The Hidden Bottleneck in MLA Serving: Reconstruction GEMMs, INT4 Quantization, and the L2 Cache Barrier

Robert Zhang[1]

[1]Purdue University
[1]zhan4808@purdue.edu

Abstract

Multi-head Latent Attention (MLA) compresses KV cache via low-rank latent projections, greatly reducing memory traffic for the attention kernel. We show that this compression creates a *new* dominant bottleneck: the **reconstruction GEMMs**, weight-absorbed batch matrix multiplications that recover full-dimensional K/V from compressed latents, consume **61% of attention-layer time** at batch size 1 on DeepSeek-V3-scale architectures (128 heads, 61 layers), exceeding the attention kernel itself. These BMMs are memory-bound at all practical batch sizes (arithmetic intensity 1–93 vs. H100 roofline crossover at 295).

We investigate INT4 weight-only quantization as a mitigation. Selective INT4 of reconstruction weights preserves quality (ΔPPL = +0.051 on wikitext-2, vs. +6.057 for naive whole-model INT4). However, a custom batched W4A16 Triton kernel achieves only 0.49× of cuBLAS FP16. INT4 underperformance arises from two interacting factors: **L2 cache residency** of FP16 weights (the 16 MB matrix fits in H100's 50 MB L2, invalidating the HBM-bound roofline assumption) and **dequantization-induced compute saturation** (the INT4 kernel is SM-bound, not memory-bound). We validate the L2 effect causally: scaling weight size from 8 MB to 128 MB across the L2 boundary, the INT4/FP16 gap drops from 1.9× to 1.08×, with NCU confirming FP16 is DRAM-bound (83% utilization) while INT4 is SM-bound (79%). To our knowledge, this is the first characterization of this failure mode.

We validate our profiling methodology through an independent FlashInfer vs. Triton kernel comparison on Llama-3-8B, reproducing known results (89% vs. 80% HBM utilization in decode; 2.6× prefill gap from TMA hardware loads) and contributing new NCU-level root cause analysis. All code and data are publicly available.

1 Introduction

Multi-head Latent Attention (MLA) [4] compresses the KV cache through low-rank latent projections, achieving a 7.1× reduction in KV memory traffic and powering frontier models including DeepSeek-V2 and V3 [5]. But this compression requires full-dimensional K and V to be *reconstructed* from compressed latents at inference time via weight-absorbed batch matrix multiplications (BMMs). Prior work has focused on MLA's memory savings; the runtime cost of reconstruction has not been studied.

We show that this cost is the **dominant bottleneck** in MLA attention layers. On DeepSeek-V3-scale architectures (128 heads, 61 layers), reconstruction GEMMs consume 61% of per-layer time at batch size 1, exceeding the attention kernel itself. This overhead is fixed per query, independent of KV length, and invisible to standard attention benchmarks. MLA's compression, in other words, shifts the bottleneck from KV cache reads to weight-matrix reads for reconstruction.

Since reconstruction is memory-bound (arithmetic intensity 1–93 vs. H100 roofline crossover at 295), INT4 weight-only quantization is a natural mitigation. Selective INT4 preserves quality (ΔPPL = +0.051), but a custom Triton W4A16 kernel achieves only 0.49× of cuBLAS FP16, far from the 3.9× predicted by roofline. We trace this to two interacting factors, which we term the **L2 cache barrier**: the 16 MB reconstruction weight matrix fits within H100's 50 MB L2 cache, so FP16 weights are served at multi-terabyte-per-second effective bandwidth rather than HBM's 3.35 TB/s, and simultaneously, INT4 dequantization shifts the kernel from memory-bound to compute-bound. Reducing precision saves HBM bandwidth that was never the bottleneck, while adding compute that becomes the new one (Figure 6). We validate the L2 effect causally by scaling weight size across the L2 boundary: the INT4/FP16 time ratio drops from 1.9× at 8 MB to 1.08× at 128 MB, with a sharp transition at 40–48 MB (Figure 7). MLA's compression paradoxically makes reconstruction weights *too small* to benefit from quantization.

