SERVER:
COMPOSABLE MEMORY
SYSTEMS (CMS)

Tiered Memory for LLM Inference — A Modeling Framework for KV-Cache Offload

Khayam Anjam *[Staff Engineer – Micron Technology]*

OCP BRONZE™ Member

LLM Inference Has Created a New Memory Wall

HBM bandwidth is no longer the only bottleneck—capacity increasingly determines scalability

Growth in:

- Context lengths
- Concurrent users
- Multi-turn conversations
- Agent workloads



Result:

- KV-cache dominates memory footprint
- HBM capacity becomes limiting
- Memory offload is becoming mandatory

The challenge is no longer whether KV should be offloaded, but where it should be placed.

Why Existing Analysis Is Difficult

- Evaluating KV offload today requires expensive experimentation

Approach	Limitation
Hardware benchmarking	Slow and expensive
Detailed simulation	Long runtimes
Roofline analysis	Ignores capacity
Rule-of-thumb sizing	Misses workload dynamics

A practical architecture-level framework is missing!



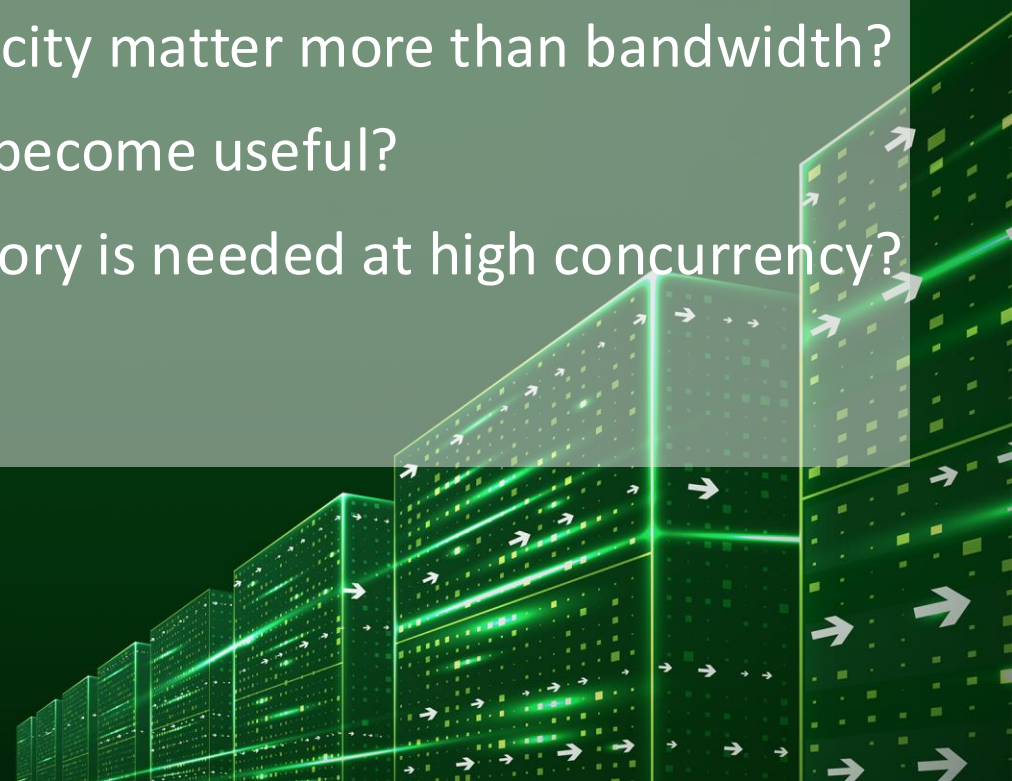
What Questions Should a Framework Answer?

- Architects need answers before hardware is built.

