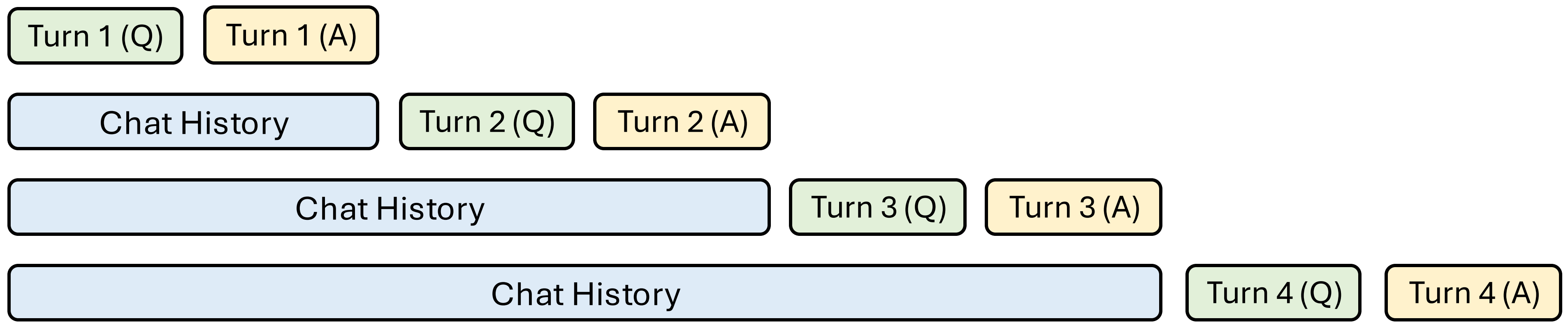
Efficient LLM Serving with Context Caching

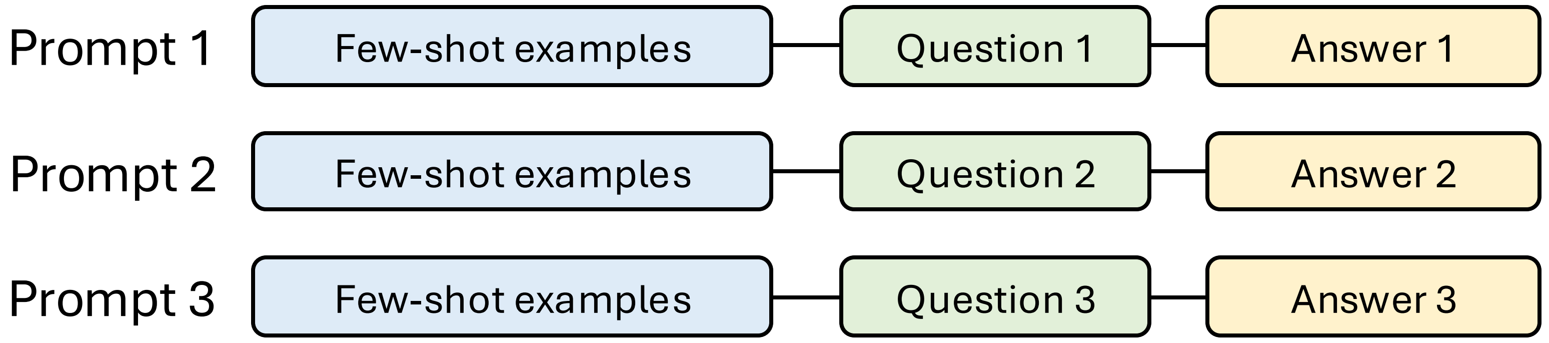
Zhiqiang Xie
Stanford & NVIDIA

Content Reuse is Pervasive

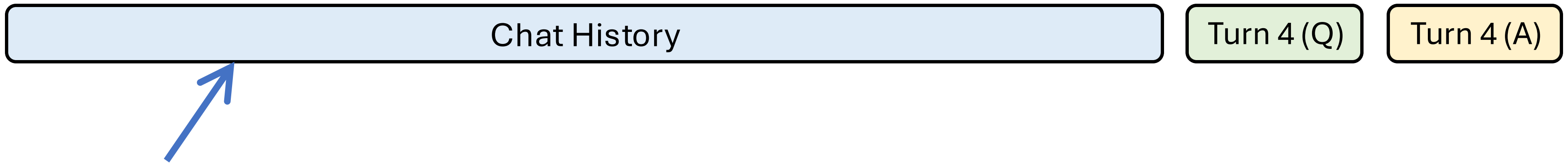
(a) Multi-turn chat



(b) Few-shot learning



Context Caching



KV Cache

- Some reusable intermediate tensors
- Can be very large (>20GB, larger than model weights)
- Only depends on the prefix tokens

