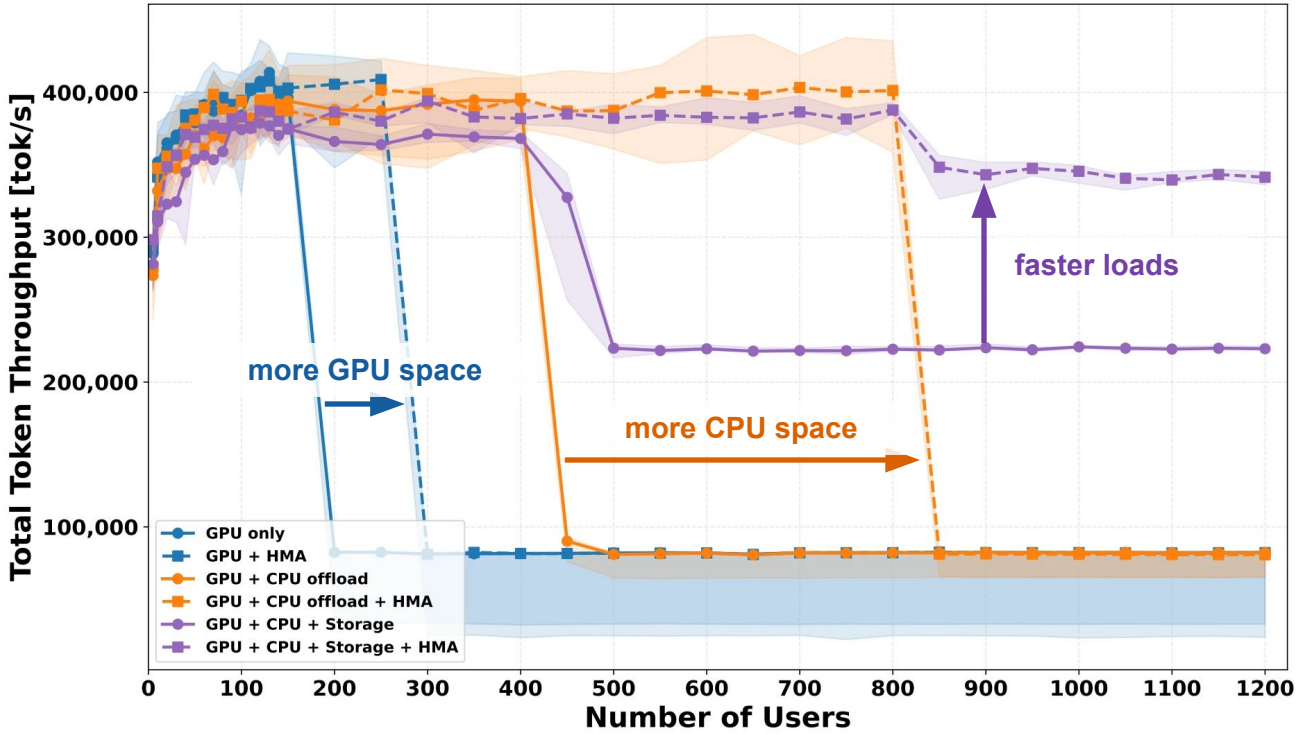
HMA-aware connector
Offloads and loads each cache group on its own

Source: llm-d blog, Serving Hybrid Models at Scale in llm-d (June 2026): llm-d.ai/blog/serving-hybrid-models-at-scale-in-llm-d

HYBRID MODELS

HMA-aware offload: 1.9x faster loads, up to 2x more users





Loads 1.8x to 1.9x faster

Sliding-window layers read 128 tokens of KV, not 128K

10 concurrent 128K-token prompts; gpt-oss-20b TP=2, gpt-oss-120b TP=4

More users on IBM Storage Scale

CPU tier ~400 to ~800 users; storage plateau ~220K to ~340K tok/s

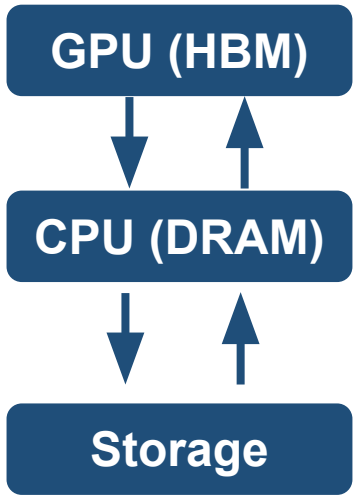
gpt-oss-120b TP=4; each user resends its own 16K-token prompt