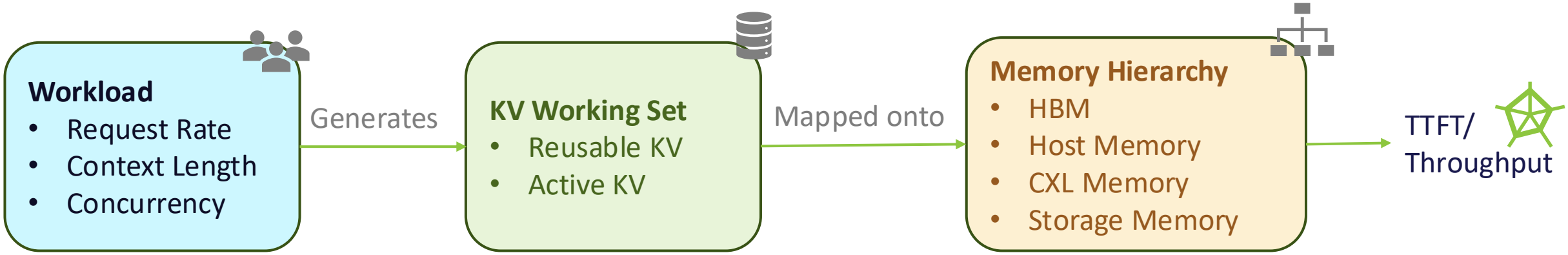
Examples:

- How much secondary-tier memory is needed?
- When does offload improve TTFT?
- When does capacity matter more than bandwidth?
- When does CXL become useful?
- How much memory is needed at high concurrency?

Our goal is a framework that answers these questions analytically.



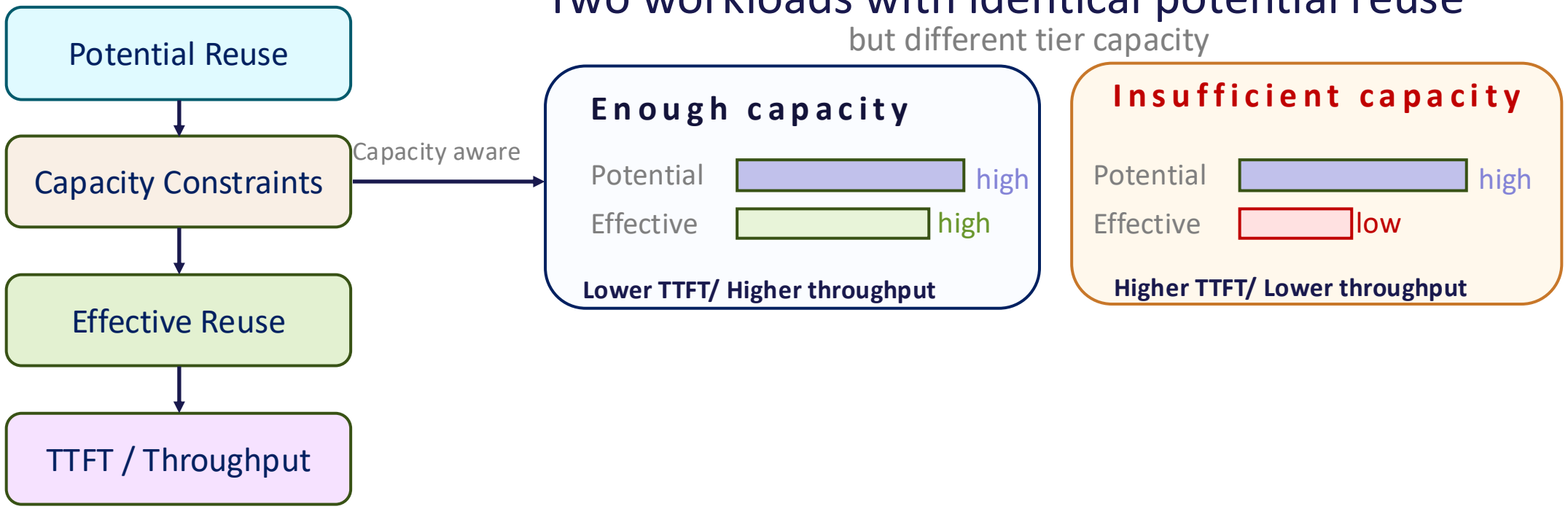
Model inference as workload × working set × memory hierarchy



Performance emerges from the interaction of all three.