Priorly: Discard KV cache after an LLM call finishes

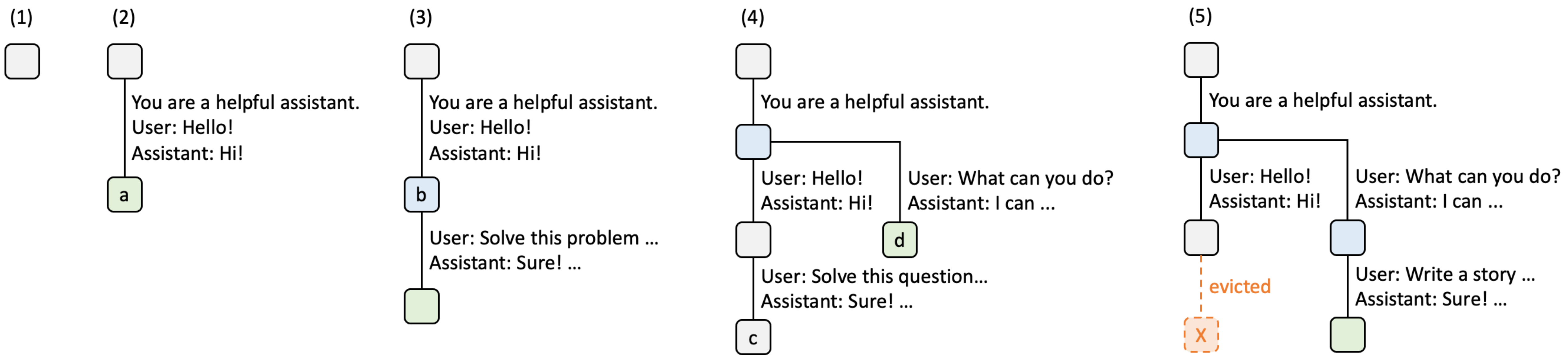
Optimization: Keep and organize the historical KV cache of all LLM calls

Caching is Huge for LLM Serving!

“Our system achieves a **95% cache rate**, further reducing inference cost.”
-- from Character AI

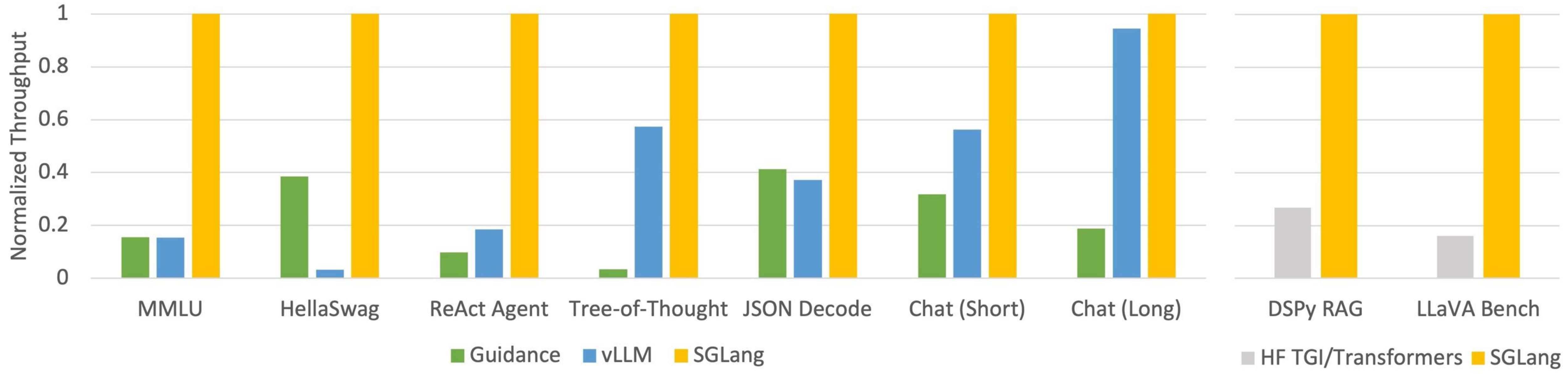
“Even without any optimization, historical data shows that users **save over 50% on average**.”
-- from DeepSeek