H100, IBM Storage Scale. Source: llm-d blog, Serving Hybrid Models at Scale in llm-d (June 2026): llm-d.ai/blog/serving-hybrid-models-at-scale-in-llm-d

GPUDIRECT STORAGE

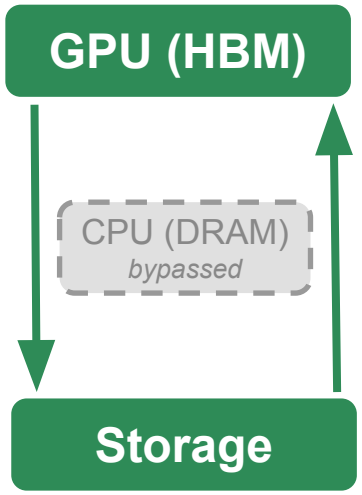
GDS skips the CPU hop: 35% lower load latency at TP8

Multi-tier (CPU staging)



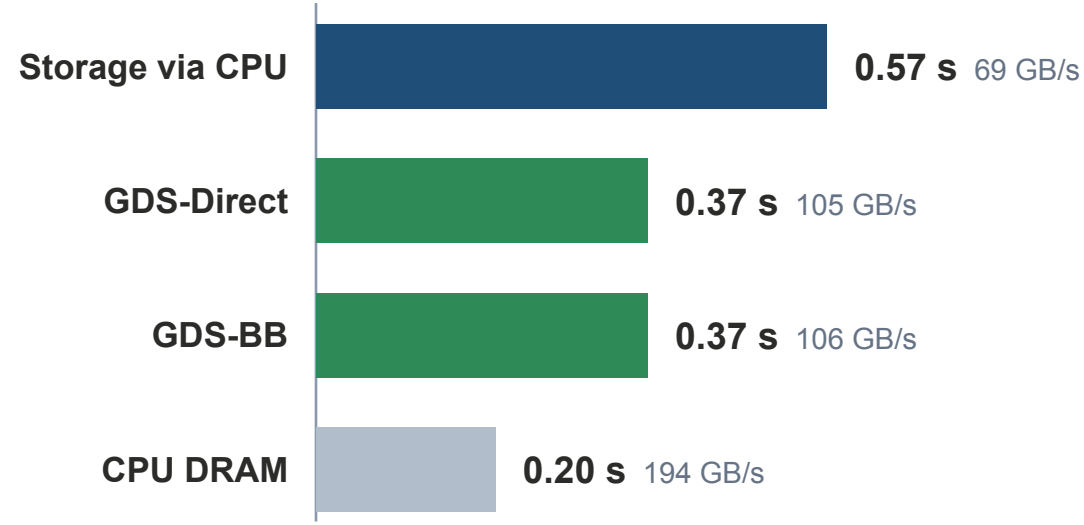
Two hops: data staged through CPU

GDS (direct path)



Direct GPU to storage transfer

KV load latency at TP8 on IBM Storage Scale



35% faster at TP8

0.37 s and ~105 GB/s with GDS vs 0.57 s and 68 GB/s through CPU staging

No gain at TP4

All storage paths land near 0.55 s

Keep CPU as the first tier

CPU DRAM is fastest (0.20 s); GDS speeds up the storage tier behind it

Llama-3.1-70B, 128K-token prompts, all runs on **IBM Storage Scale**. Storage via CPU = file system read staged through CPU DRAM; GDS-Direct and GDS-BB = GPUDirect Storage paths; CPU DRAM = CPU tier.

Four-Tier llm-d KV Cache Hierarchy

THE 4-TIER ARCHITECTURE • FROM LOCAL HBM TO FLEET STORAGE & P2P

Tier 1: GPU HBM Hot Compute

Medium: On-package HBM3/HBM3e (80–192 GB/GPU, >3.3 TB/s).
Function: Holds active attention tensors for executing tokens.
Scope: Single GPU / TP group; sub-microsecond access.

Tier 2: CPU & SSD Warm Node

Medium: Pinned Host DRAM (256GB–2TB) & local NVMe via PCIe Gen5.
Function: Absorbs HBM evictions via vLLM native Offloading Connector.
Scope: Single node; delivers up to 28x faster TTFT than recompute.

Tier 3: Lustre FS Shared Fleet

Medium: Google Cloud Managed Lustre / POSIX RWX parallel FS.
Function: Persists terabytes to petabytes via vLLM native offloading connector.
Scope: Cluster-wide across GKE pods; survives pod restarts.

Tier 4: P2P RDMA Cross-Pod

Medium: Peer pod CPU DRAM accessed over NIXL / Mooncake RDMA.
Function: Transfers warm KV blocks across pods at 100–400 Gbps.
Scope: Fleet memory pooling with zero impact on source GPU.

