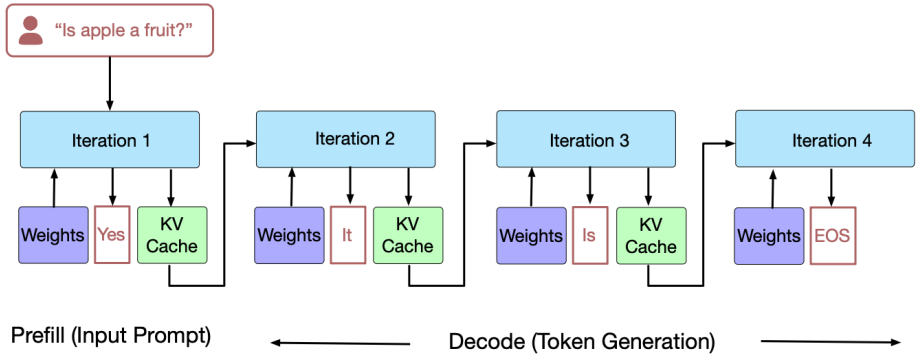
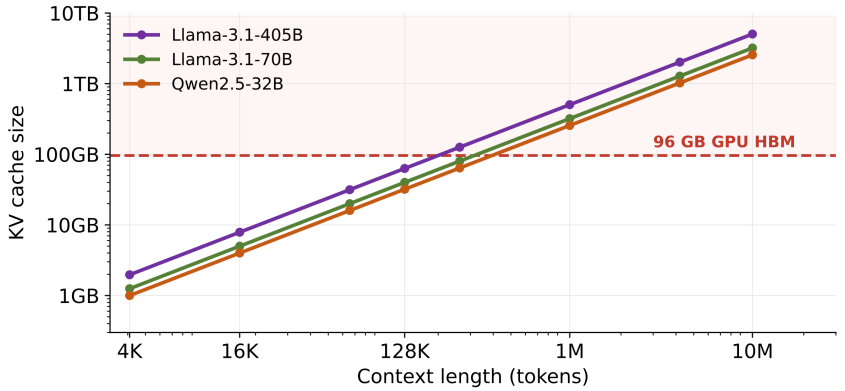
No Buffer, No Bottleneck: Efficient Zero-Copy KV Cache Offloading for Long-Context LLMs

Shutian Luo and Haiying Shen
University of Virginia

Long Contexts Make KV Cache the Bottleneck

- Long-context inference stores keys and values for all previous tokens
  - Avoids recomputing previous tokens
  - KV cache grows linearly with context length





Million-token contexts require hundreds of GB to multiple TB of KV cache per request.

Cheap Storage Meets Fast Compute

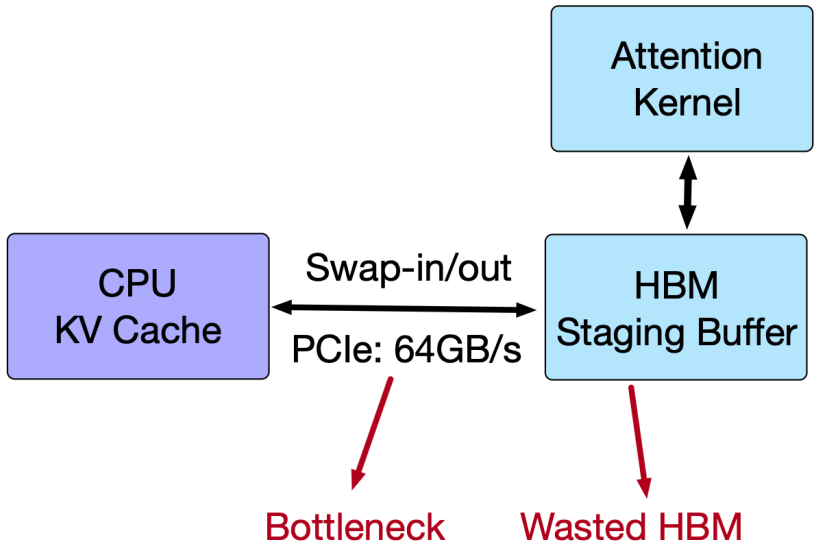
- Store KV in CPU DRAM → Compute attention on GPU

CPU DRAM	GPU HBM
Large capacity	High bandwidth
Low cost	Limited capacity
TB scale	~100 GB/GPU
Best for KV storage	Best for attention compute

Can direct CPU-memory access be fast enough?