Context Caching with RadixAttention

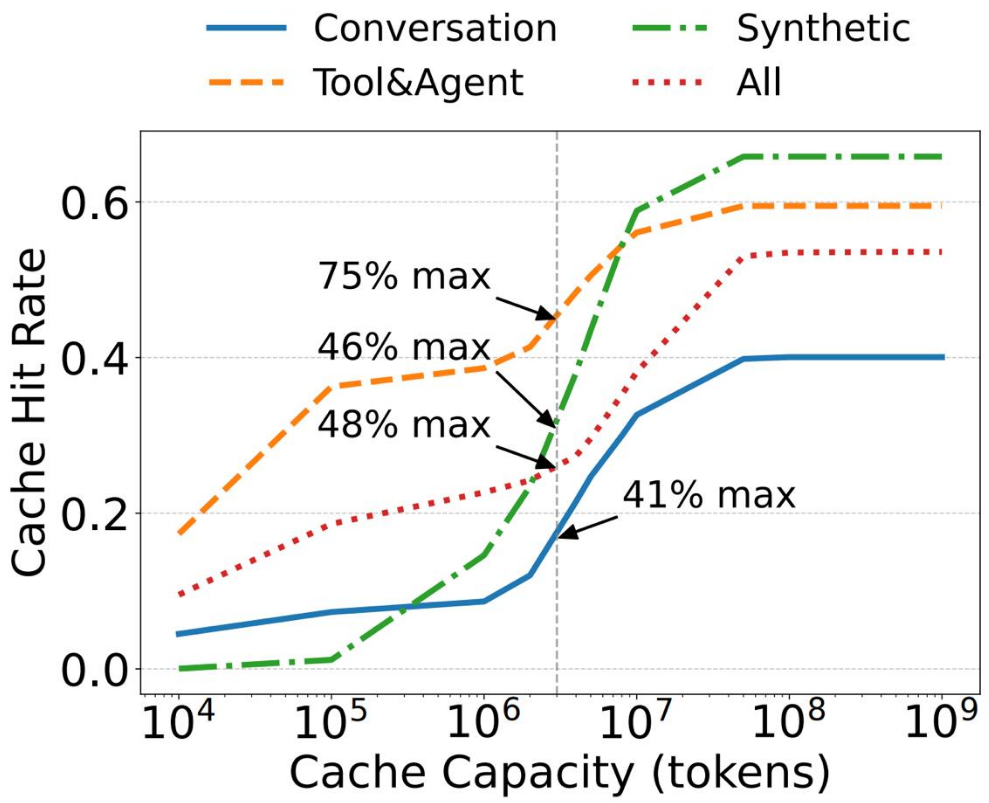


RadixAttention enables **efficient prefix matching, insertion, and eviction.**

Performance Benefit



Context Caching: a Capacity Problem

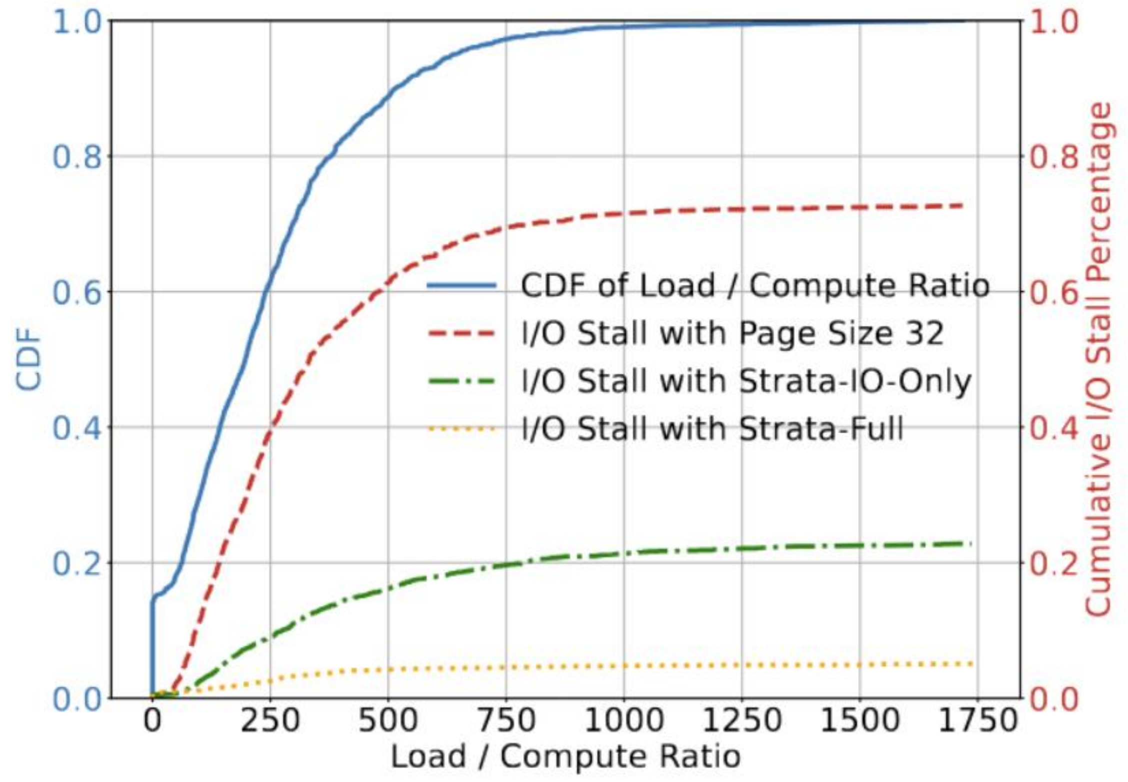


With **larger model** and **longer context**, it became more challenging to obtain caching benefit with on-device HBM only

=> KV cache goes hierarchical

Figure from Mooncake (FAST '25)

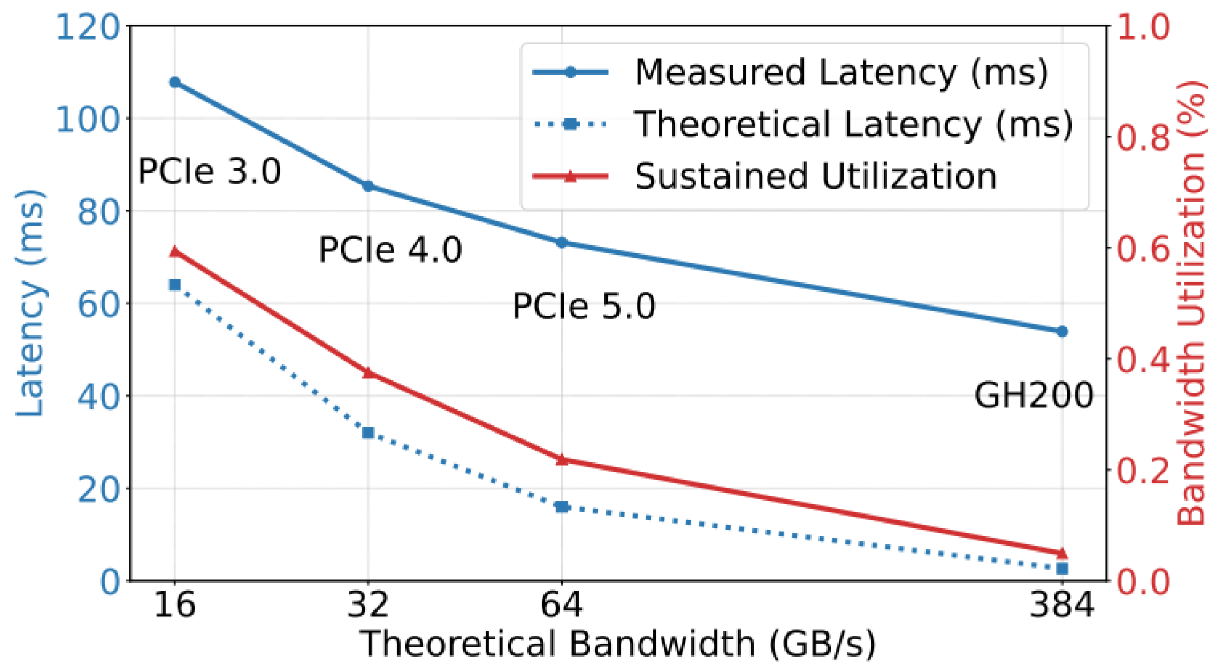
Hierarchical Caching: I/O and Scheduling Challenge



