Agentic AI Serving System with Computational Shared Memory Pool

[FTI Workshop]

Jongryool Kim

Senior Director

AI System Infra., Memory System Research

SK hynix, Inc.

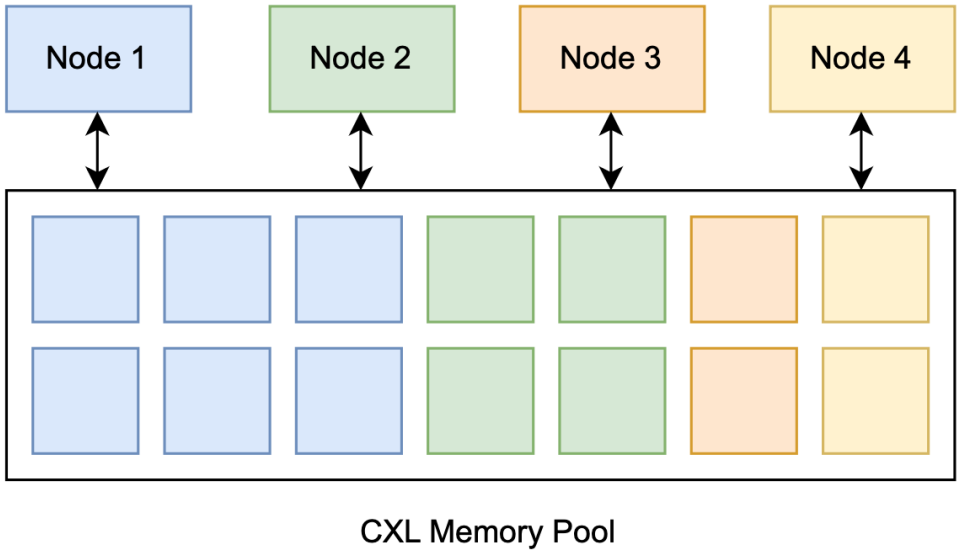


This Talk

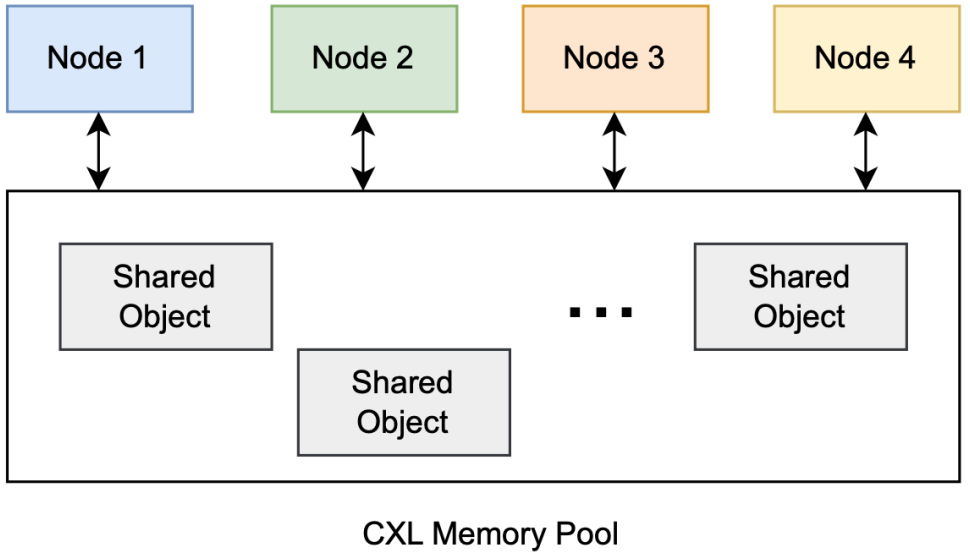
- KV cache sharing to improve GPU utilization
  - KV cache selection offloading to a computational memory pool
- Token sharing to reduce CPU load
  - Tokenization offloading to a computational memory/memory pool to mitigate CPU resource constraints

Memory Pool

- Memory pooling: Mitigate memory stranding by allocating CXL memory based on memory usage for each node
- Memory sharing: Remove data transfer overhead and data duplication by sharing data objects between nodes



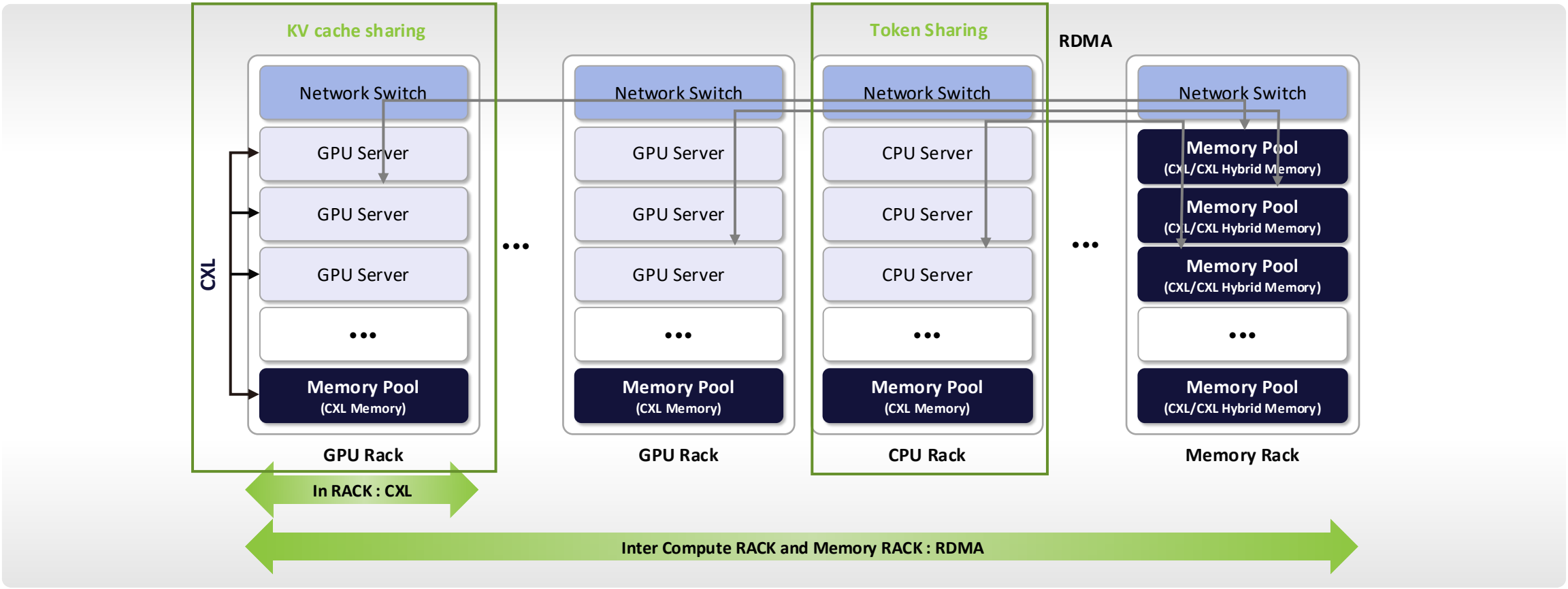
Memory Pooling



Memory Sharing

Multi-rack Disaggregated Memory System

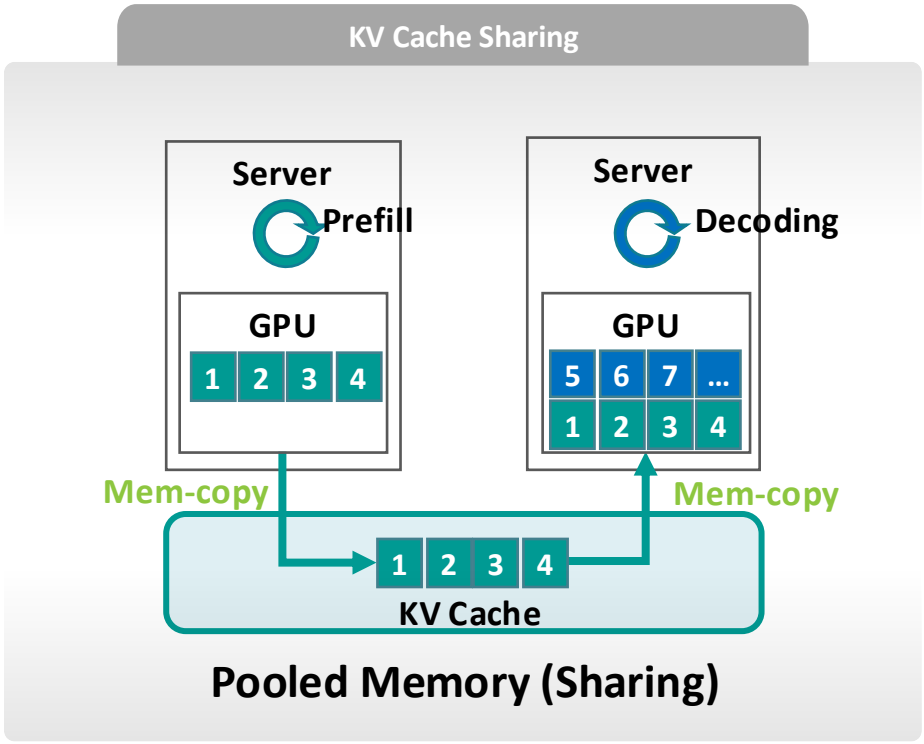
- Memory-centric AI Rack System (MARS): A disaggregated rack-scale memory system that coordinates compute and memory as a shared infrastructure