Existing KV Offloading Relies on HBM Staging

- Slow PCIe makes CPU memory a backup tier
- Dynamic copy nodes disrupt CUDA Graph replay



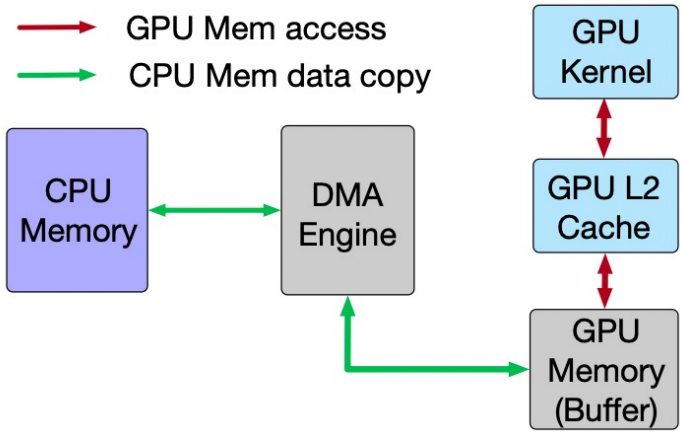
New Hardware, New Opportunity

- Before
  - CPU memory → DMA copy → HBM staging → Kernel
- With NVLink-C2C
  - CPU memory → Direct kernel load

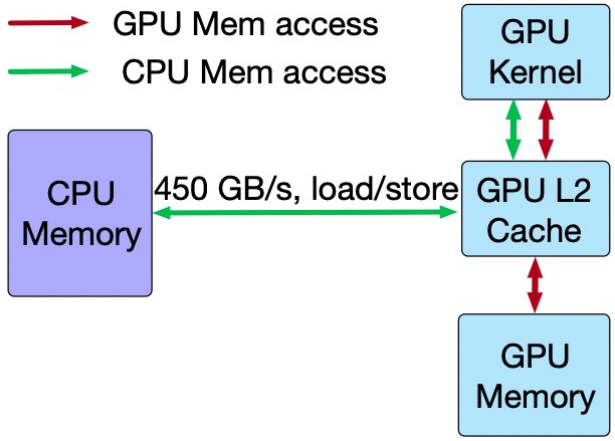


CPU-GPU Bandwidth 450GB/s (7× PCIe Gen5)

NVIDIA Superchip
(GB200/GH200 with NVLink-C2C)