Google Cloud Managed Lustre + llm-d

Decentralized, persistent KV cache storage purpose-built for high-bandwidth AI workloads on Google Kubernetes Engine.

- **Parallel Striped Throughput**
  Stripes large cross-layer KV blocks across distributed Object Storage Targets (OSTs) for massive parallel read bandwidth.
- **Zero Cold-Start Autoscaling**
  Newly scheduled GKE GPU pods immediately mount the RWX Lustre volume and hydrate pre-computed KV blocks.
- **Hybrid CPU + Lustre Caching**
  Combines pinned host DRAM as an L2 buffer with Managed Lustre as an L3 fleet store for optimal latency and capacity.
- **Turnkey GKE CSI Integration**
  Deploys directly via the Google Cloud Managed Lustre CSI driver and llm-d tiered-prefix-cache Helm/Kustomize guides.



Verified TCO, Latency & Throughput Gains

>50%

Inference TCO Savings with Managed Lustre
Google Cloud benchmark on 6-node A3 Mega (Llama-3.3-70B, 50k+ token prompts, 95% cache hit rate).

>40%

40% Faster TTFT (Hybrid Tiering): Combining Host CPU RAM + Managed Lustre cuts TTFT by 40% vs. CPU-only offload on 50k-token contexts.

PUBLISHED GOOGLE CLOUD & LLM-D METRICS

Nearly 60% Fewer GPU-Hours: Offloading KV cache to Google Cloud Managed Lustre eliminates redundant prefill across 48 H100 GPUs.

Faster TTFT with Precise Prefix Routing
llm-d Endpoint Picker KVEvents routing vs. naive round-robin ingress under shared-prefix load.

30% Lower End-to-End Latency: Sustained end-to-end latency reduction across multi-turn agentic workflows on GKE.

2x+ Throughput & 28x CPU Speedup: Doubles cluster QPS with EPP routing and delivers 28x single-node TTFT speedup via native CPU offload.

GETTING STARTED • GITHUB.COM/LLM-D/LLM-D/TREE/MAIN/GUIDES

Upstream llm-d guides simplify tiered prefix cache deployment on Kubernetes clusters

STEP 01

Provision RWX PVC

Create a Google Cloud Managed Lustre ReadWriteMany PVC (or CephFS/FSx) in your Kubernetes namespace.

STEP 02

Select Connector

Set CONNECTOR=native (or lmcache) and VARIANT=cpu or VARIANT=fs from guides/tiered-prefix-cache.

STEP 03

Configure CPU Hub

Allocate pinned host DRAM buffer via --CpuOffloadingGb in the vLLM container args for async DMA.

STEP 04

Mount Shared FS

Mount the Managed Lustre/IBM Storage PVC at /mnt/kv-cache so vllm native offloading connector shares blocks across pods.

STEP 05

Enable KVEvents

Turn on vLLM ZMQ KVEvents publishing so every pod streams block store/evict updates in real time.

STEP 06

Deploy EPP Router

Apply guides/precise-prefix-cache-routing to launch the Gateway API Endpoint Picker with global indexing.

STEP 07

Tune Weights

Configure prefix-cache-scorer alongside queue-scorer and kv-cache-utilization-scorer in EPP config.

STEP 08

Verify Hits

Monitor Prometheus prefix_cache_hit_rate, TTFT p90, and Lustre throughput via llm-d dashboards.

SUMMARY & COMMUNITY • CLOUD NATIVE THEATER AT KUBECON NA 2026

Explore the llm-d 1.0 guides and transform your cluster into a unified cache

Unify All 4 Tiers of Cluster Memory: Extend GPU HBM with Host CPU DRAM, Google Cloud Managed Lustre (RWX), IBM Storage Scale, and P2P RDMA so no prefill compute is ever wasted.

Pair vLLM Native Offloading with EPP Routing: Co-designing real-time ZMQ KVEvents routing with tiered offloading delivers up to 57x faster TTFT, 40% Lustre TTFT gains, and >50% TCO savings.

Try the Open-Source Guides Today: Visit github.com/llm-d/llm-d/tree/main/guides for turnkey Kubernetes manifests, Helm charts, and benchmark recipes.

"The largest KV cache tier already exists in your cluster—llm-d 1.0 and vLLM turn isolated GPUs into a fleet-wide memory hierarchy."

Thank you! Questions?

Cloud Native AI + Inference Day | Salt Lake City, Nov 9, 2026

Try llm-d
github.com/llm-d/llm-d

Kfir Toledo
IBM Research

Nishtha Jain
Google Cloud

Google Cloud | IBM | llm-d