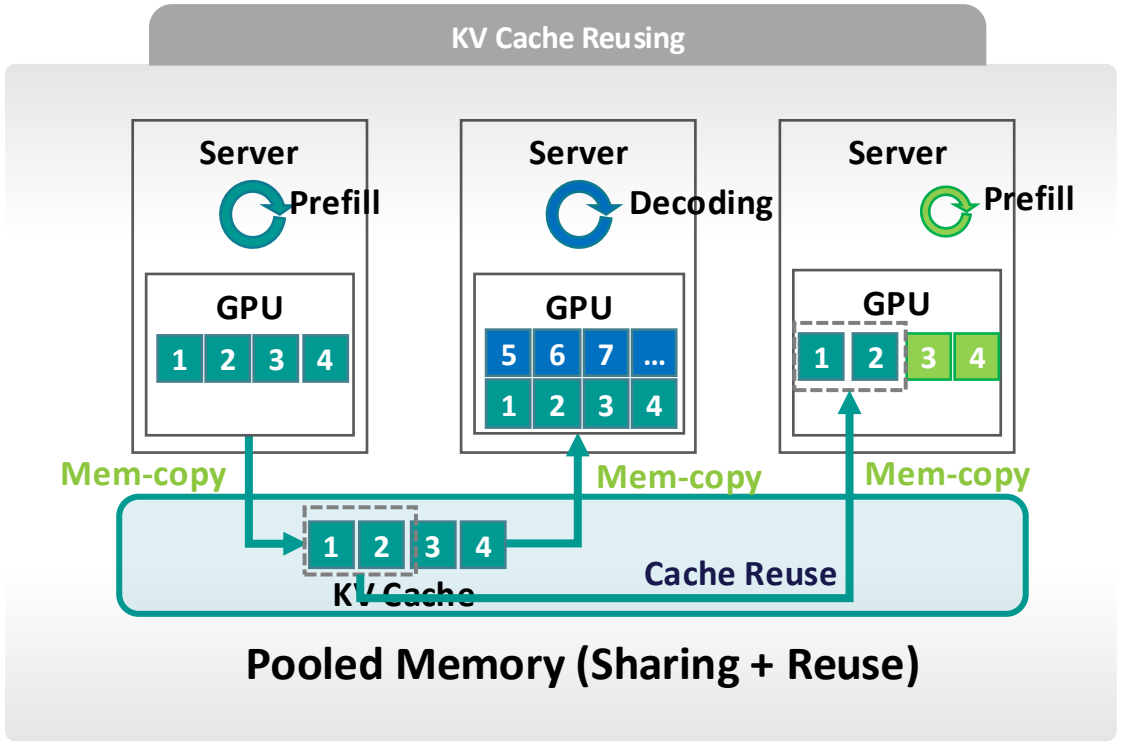


KV cache sharing for GPU

KV Cache Sharing

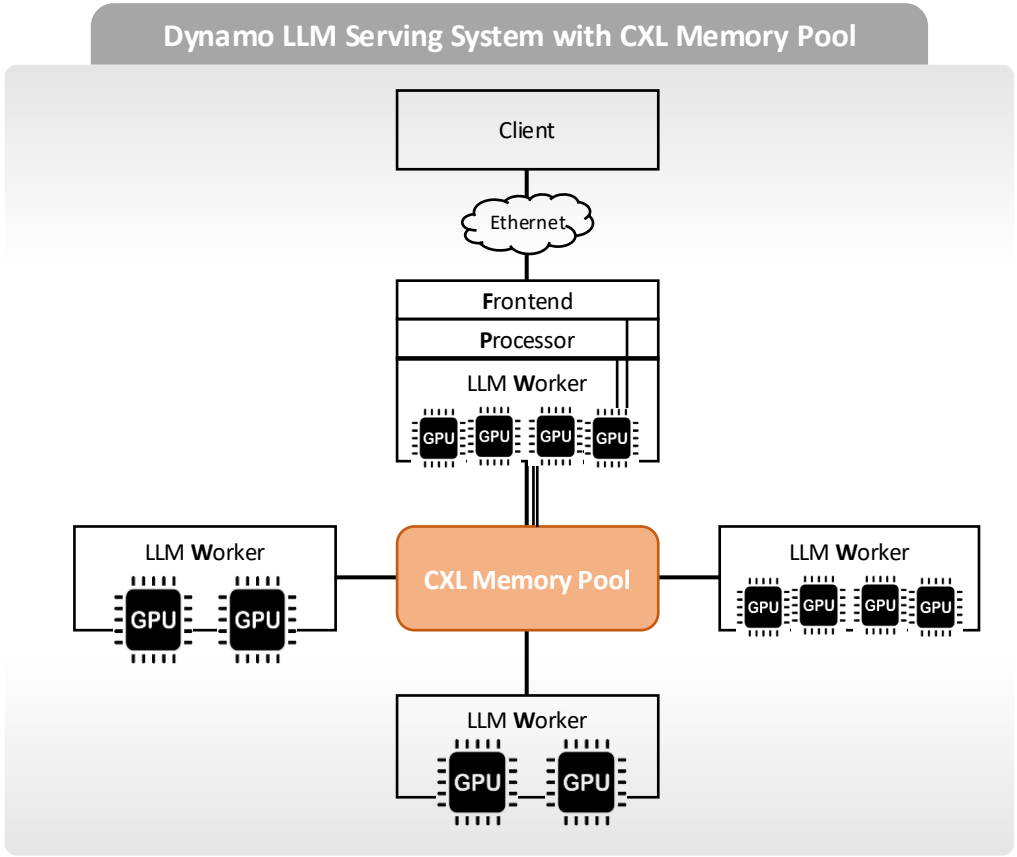
- KV cache transfer (sharing) and reuse over pooled memory improved performance in LLM Inference
  - Direct fast reuse of KV cache stored in pooled memory during the sharing process whenever needed.
  - Bypasses secondary save-and-load operations, drastically cutting down data path latency





Dynamo LLM Serving with CXL Pooled Memory (@ FMS 2025, OCP 2025, SC 2025)

- Developed and Demonstrated a Pooled Memory-Based Distributed LLM Serving System