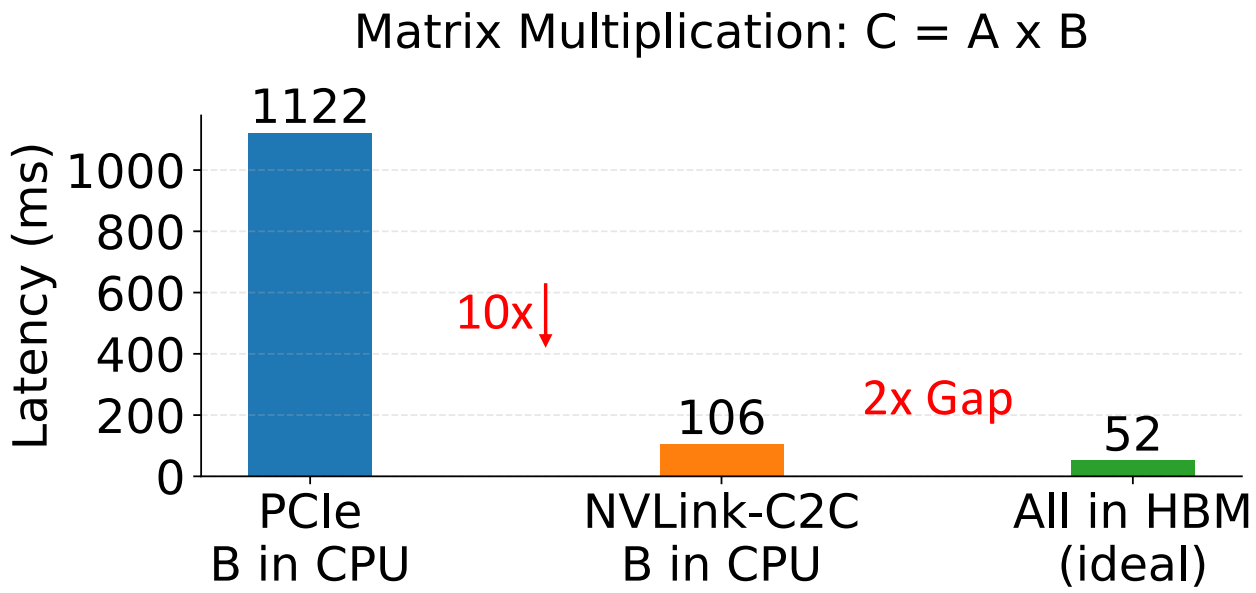
CPU memory as a backup



CPU memory as an extension

Fast Interconnect, but still a Gap

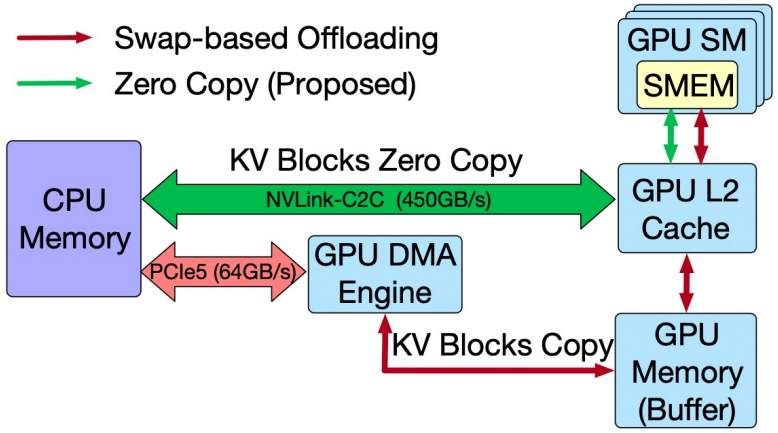
- Same GEMM kernel; only B’ s memory location changes



Takeaway: Hardware shrinks the gap 10× — kernels must close the rest.

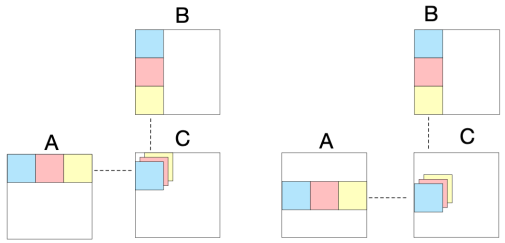
Zero-copy Made Practical: Memory-Kernel Co-design

- Redesign access patterns for CPU-resident KV
  - Fetch each KV tile across C2C once
  - Reuse it on chip through shared memory
- Two targets
  - No buffer: no HBM staging
  - No bottleneck: C2C traffic no longer dominates



Observation 1: CPU-memory-aware Tiling

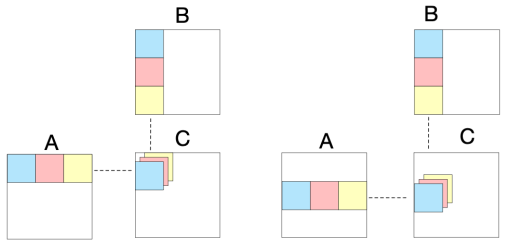
- Tiling background: stage a tile in SMEM, reuse it
- GEMM: C = A × B, with B in CPU memory
  - Naive zero-copy: default tiling fetches B repeatedly



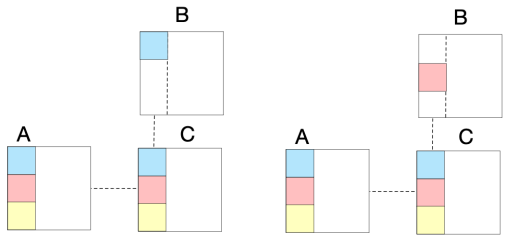
Default Tiling

Observation 1: CPU-memory-aware Tiling

- Tiling background: stage a tile in SMEM, reuse it
- GEMM: C = A × B, with B in CPU memory
  - Naive zero-copy: default tiling fetches B repeatedly
  - CPU-aware: B's tile stays in SMEM, reused