- Inefficient I/O in standard practices can cause up to 74% memory stall
- Even with optimized I/O, memory stall can still be significant (~24%) with insufficient scheduling

Benchmark profile for Qwen2.5-14B on the LooGLE dataset

Hierarchical Caching: an I/O Problem



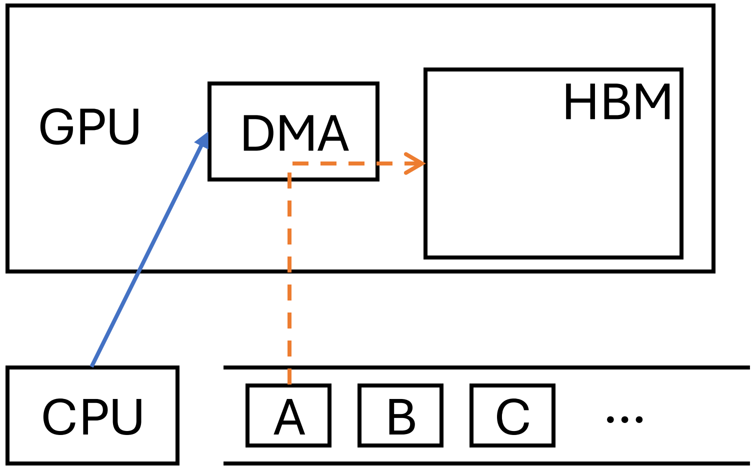
Latency of loading KV caches of 8192 tokens (using page size 32) of Llama-3.1-8B from CPU to GPU.

Marginal latency improvement due to **low bandwidth utilization** despite improvement on available bandwidth

Hierarchical Caching: an I/O Problem

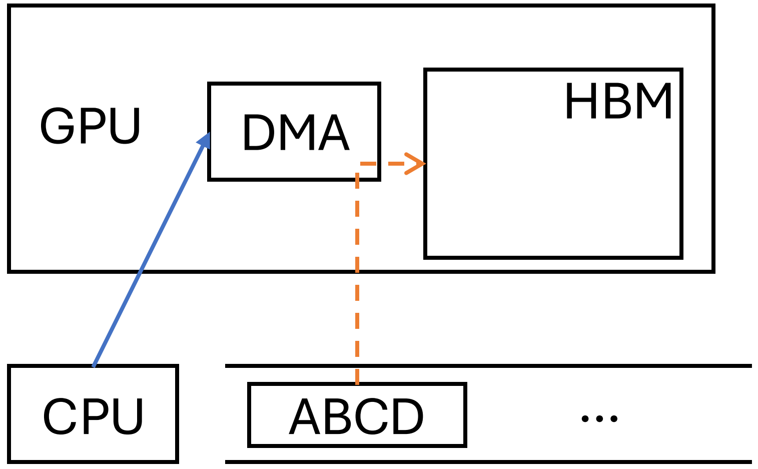
Little's law:

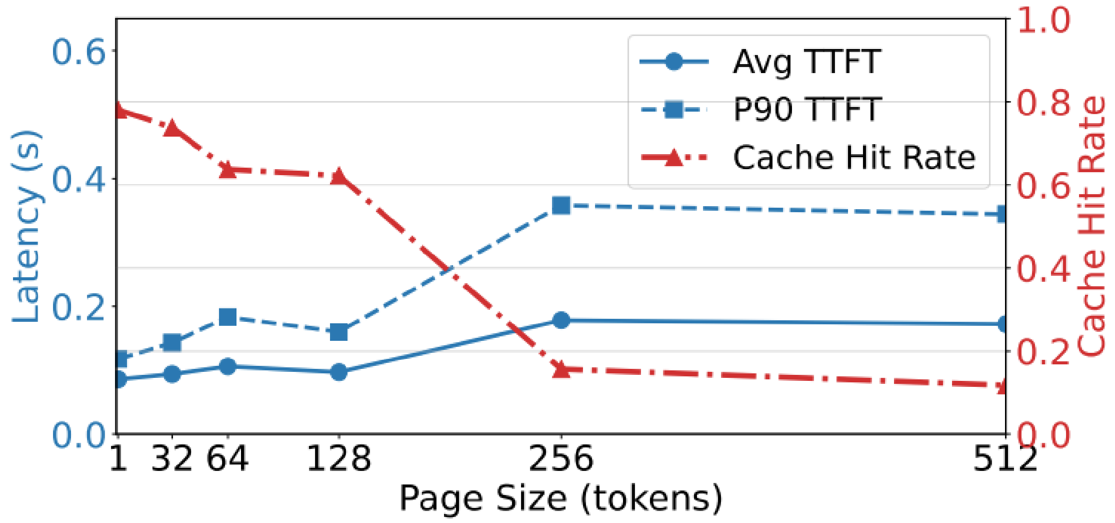
Throughput = Concurrency * Size / Latency



Fragmented KV caches cause excessive small data transfer which bounded throughput due to CPU communication latency and limited parallelism