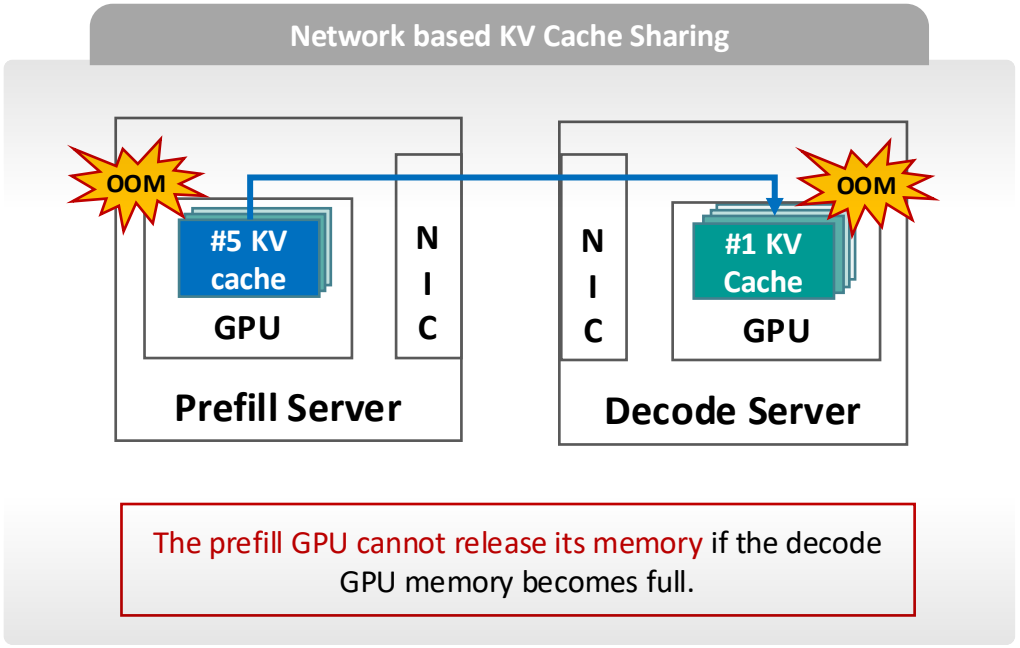


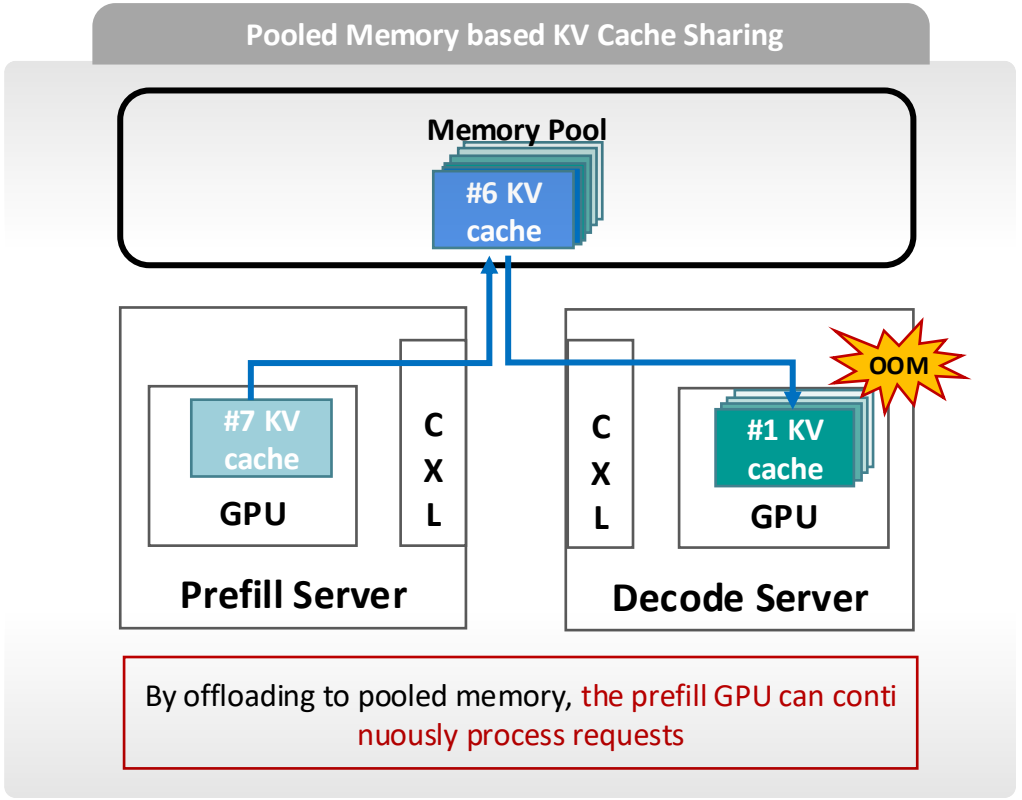




Fast HBM Reclamation

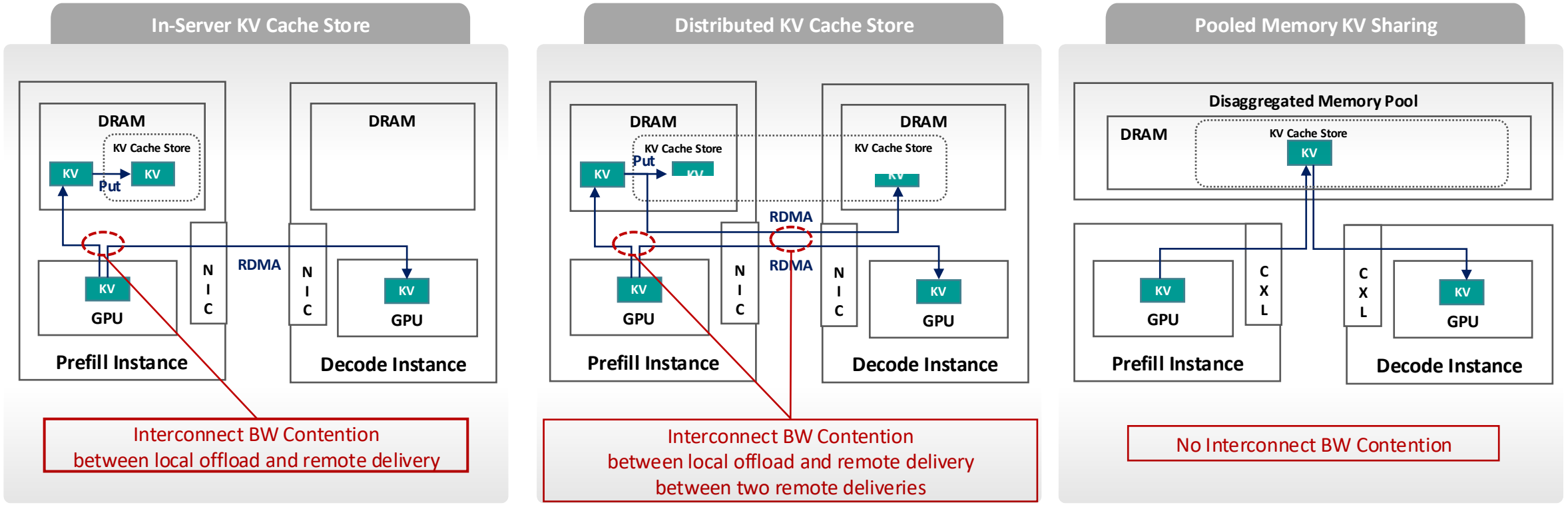
- Release GPU HBM immediately by offloading KV cache to pooled memory
  - Accelerates GPU HBM reclamation by offloading data to pooled memory immediately.
  - Maximizing Prefill hardware utilization.





KV Cache Store/Reuse without BW Contention

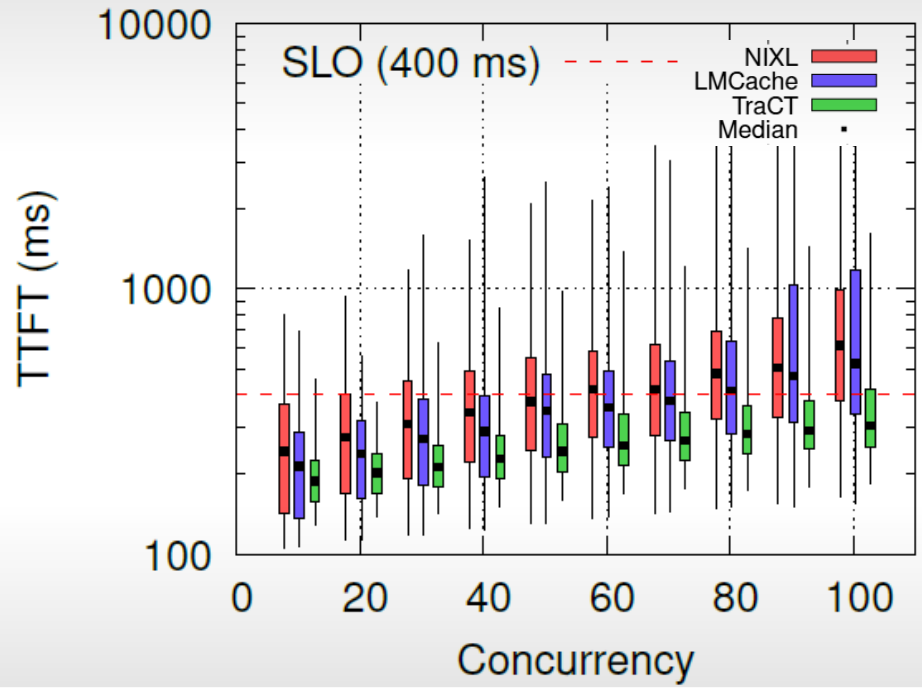
- Avoid bandwidth (BW) contention across HBM, PCIe, and Network paths for KV cache Store/Reuse
  - Avoids dual DRAM/NIC data paths, resolving the BW contention seen in LM-Cache/Mooncake-like setups



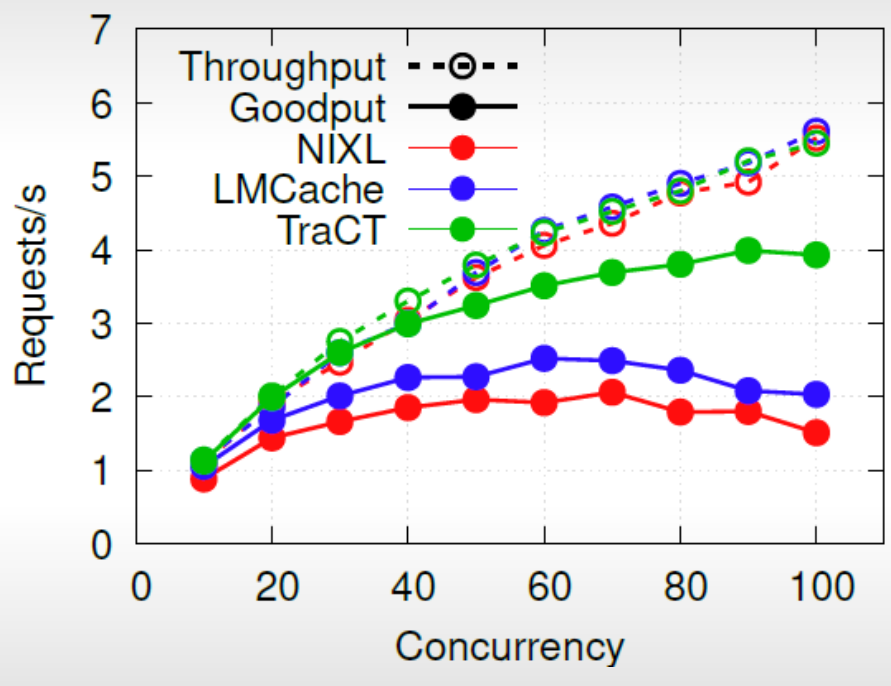
LLM Serving Performance

- LLM Serving using CXL Pooled Memory can improve the LLM serving performance
  - 1.67× TTFT improvement over LM-Cache, 1.9x higher peak good put over NIXL (ALL recomputation)