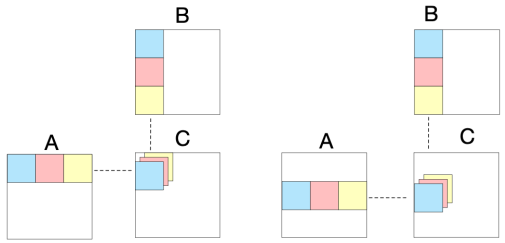
Default Tiling



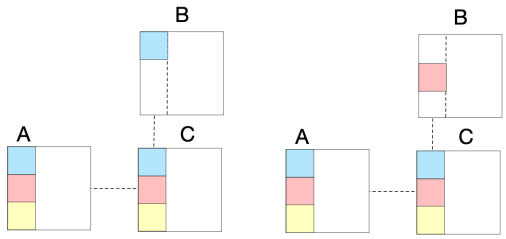
CPU-memory-aware Tiling

Observation 1: CPU-memory-aware Tiling

- Tiling background: stage a tile in SMEM, reuse it
- GEMM: C = A × B, with B in CPU memory
  - Naive zero-copy: default tiling fetches B repeatedly
  - CPU-aware: B's tile stays in SMEM, reused
- Data movement shifts from C2C to HBM
  - C2C traffic: 33 GB → 0.4 GB
  - HBM traffic: 23 GB → 89 GB
  - Result: kernel becomes HBM-bandwidth-bound