Hierarchical Caching: an I/O Problem





Large page sizes comprise cache hit rate

Hierarchical Caching: a Scheduling Problem

General perception of KV cache layer wise overlapping:

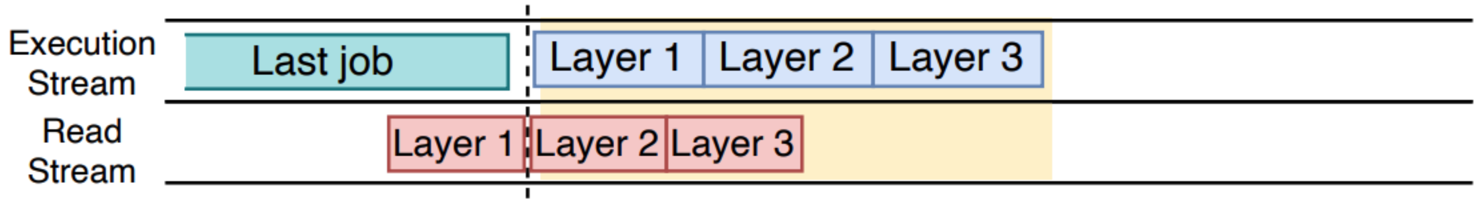
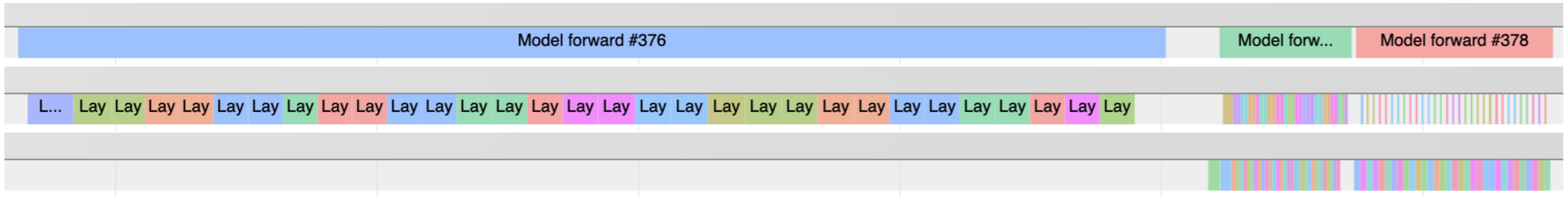


Figure from CacheAttention (ATC'24)

The real image is highly skewed:

Balanced load and compute

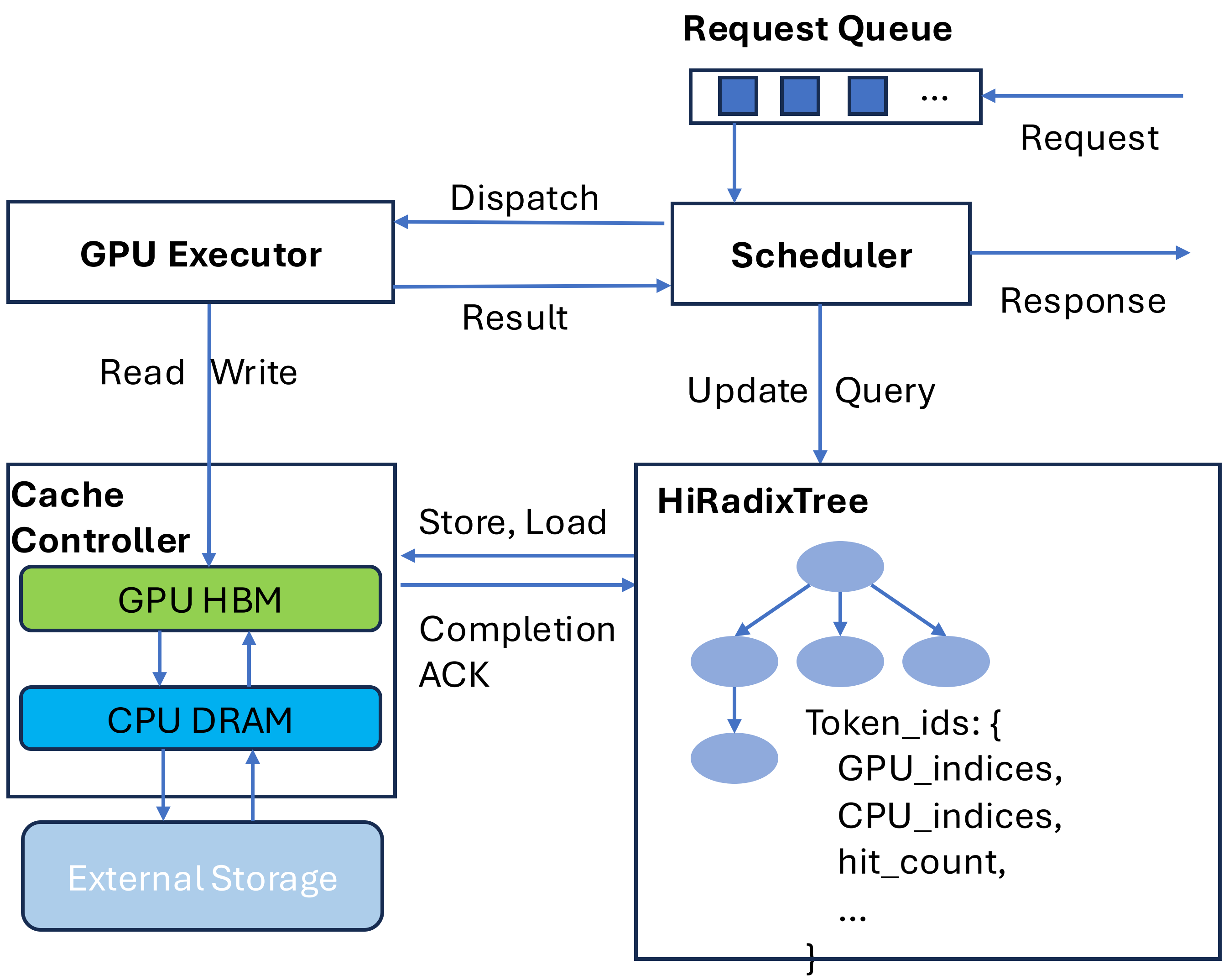


No loading from host

Memory stall due to loading

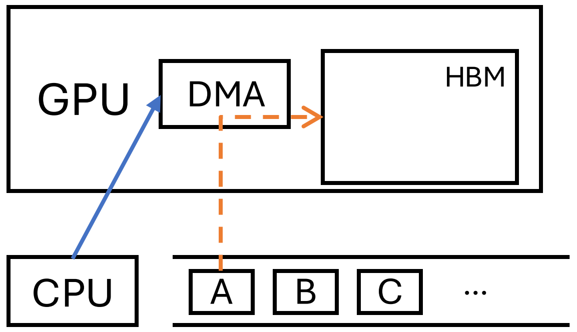
Traces for a run of Loogle wiki-QA dataset

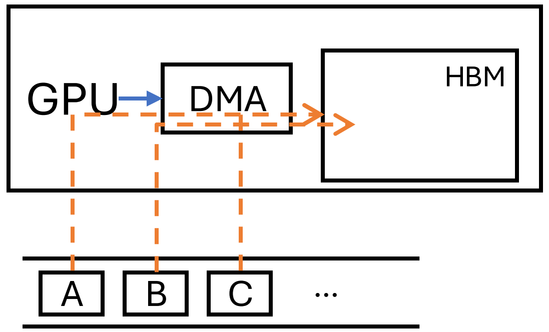
Strata: efficient data plane and intelligent control plane



- Efficient I/O data plane using specialized CUDA kernels
- Intelligent control plane achieves resource-aware scheduling and effective latency hiding

GPU-assisted KV Cache I/O

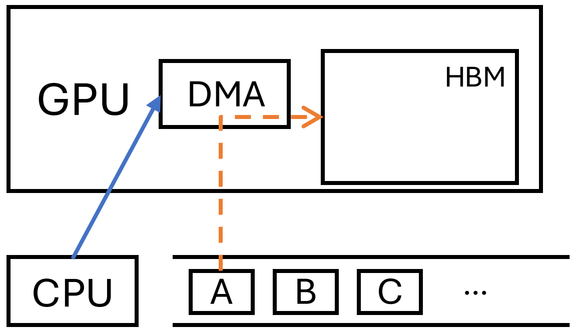


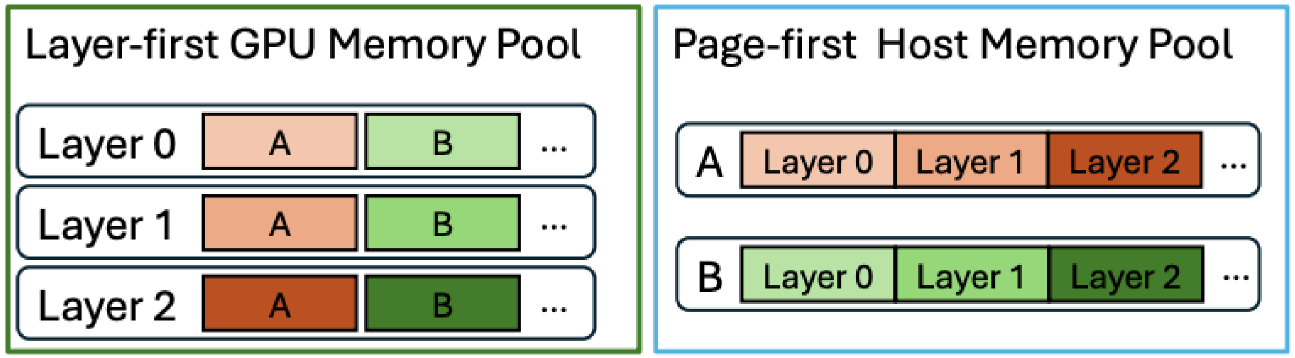


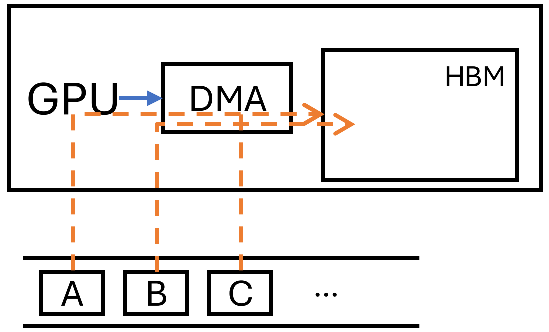
Throughput = Concurrency * Size / Latency
By initiating data transfer from GPU:

- Finer granularity (128-byte load instruction)
  - Effectively page size agnostic
- Lower latency
- More parallelism