Memory Pool for a Compute Rack



Memory Pool for a Compute Rack

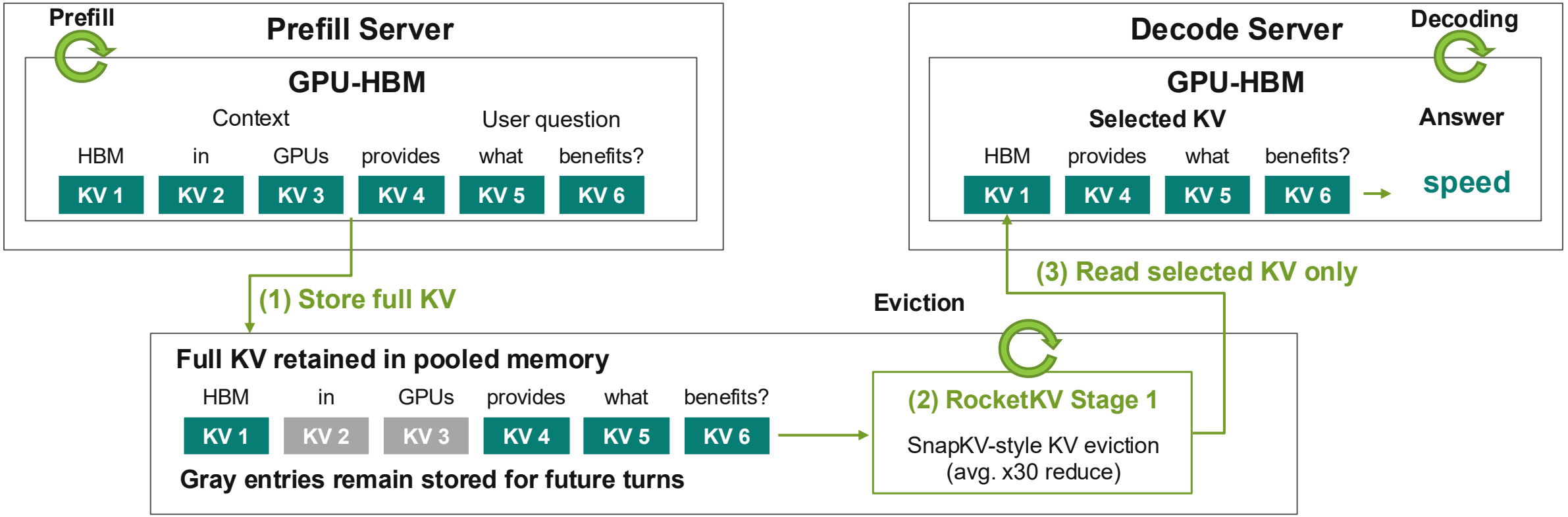


KV cache selection

KV Cache Selection Offloading to Pooled Memory

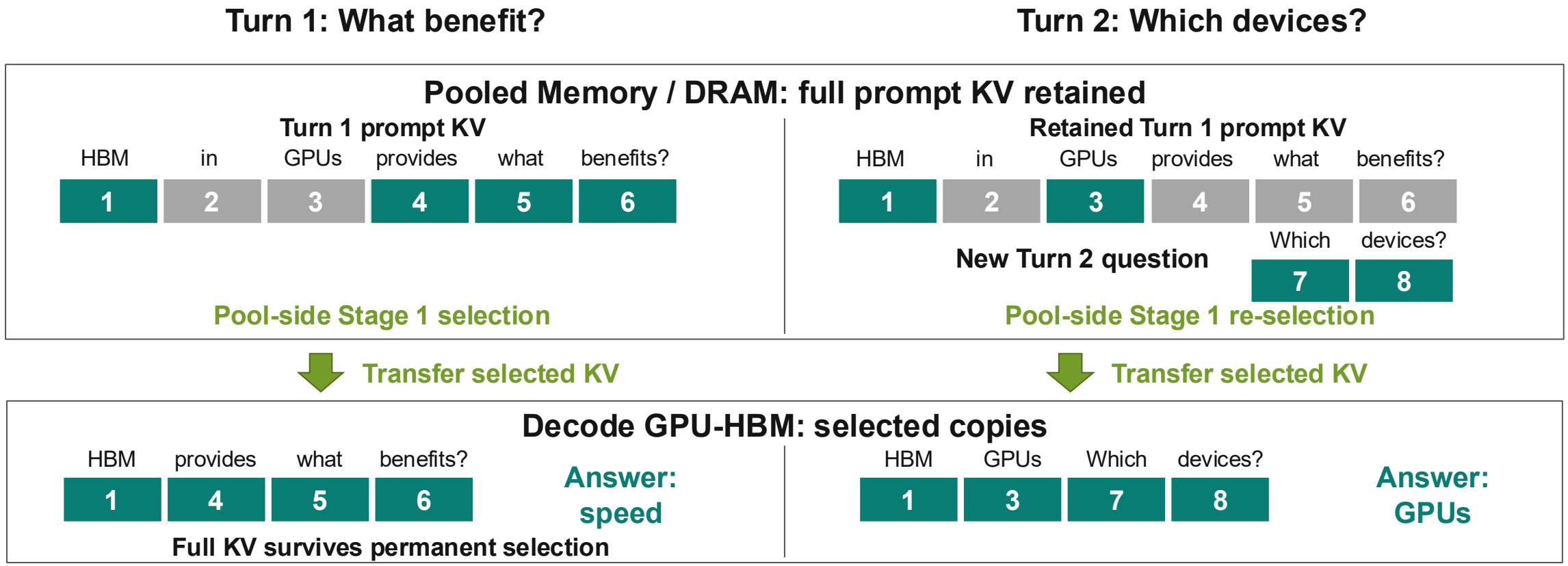
- Offload KV cache selection for sparse attention to pooled memory and load only selected KV cache.

  → can improve the decoding latency



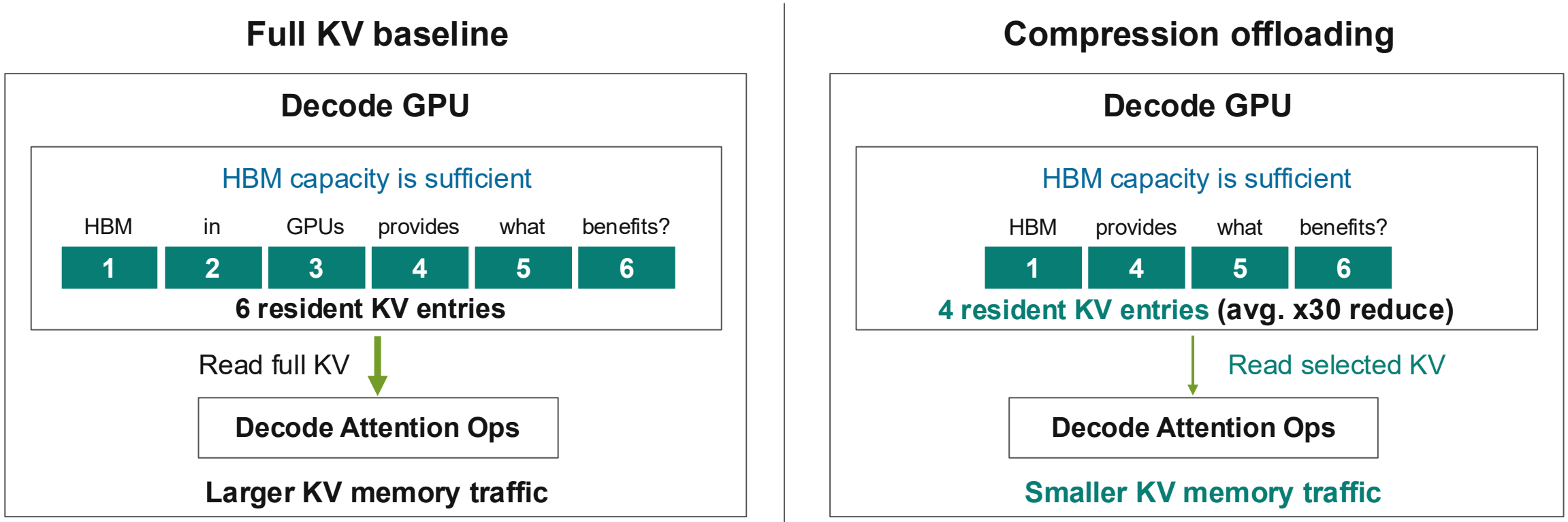
Why do we store the entire KV cache?

- Evicting KV entries can lead to context loss unless they are recomputed from the conversation history.
- To preserve full context without re-computation, the KV cache for the entire conversation must be retained.