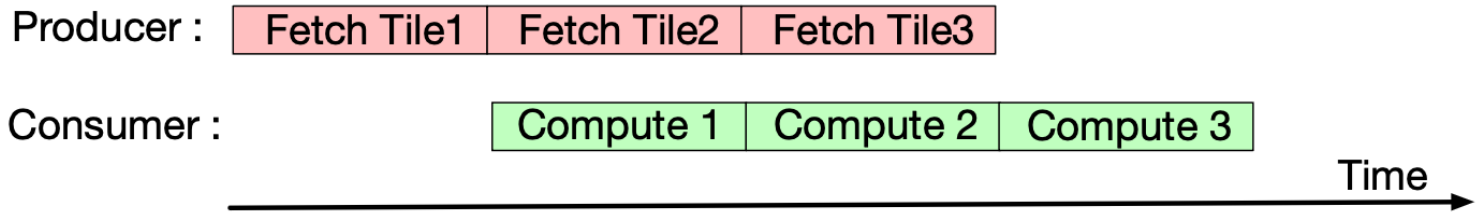
Default Tiling

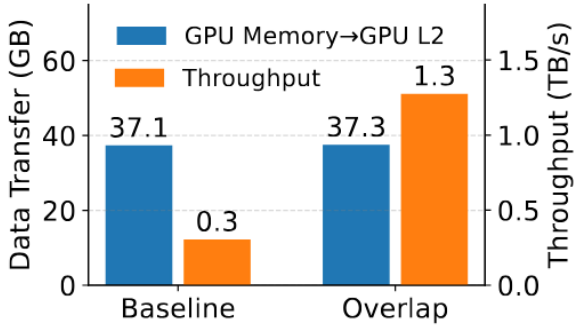


CPU-memory-aware Tiling

Observation 2: Hide HBM Access Behind Computation

- Warp-level pipelining

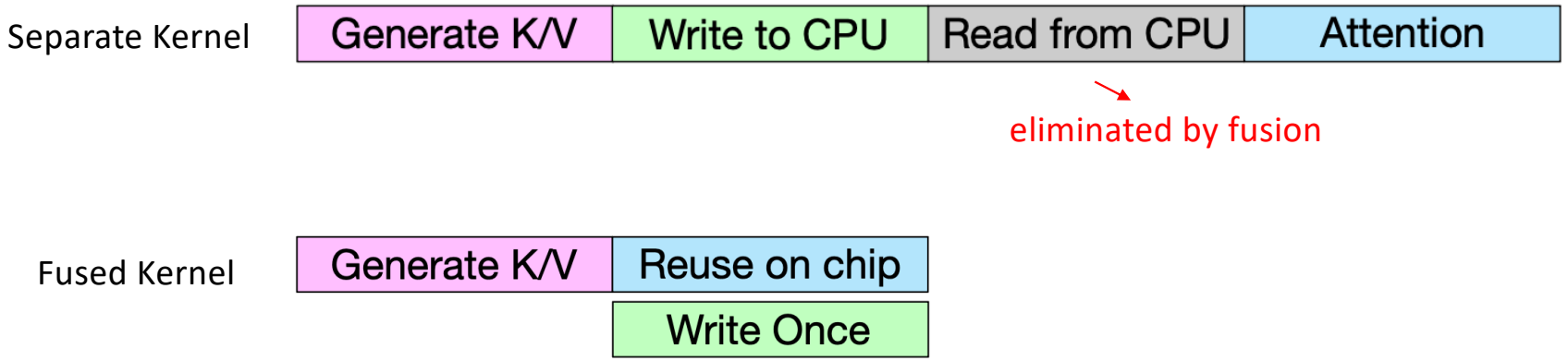




HBM Throughput: 0.3 → 1.3 TB/s (4.3×)