Memory Layout Decoupling

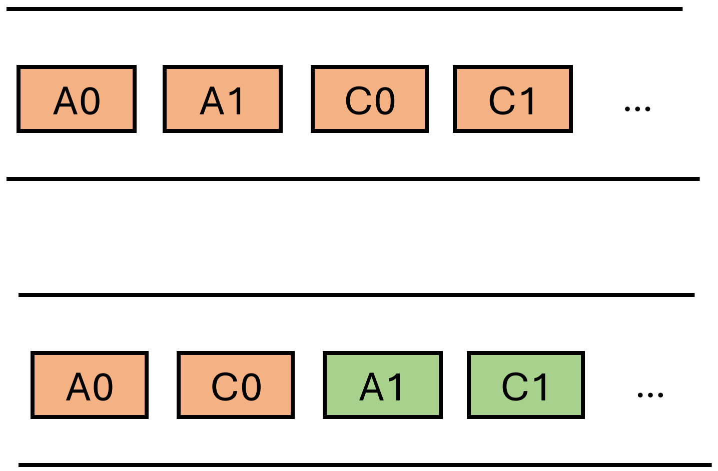






The data transfer kernel also grants us the flexibility to decouple the memory layout of CPU and GPU

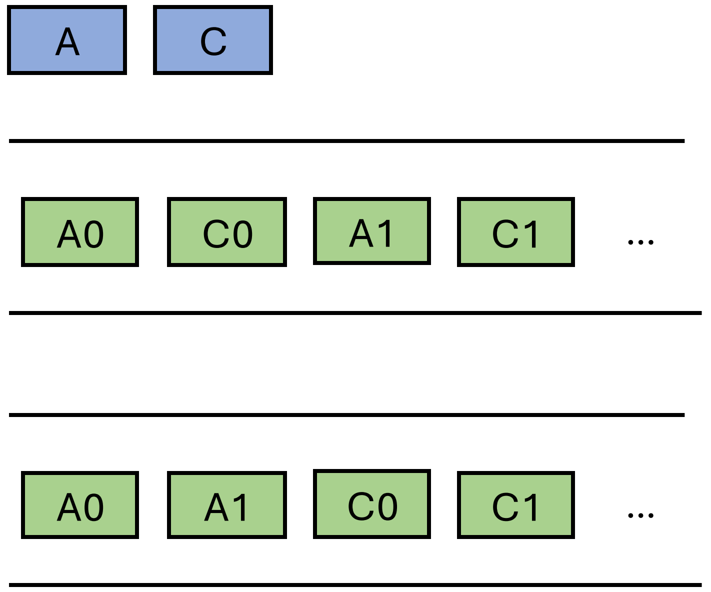
Eliminate Delay Hit to Reduce Cache Miss



Delay hit: while A1 and C1 should have been cache hit, it take time for cache of A0 and C0 to become effective

=> To defer certain requests for higher hit rate

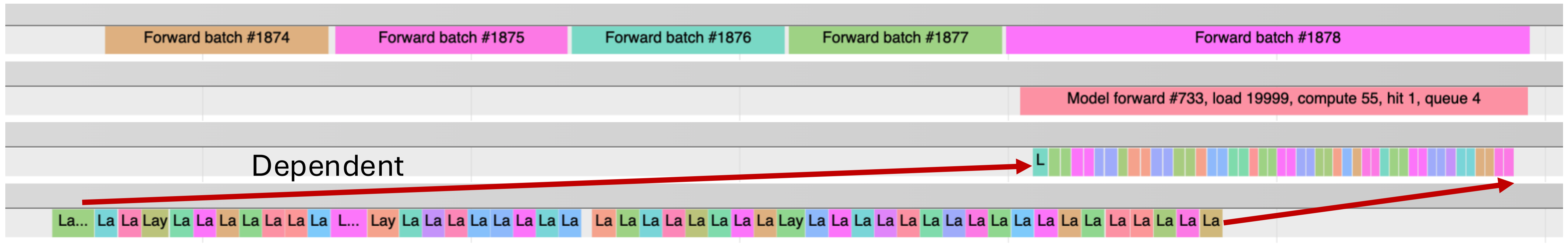
Balanced Batch Formation



Bundle hit: while hit rate remains the same, bundling requests together can further eliminate redundant memory footprint

=> To prioritize certain requests for better compute / IO balance

Hide I/O Stall with Bubble Filling


Forward batch #1874
Forward batch #1875
Forward batch #1876
Forward batch #1877
Forward batch #1878
Model forward #733, load 19999, compute 55, hit 1, queue 4
Dependent
L
La... La La Lay La La La La La La La L... Lay La La La La La La La La La La La La La La La La La Lay La La La La La La La La La La La La La La La La La La La