Selective KV Cache Transfer Reduces Latency

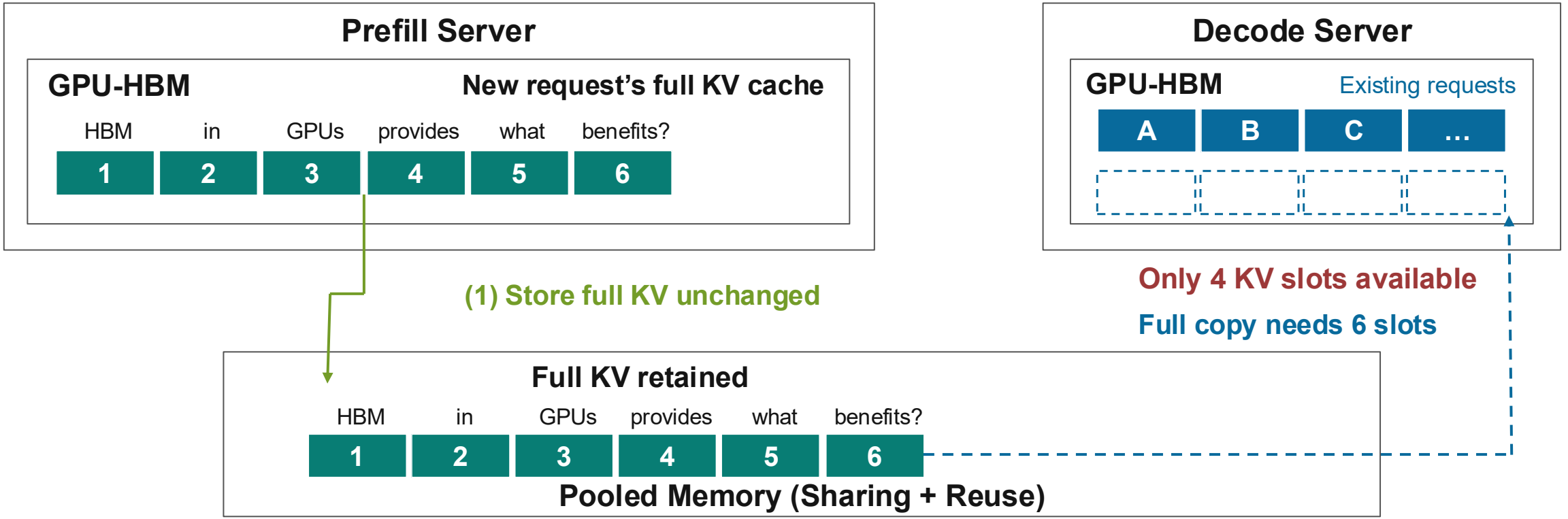
- Transfer only selected KV cache entries to the decoding GPU.
  - Reduce the amount of data transferred and shorten KV cache transfer time.
  - Enable decoding to start sooner, lowering Time to First Token (TTFT).



Smaller KV memory traffic without capacity pressure

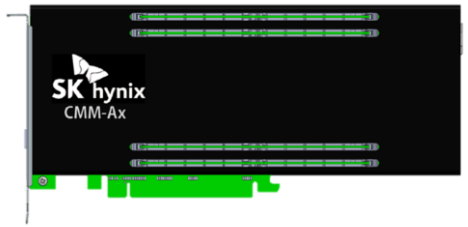
Selective KV Caching Reduces Queuing Delay

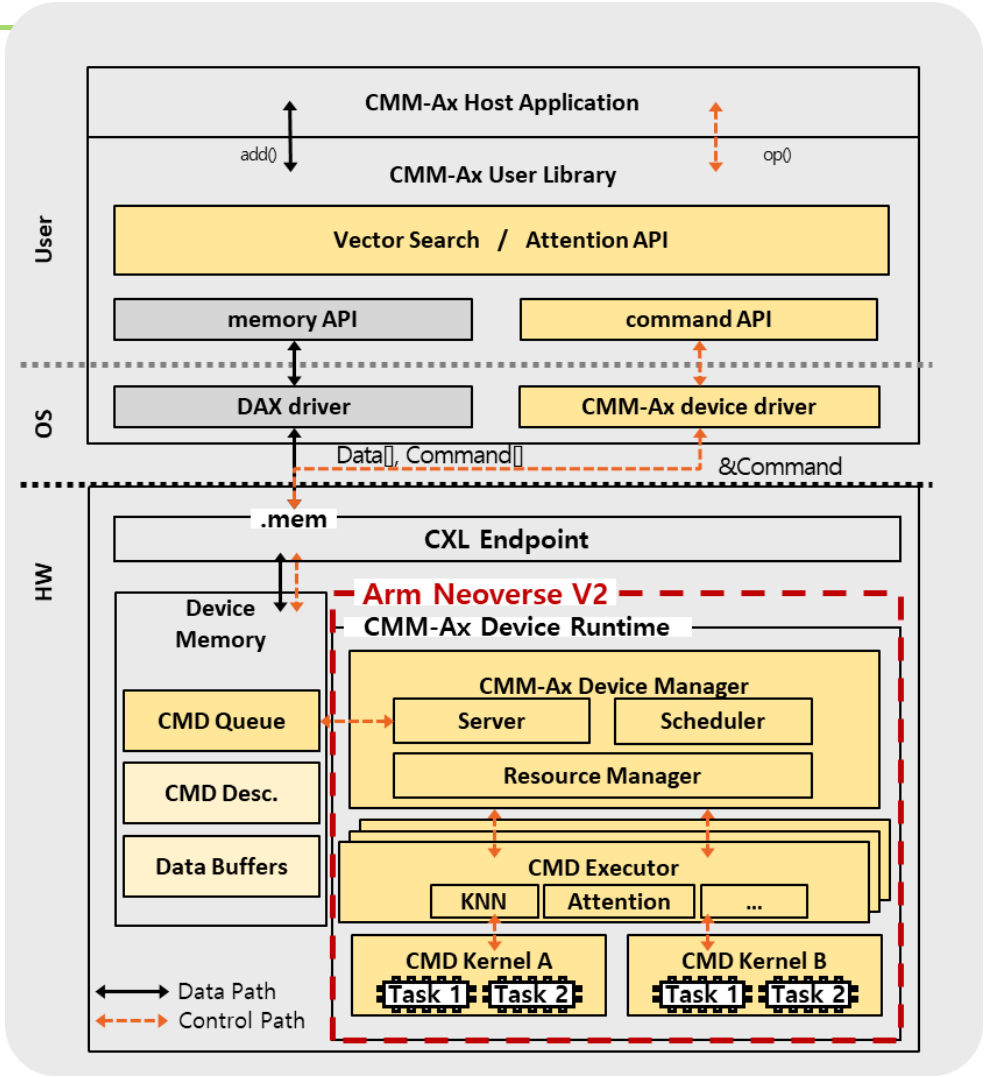
- Transferring only selected KV cache entries reduces the HBM footprint per request on the decoding GPU.
  - This alleviates HBM capacity constraints that can otherwise block requests from joining the decoding batch.
  - Requests can enter the decoding batch sooner, reducing queuing delay → low e2e latency



CMM-Ax 3.0

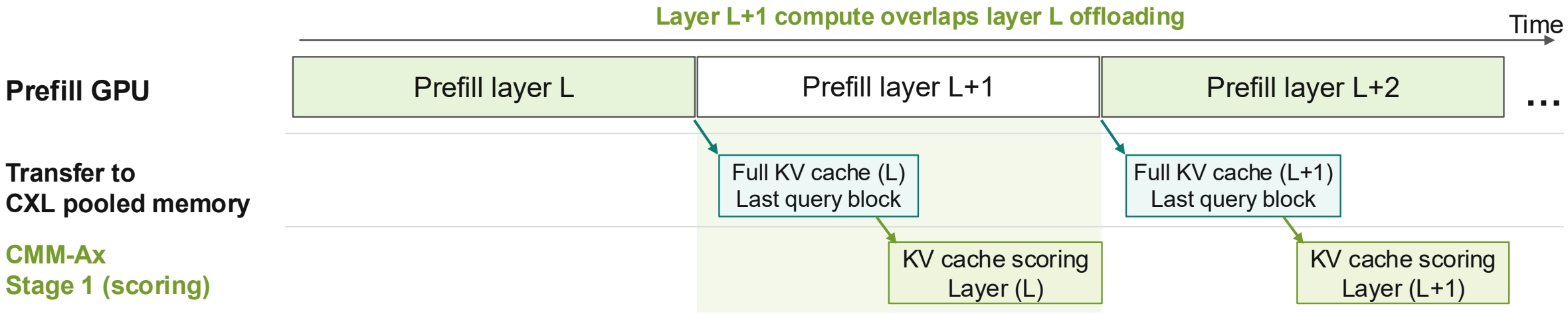
- General Cores only
  - High Performance & Highly Flexible PNM
- CXL.mem Command Signal
  - Fast Offloading
  - Finer-grained Command Offloading
- Full Stack SW environment
  - Attention API/Vector Search Library for AI
  - Programming Model
  - Device Runtime

CMM-Ax		
Form Factor	FH3/4L AIC Full Height ¾ Length Add-in-Card	
Interconnect	CXL 2.0 / Device type 3 / PCIe 5.0 x16	
Memory*	DDR5-6400 4-ch / 512GB / 200GB/s	
Controller	16 ARM Neoverse V2 cores at 3.0GHz	

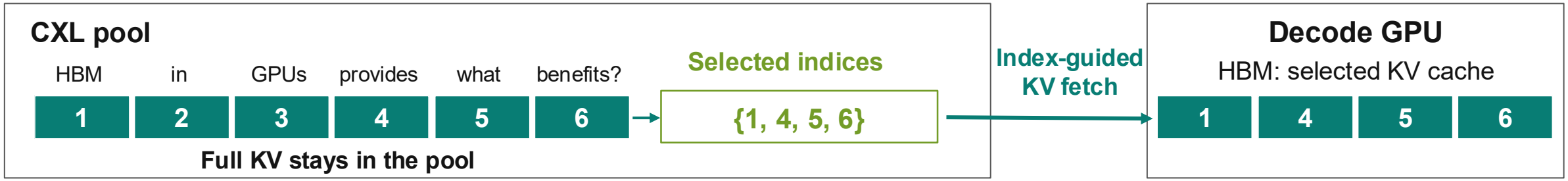


Pipeline Optimization

- Overlap the next layer's prefill computation with KV cache transfer and pool-side compression.
  - CMM-Ax runs RocketKV Stage 1 and selects KV indices for each layer.