Observation 3: Kernel fusion

- Fuse projection and attention to eliminate the host round trip



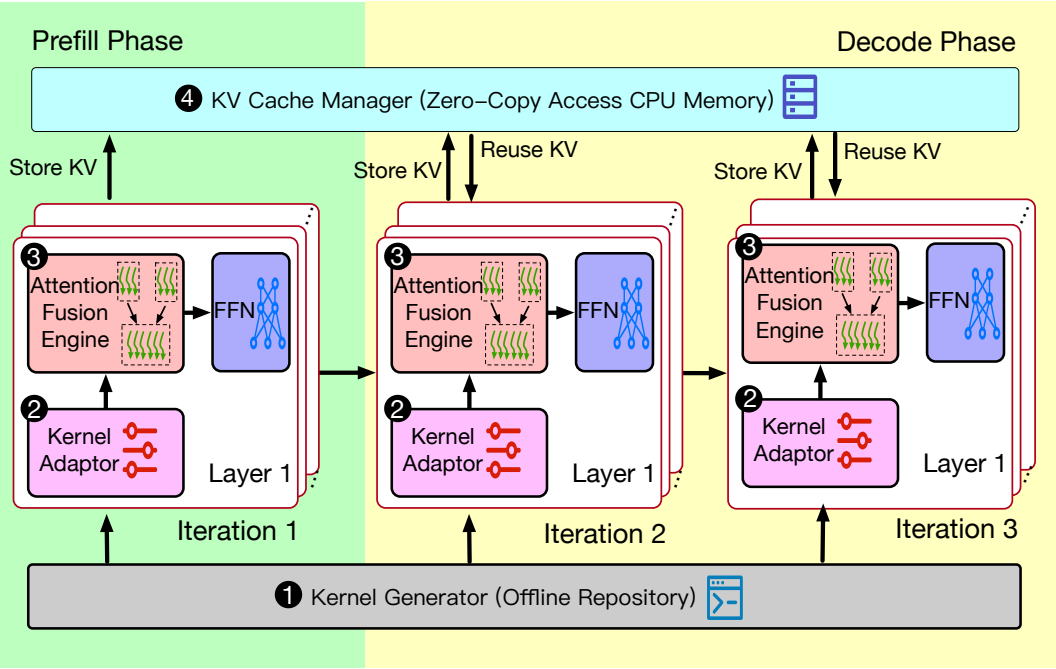
DirectKV System Architecture

❶ Kernels are generated offline

❷ Adaptor selects kernels

❸ Projection/RoPE/attention are fused

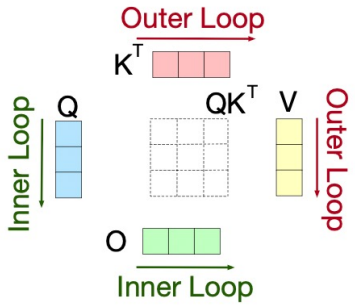
❹ KV cache stays in CPU memory

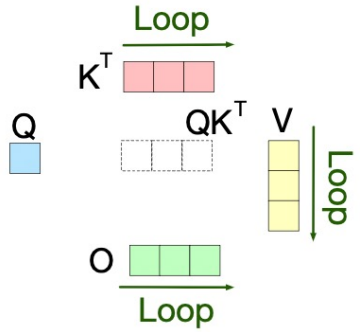


Phase-Specific CPU-Aware Tiling

- Use phase-specific tiling to minimize KV traffic

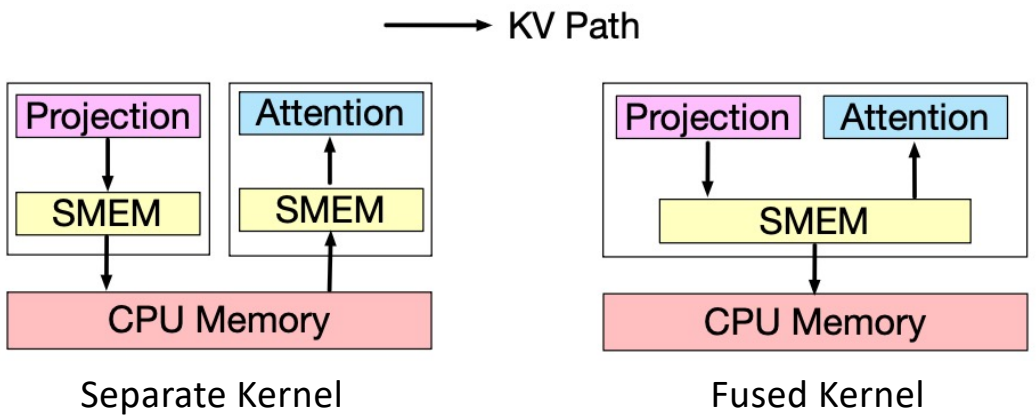
Prefill	Decode
Long query	One-token query
2D Q and KV tiling	1D KV sweep
Exploit reuse across Q tiles	Stream each KV tile once





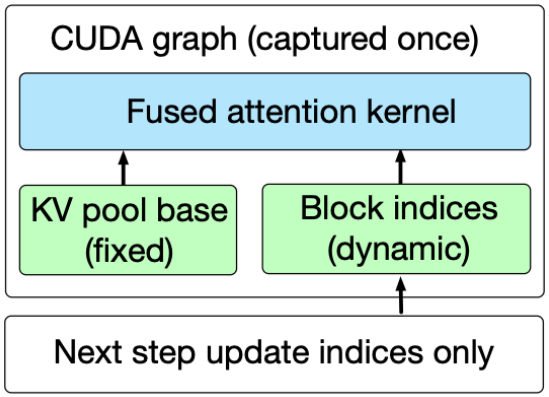
Warp-pipelined Fused Kernel

- One kernel, three warp roles
  - Producer: fetch old KV tiles or generate new K/V
  - Consumer: run attention on the current tile
  - Storer: write new K/V to CPU memory once
- Shared memory enables on-chip reuse across projection, attention, and write-back



Fused Kernel Preserves CUDA Graph Replay

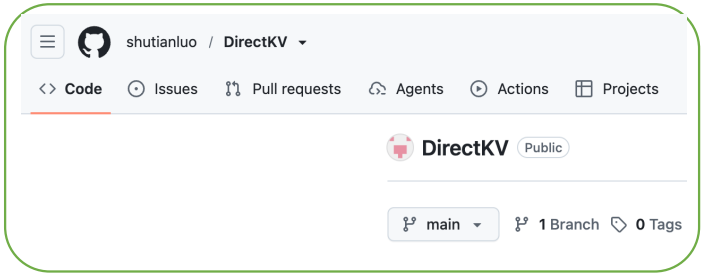
- KV addresses are computed inside the kernel
  - Kernel takes a KV-pool base pointer and block indices
- Graph topology remains unchanged
  - No recapture; near-zero launch overhead