When I/O stall is inevitable, we place another batch to **fill the bubble**

=> To overlap I/O stall with computation

Performance Improvement with Strata

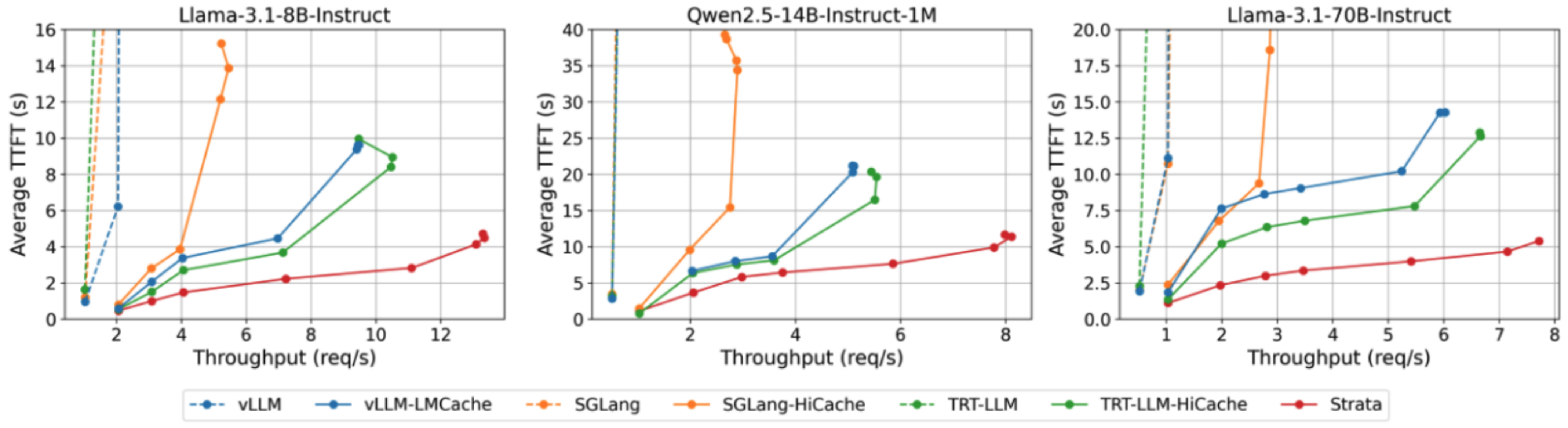
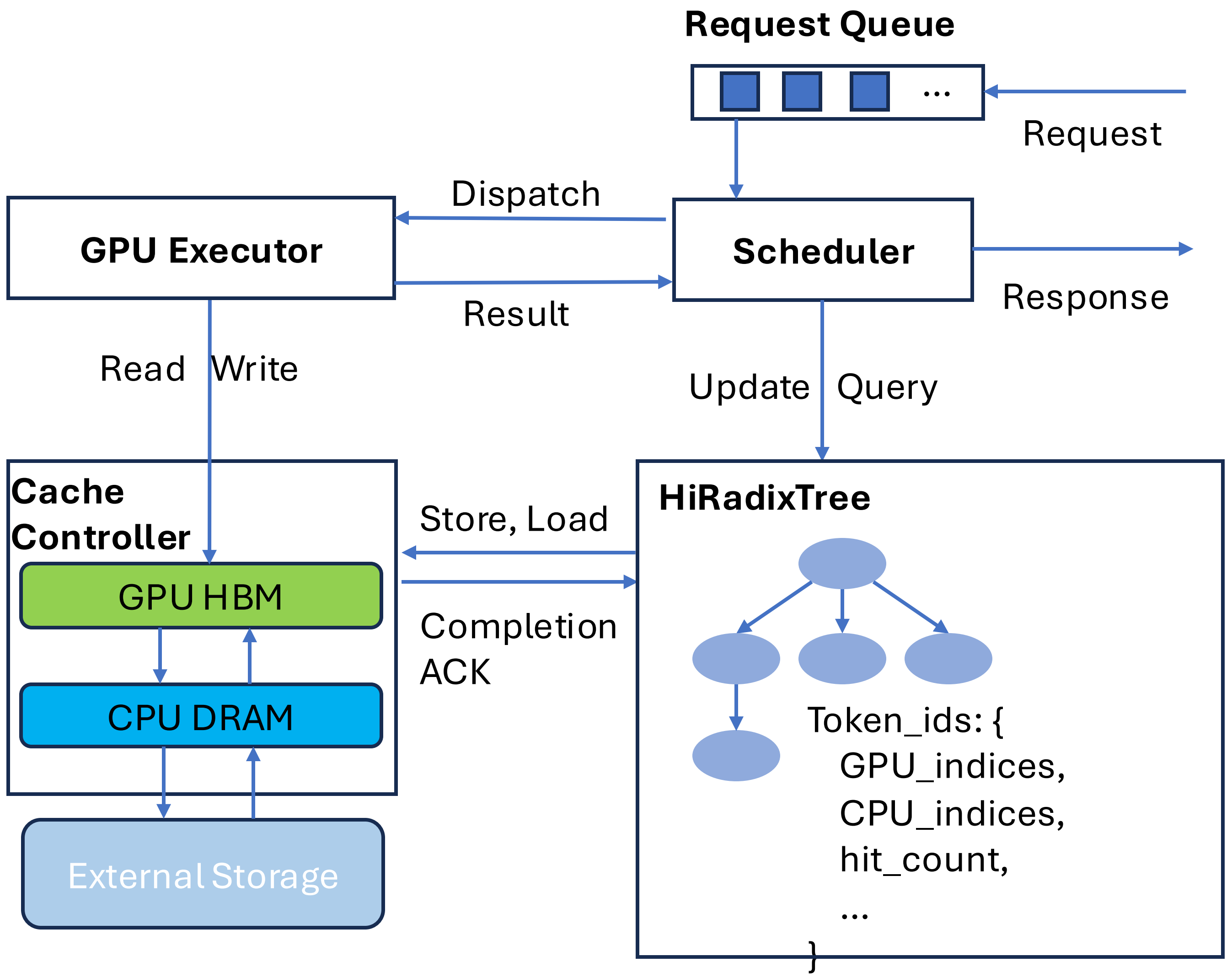


Figure 5. End-to-end benchmark performance with LooGLE dataset on H200.

Up to 5x, 5x and 3.75x higher throughput at the same latency comparing to SGLang-HiCache, vLLM-LMCache and TRT-LLM-HiCache respectively

Strata: efficient data plane and intelligent control plane



- Efficient I/O data plane using specialized CUDA kernels
- Intelligent control plane achieves resource-aware scheduling and effective latency hiding

https://github.com/sgl-project/sglang