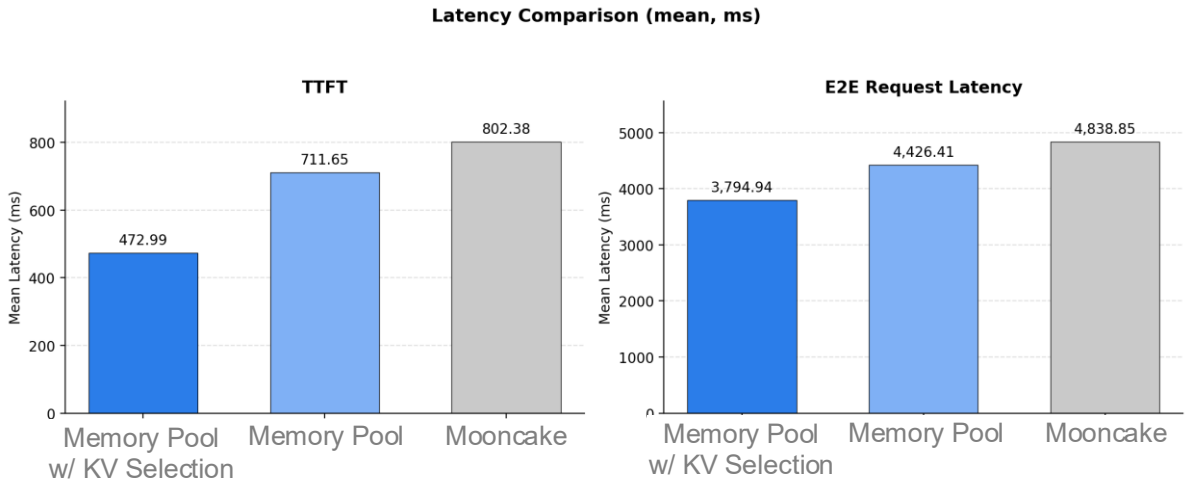
Later: Decode fetches KV with the selected indices



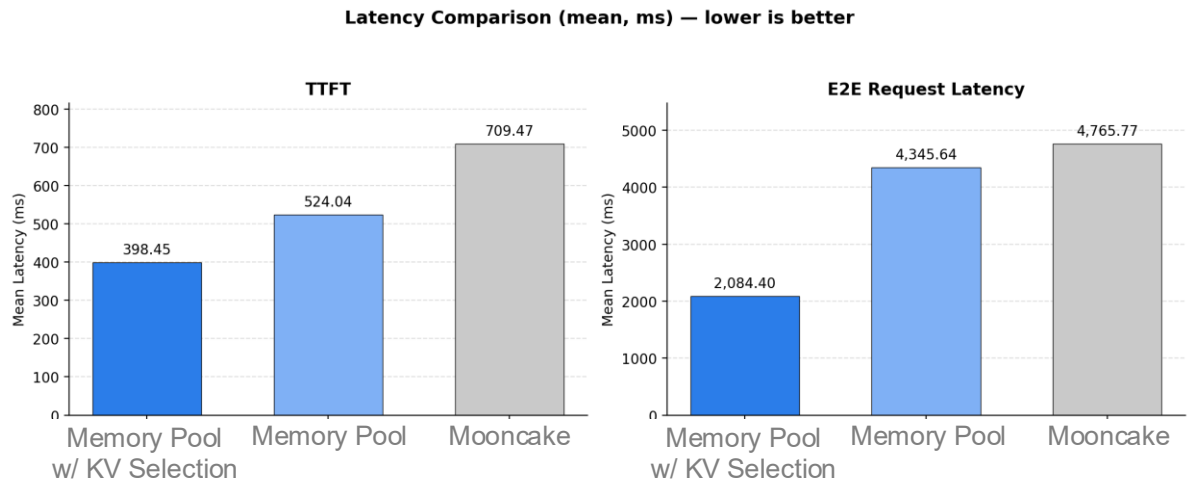
Results

- Overlap the next layer's prefill computation with KV cache transfer and pool-side compression. (with Structera CXL)
  - CMM-Ax runs RocketKV Stage 1 and selects KV indices for each layer.
  - 1.4x TTFT improvement and 2.29x E2E Latency improvement over Mooncake (Arcee trace)

Opus-4.5 trace runtime comparison



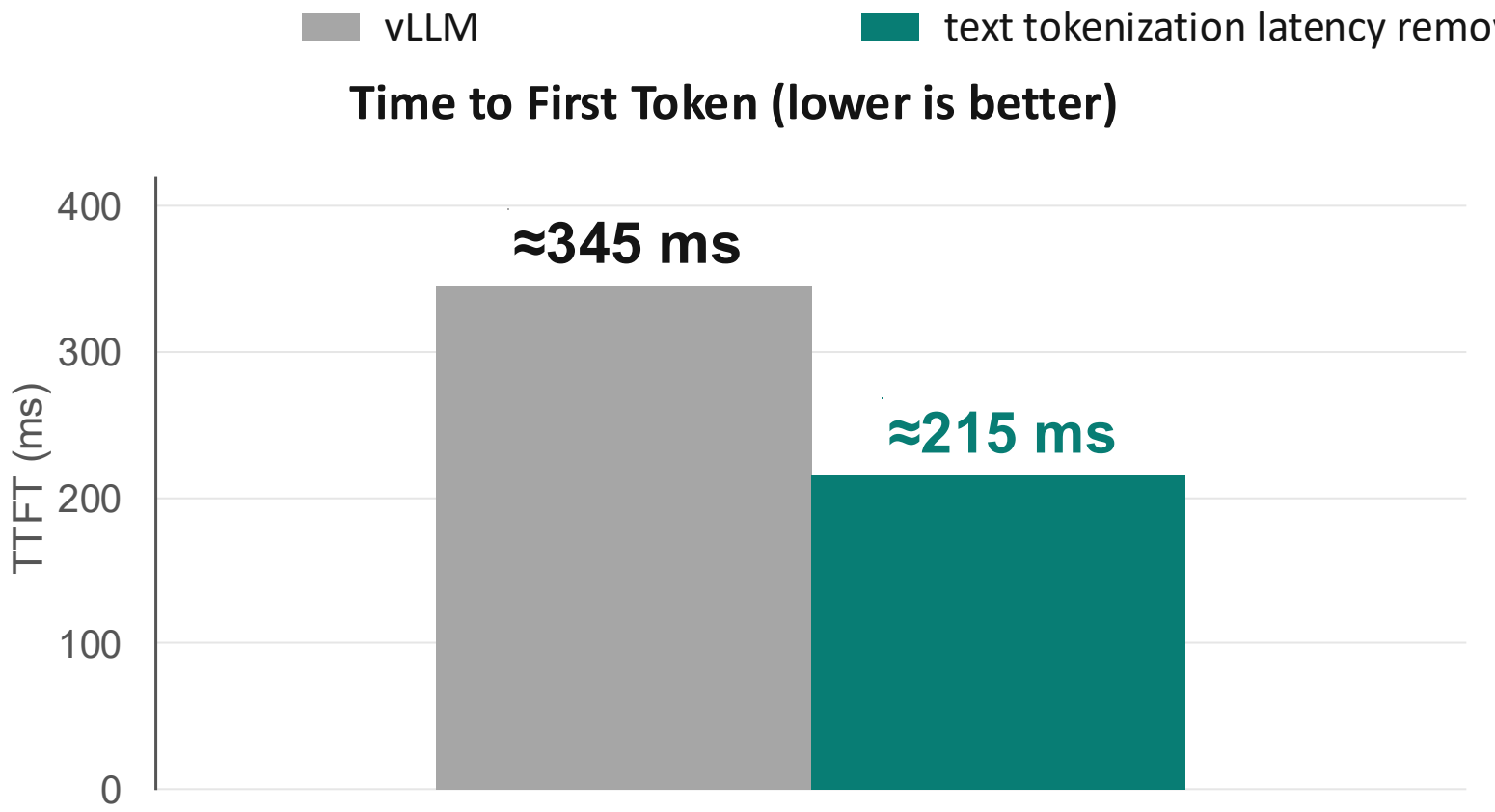
ARCEE trace runtime comparison



Token sharing & Offloading for CPU

Text Tokenization Cost

- 25.6K input tokens, prefix KV cache reuse 96%*.