Reuse Alone Is Not Enough

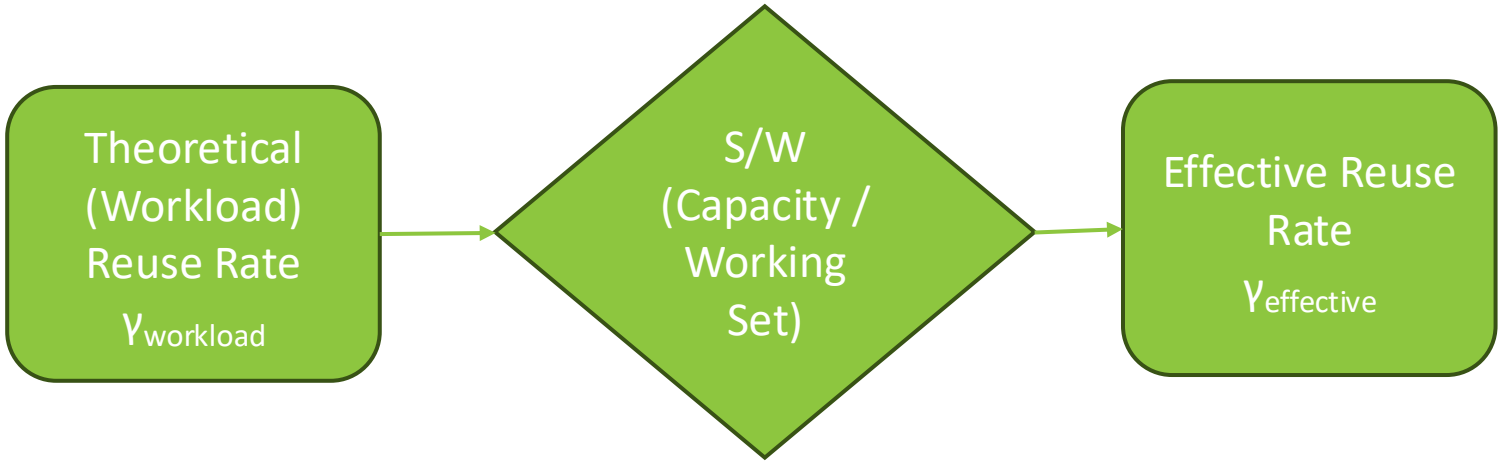
Potential reuse is not the same as effective reuse



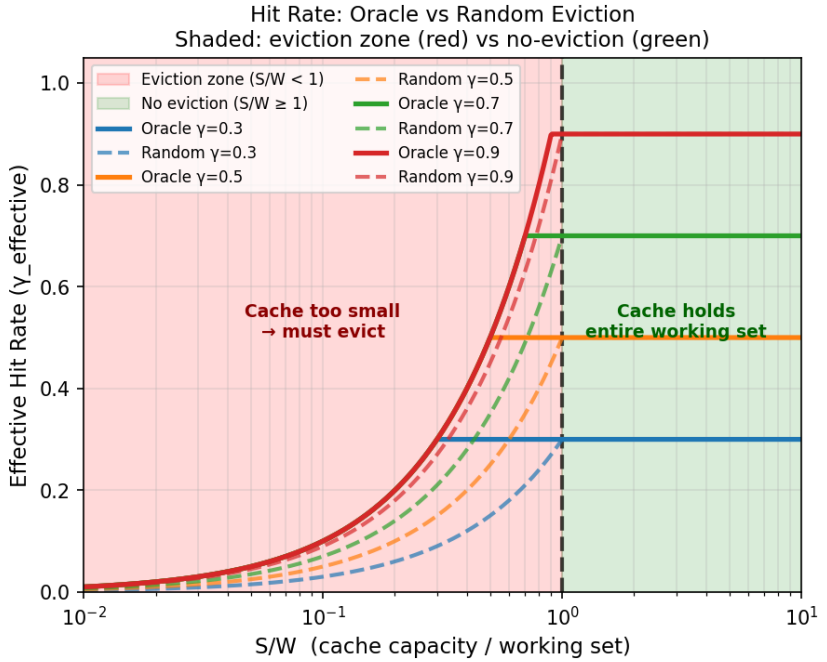
Takeaway: Reuse must be capacity-aware.