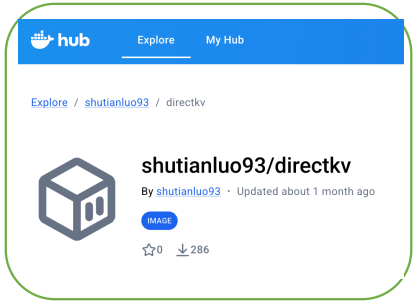
Same graph, different KV blocks

Evaluation Setup

- Platform
  - NVIDIA GH200 with 96 GB HBM via NVLink-C2C interconnect
- Models
  - Llama-3.1-8B, OPT-13B, OPT-30B
- Workloads
  - ShareGPT + Alpaca, Poisson arrivals up to 30 req/s
  - Context lengths from 1K to 32K tokens
- Baselines
  - SGLang: HBM-resident KV cache
  - Pie [arXiv'24]: swap-based KV offloading
  - FlexGen [ICML'23]: cost-model offloading over GPU/CPU/disk
  - Neo [MLSys'25]: hybrid CPU/GPU attention offloading

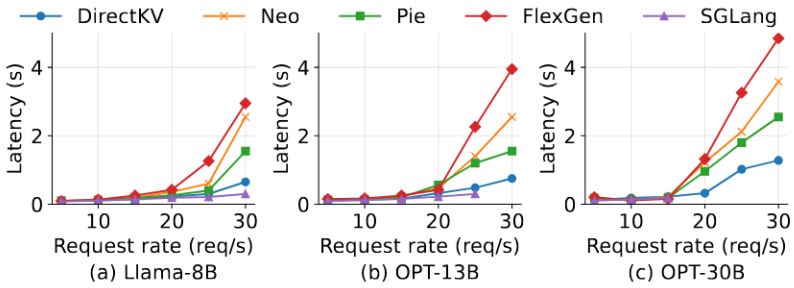




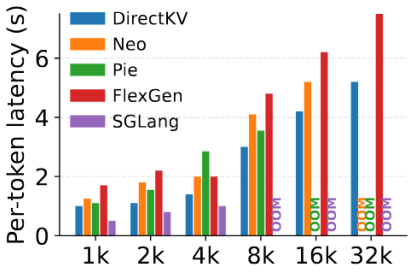


End-to-end Performance

- 1.2× faster on average
- Up to 2.4× under high load

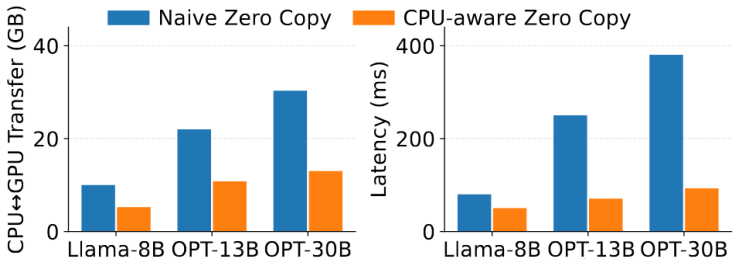


- DirectKV serves long contexts the others cannot.
  - SGLang is fastest while KV fits in HBM — but OOMs at long contexts

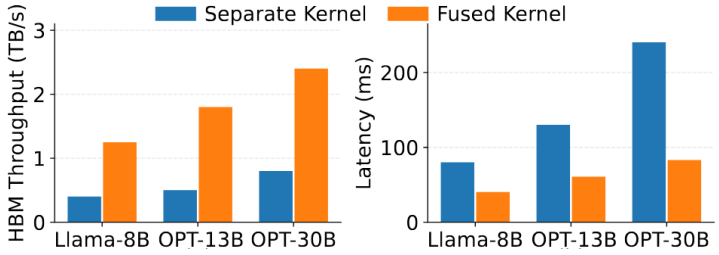


Component-level Evaluation

- Benefits of CPU-memory-aware tiling
  - CPU-aware zero-copy reduces CPU–GPU transfer volume by up to 50% compared with naive zero-copy.



- Advantage of Fused Kernel
  - Fused design sustains up to 3.5× higher HBM throughput and delivers 50–65% lower latency across models



Summary

- Long contexts require HBM capacity expensive
- NVLink-C2C makes CPU memory a practical GPU working tier
- Memory-kernel co-design makes zero-copy efficient
  - CPU-memory-aware tiling
  - Warp-level pipelining
  - Kernel fusion

Takeaway: NVLink-C2C makes CPU memory
a practical extension of GPU memory.
Memory–kernel co-design makes that access efficient.

Thank You

Email: ksy8xs@virginia.edu