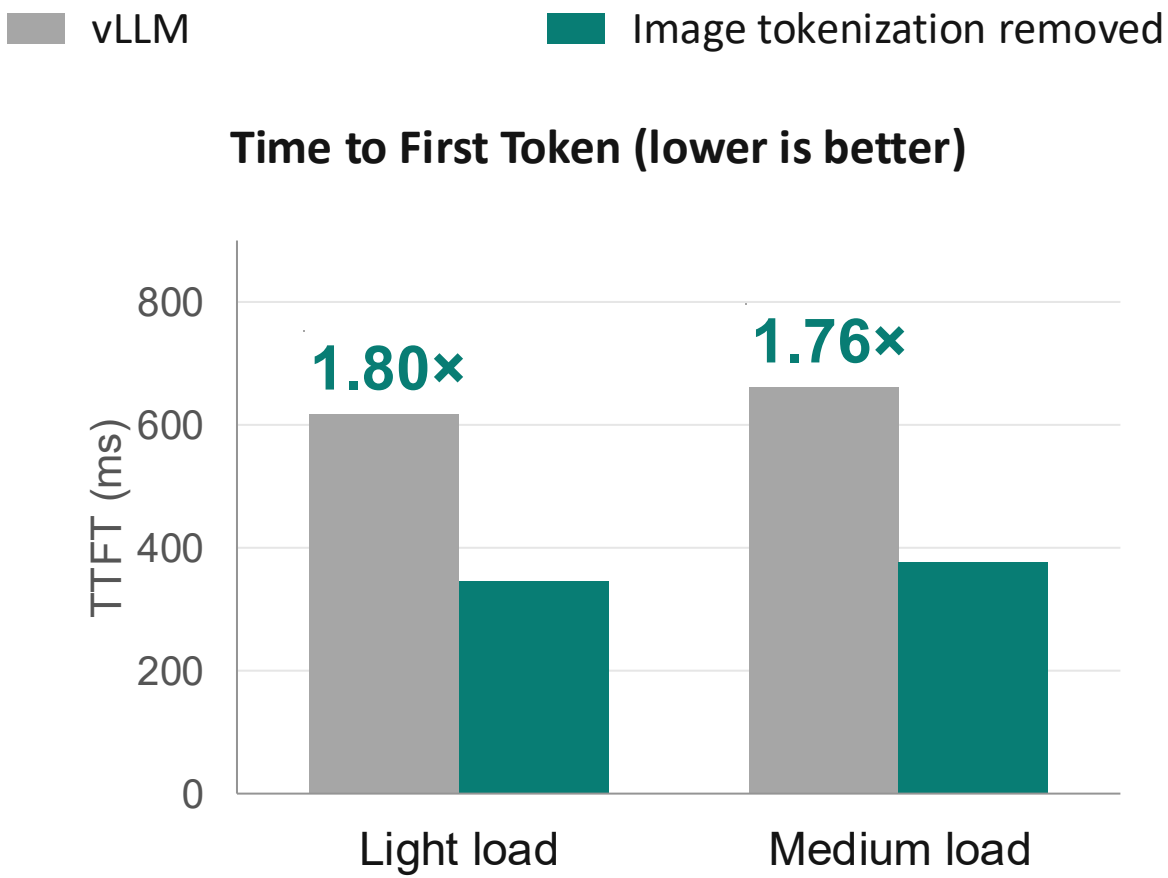
≈ 38%

lower TTFT

* Coding-agent workloads have reported prefix KV cache hit rate of ≈ 96%.

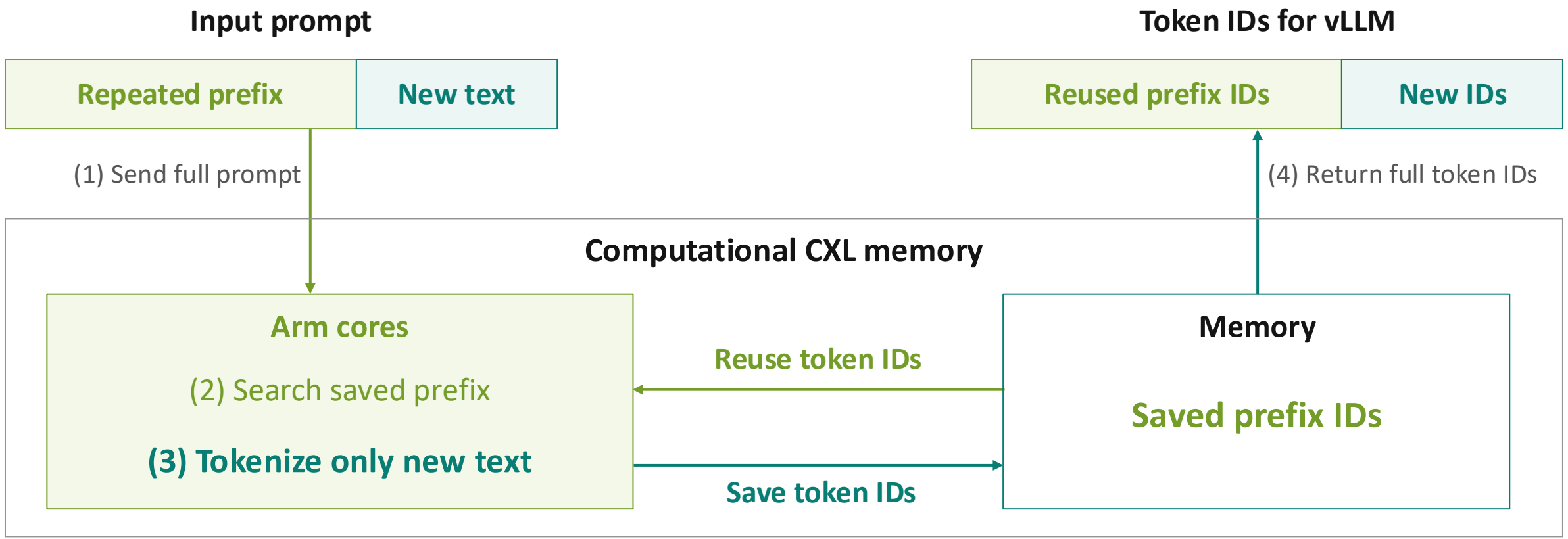
Image Tokenization Cost

- Removing image tokenization improves serving throughput and latency.



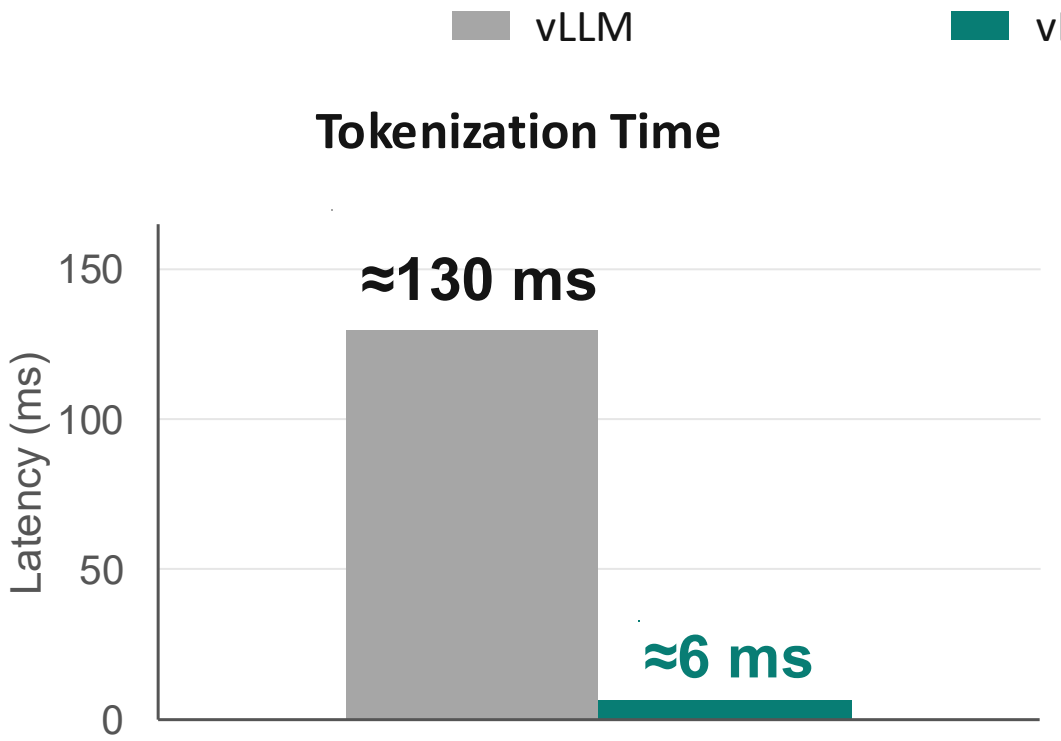
Token Sharing in Computational CXL

- Computational CXL reuses prefix token IDs and tokenizes only new text.