Effective reuse = workload reuse gated by capacity

Policy-agnostic framework: no dependence on proprietary cache implementations



Bound	Formula
Random (conservative)	$\gamma_{effective} = \gamma_{workload} \times \min(1, S/W)$
"Oracle" – best case (optimistic)	$\gamma_{effective} = \min(\gamma_{workload}, S/W)$



Capacity determines the range of achievable reuse; cache policy only determines where within that range a system operates.

KV Cache Is Not One Working Set

KV cache naturally separates into two populations.



System Prompt Tier (Cross-Session)

- Shared instructions reused across sessions
- Moderate reuse: 40–60% over hours
- Working set accumulates with request rate × observation window
- Capacity-driven, latency-tolerant
- Best placed: CPU DRAM (LPDDR5x), storage-class memory

Session Context Tier (Intra-Session)

- Active conversation history within sessions
- Very high reuse: ≥95%(system prompt incl) over minutes
- Working set scales with concurrency (concurrency × context × bytes/token)
- Bandwidth-critical, hot data
- Best placed: HBM, spills to DDR/LPDDR

Framework

Inputs: Workload spec (request rate · context length · concurrency · reuse rate) + System / tier spec (per-tier capacity · bandwidth · latency: HBM, DDR/LPDDR, CXL, storage)

1 · Operator decomposition

Hardware-calibrated primitives: GEMM, attention, communication, memory

2 · Working-set decomposition

Two populations: system-prompt (cross-session) + session-context (intra-session)

3 · Capacity-gated effective reuse

Policy-agnostic bounds: random (lower) and oracle (upper)

4 · Tier-aware bandwidth & latency

Per-tier retrieval cost mapped onto the memory hierarchy

Outputs: TTFT · Throughput · Minimum tier capacity · Capacity-vs-bandwidth trade-offs

Suggested CMS scope: workload schema · tier specification · effective-reuse metric · benchmark · reporting template



Validation Platform

Framework validated on a GH200-class tiered memory system

Hardware: GH200 NVL2

- 2× H200 GPU, 288 GB HBM3e
- 2× 72-core Arm Neoverse V2 @ 3.1 GHz
- 2× 512 GB LPDDR5x @ 384 GB/s
- NVLink 900 GB/s CPU-GPU interconnect

Configuration

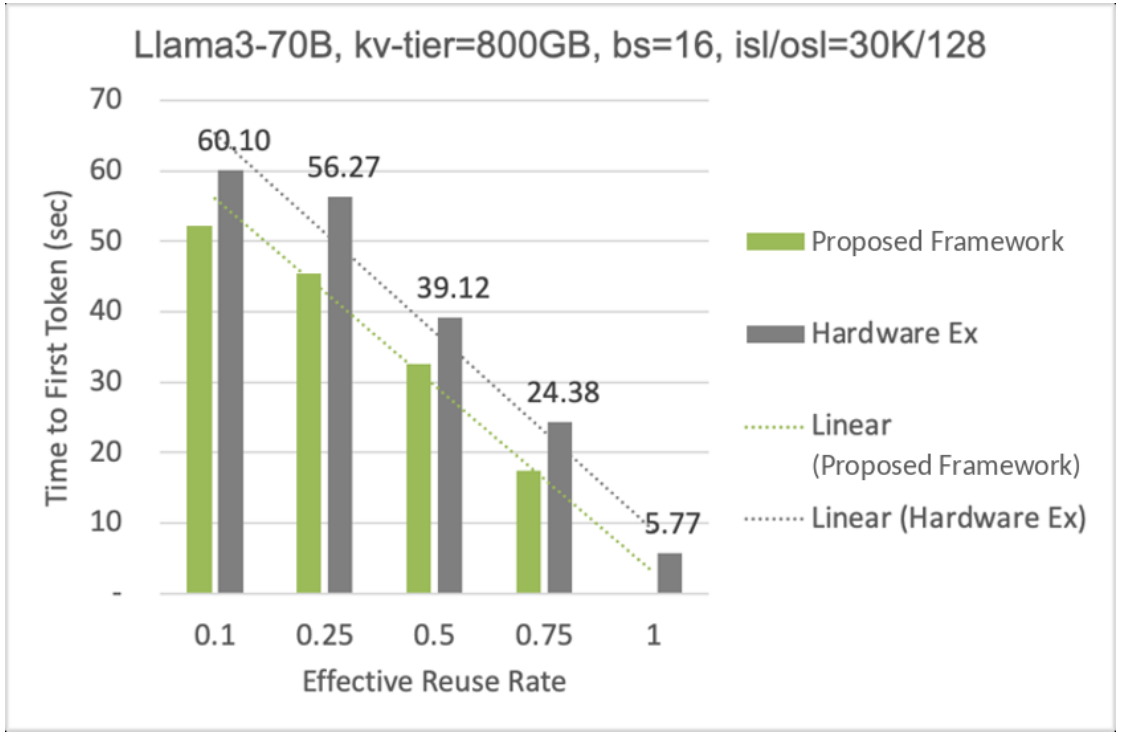
- Model: **Llama 3.1-70B, FP16**
- Parallelism: tensor parallel 2
- Engine: vLLM + LMCache
- ISL/OSL = 30K/128
- prefix tokens = 30K (100% reuse rate)
- RPS: 3.8 req/s
- *Validation focuses first on the reusable-prefix population;* session-context pressure is captured through concurrency scaling.

Validation #1 — Miss Rate Determines TTFT

Latency scales strongly with cache misses

Observations

- Reduced hit rates increase TTFT
- Framework predicts trend accurately
- Effective reuse is a meaningful predictor



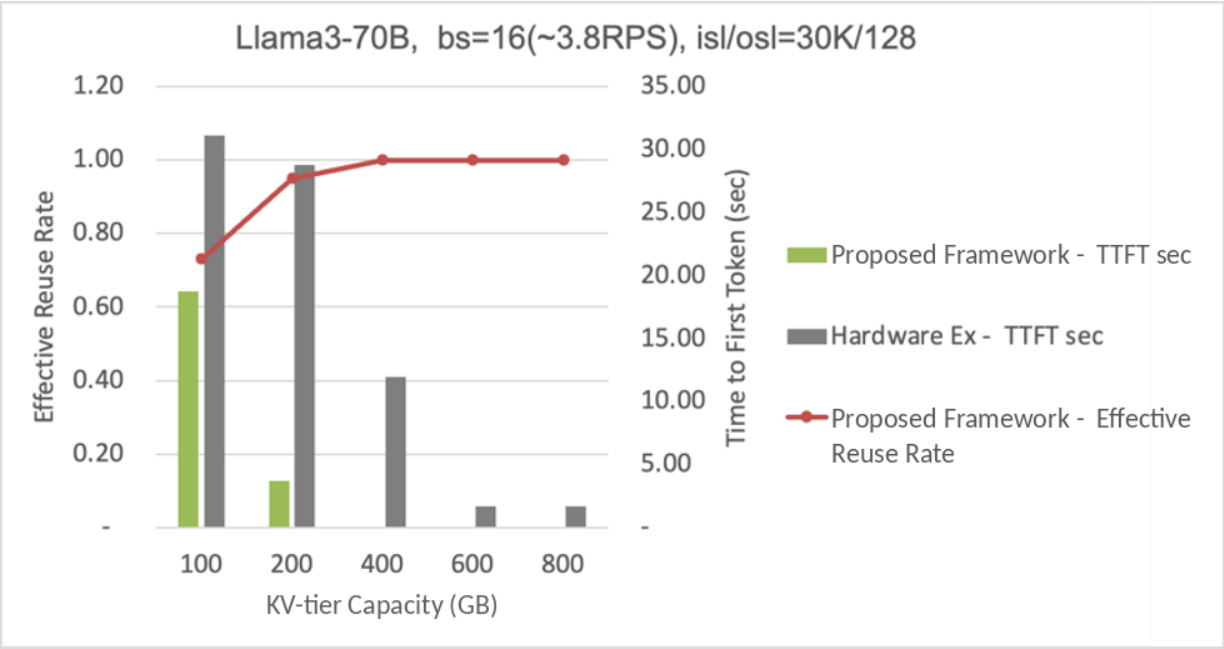
Takeaway
Miss-rate modeling captures first-order latency impact.