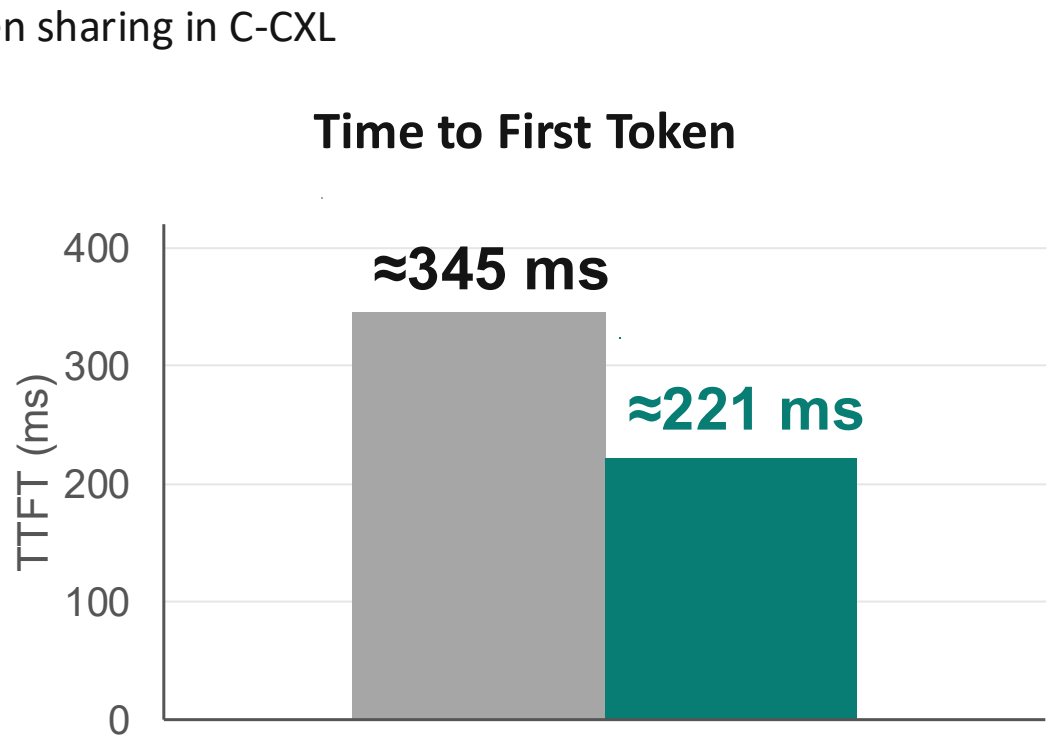


2) Token Sharing in Computational CXL

- 25.6K input tokens, prefix KV cache reuse 96%*.



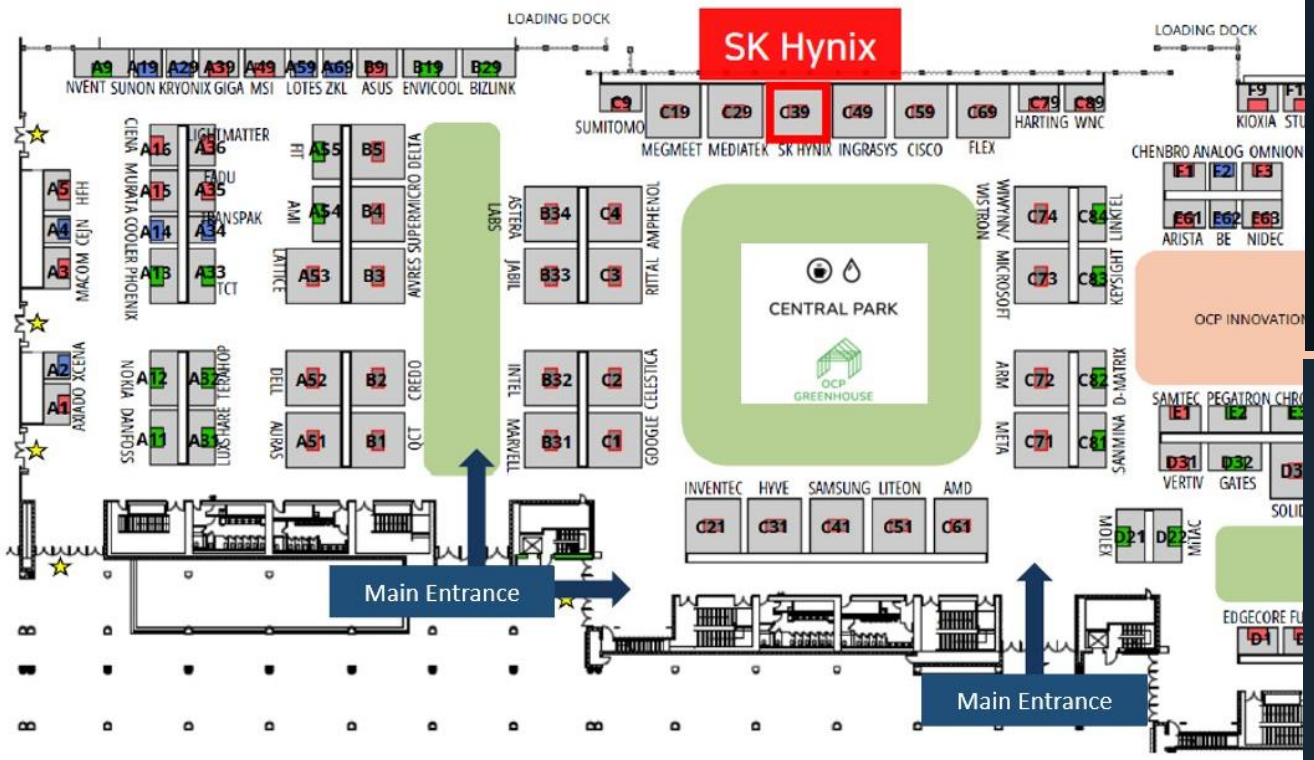
≈ 95% lower

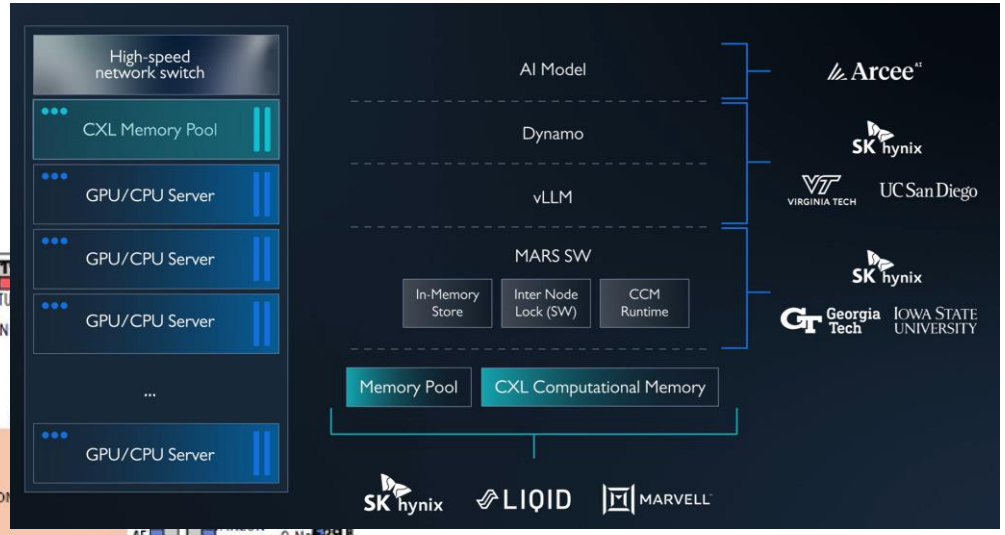


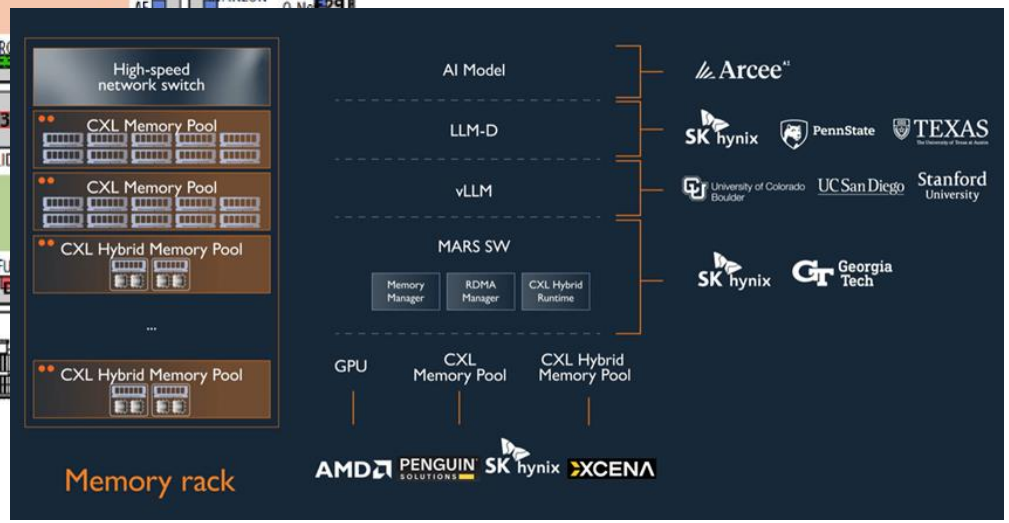
≈ 36% lower

Learn More about Memory Pool

- SK hynix booth : C39








Thank
You!
OCP
GLOBAL
SUMMIT
October 12-15, 2026
San Jose, California