Validation #2 — Capacity Creates Thresholds

Memory capacity creates step-function behavior

Observations

- Capacity increases affect up to a threshold
- Reuse suddenly becomes realizable
- TTFT drops dramatically
- Once threshold reached capacity has little effect



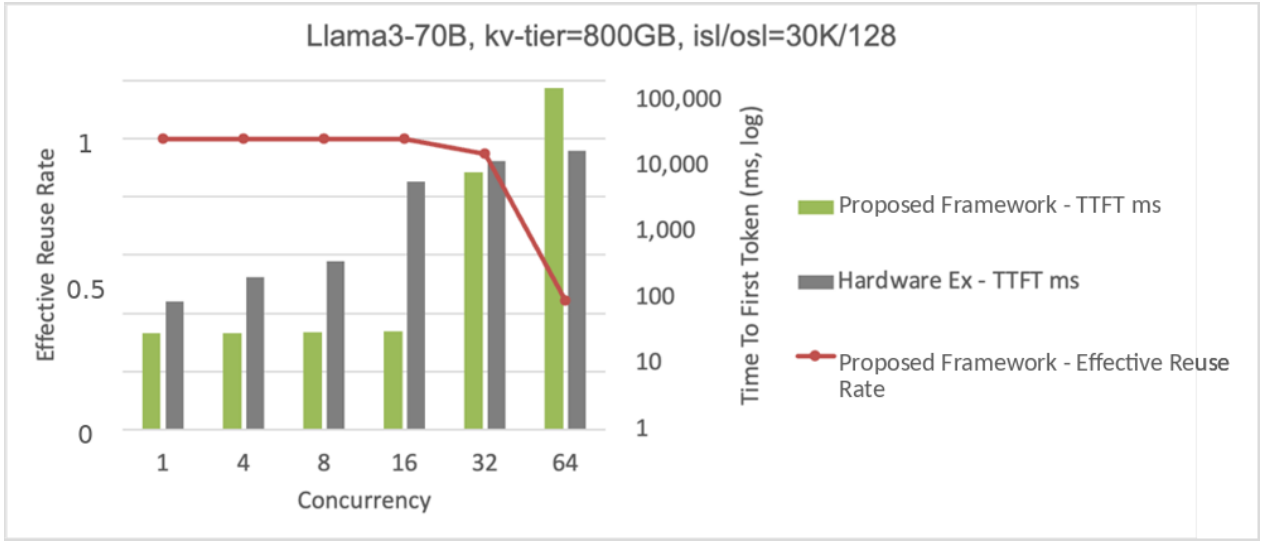
Takeaway
Capacity and performance are not linearly related.

Validation #3 — Concurrency Creates Pressure

Concurrency turns reuse into a scaling problem

Observations

- Effective reuse remains near 100% while the reusable working set fits within available capacity.
- Increasing concurrency expands the active KV working set and eventually exhausts memory capacity.
- Once the capacity threshold is crossed, effective reuse collapses and TTFT increases sharply.



Takeaway
A memory tier sufficient today may fail at production concurrency.

Design Rules Emerging from the Analysis

Design rule	Meaning
KV is multi-population	Shared prefixes and active session context behave differently.
Miss rate drives latency	TTFT improves as miss rate drops
Capacity gates reuse	The rate of reuse depends if the working set fits.
Concurrency must be explicit	Production load changes the working set.

Conclusions

- Capacity-aware reuse is the right abstraction
  - KV consists of multiple populations with different requirements
  - The framework predicts: miss-rate effects, capacity thresholds, concurrency pressure
- Applicable to:
  - HBM
  - DDR/LPDDR
  - CXL
  - storage-class memory

- KV-cache offload is not a binary optimization. It is a placement problem across a memory hierarchy. Memory hierarchy design will increasingly determine scalable LLM inference performance.

Call to Action: Treat KV-Cache Offload as a Memory-Hierarchy Problem

Gaps OCP CMS can address

- No vendor-neutral workload schema, capacity-aware reuse metric, tier-comparison method, or results format.
- **Completed**
- Analytical framework and initial GH200-class dense-model validation.
- **Remaining**
- Validate MoE and new KV optimizations across more memory tiers and multi-vendor platforms.
- **Micron's proposed inputs**
- Share framework, capacity-gated effective-reuse metric, and validation data, published tiered-memory analysis paper.

Suggested CMS path: align scope → co-draft v0.1 with members → validate with partners → review as a group

Thank You!