We validate our profiling methodology through an independent FlashInfer [13] vs. Triton [10] kernel comparison on Llama-3-8B (Section 3), reproducing known performance gaps and contributing new NCU-level root cause analysis.

Contributions:

- **Reconstruction GEMM overhead**: first quantification showing 61% of MLA attention-layer time at bs=1 (2.17 ms/token across 61 layers).
- **The L2 cache barrier**: roofline-predicted 3.9× INT4 speedup collapses to 0.49× due to L2 residency of FP16 weights and dequantization-induced compute saturation of the INT4 kernel, validated by a weight-size sweep across the L2 boundary with NCU counter analysis. To our knowledge, this is the first characterization of this failure mode.
- **Selective INT4 quality**: reconstruction weights tolerate INT4 well (+0.051 PPL), establishing them as targets once kernel support matures.
- **NCU-level kernel analysis**: TMA vs. global-load root cause, corrected L1 metrics on Hopper, and the occupancy paradox (fewer warps, more bandwidth).

2 Experimental Setup

2.1 Hardware

All measurements on a single NVIDIA H100 80GB SXM5 (HBM3). Reference peaks: 3.35 TB/s HBM bandwidth, 989.4 TFLOPS BF16 tensor core, 132 SMs, 50 MB L2 cache. GPU clocks were not locked (`nvidia-smi -lgc`); Section 7 quantifies the resulting variance.

2.2 Software

SGLang (commit `7ef18be`), PyTorch 2.9.1+cu128, FlashInfer 0.6.6, Triton 3.5.1, CUDA 12.8, NCU 2025.1.1.

2.3 Model Configurations

Table 1: Attention configurations profiled.

Model	H_q	H_{kv}	d	Type	KV LoRA
Llama-3-8B	32	8	128	GQA 4:1	—
DS-V2-Lite	16	16	128	MLA	512

Model choice rationale. We use three models, each chosen for a specific role. **Llama-3-8B** (GQA) serves as the attention kernel benchmark: it is the most widely deployed open-weight model family and the standard target for FlashInfer/Triton comparisons, ensuring our methodology reproduces known results. **DeepSeek-V3 shapes** (H=128, d_{lora}=512) are used for the MLA reconstruction analysis, as DeepSeek-V3 is the only production-scale MLA model and defines the shapes relevant to real serving workloads. **DeepSeek-V2-Lite** (15.7B) is used for the perplexity evaluation, as the smallest publicly available MLA model that can run on a single GPU, allowing controlled quality measurement with identical architecture to V3 but tractable evaluation cost. **Qwen2.5-7B** provides end-to-end decode decomposition as a representative GQA serving workload.

2.4 Methodology

Timing. CUDA events with 10+ warmup iterations, median of 100+ timed iterations. Reproducibility verified across 3 independent trials (Section 7).

FLOP counting. Standard attention: $4 \cdot B \cdot \frac{S^2}{2} \cdot H \cdot d$ ($QK^\top$ matmul + AV matmul, $\times 2$ for multiply-accumulate, $\div 2$ for causal masking).

Bandwidth formula. Decode: BW $= (\text{KV}_{\text{read}} + Q_{\text{read}} + O_{\text{write}})/t_{\text{median}}$, where KV accounts for 99.8% of bytes.

NCU profiling. `ncu --set full --profile-from-start no` with kernel replay. Metrics collected: `dram__throughput`, `sm__throughput`, `sm__warps_active`, L1/L2 sector hit/miss counts, `launch__registers_per_thread`, `smsp__inst_executed_pipe_tensor`. NCU profiling overhead is excluded from bandwidth computations (separate timing runs).

3 Attention Kernel Validation

Before presenting our MLA findings, we validate the profiling methodology by comparing FlashInfer and Triton attention kernels on Llama-3-8B, a well-studied GQA configuration. This reproduces known performance gaps and establishes the measurement infrastructure used in subsequent sections.

3.1 Decode: Bandwidth Scaling

Table 2: Decode performance, Llama-3-8B GQA, $L_{kv} = 2048$.

BS	FI (ms)	Tri (ms)	FI BW	Tri BW	Speedup
1	0.028	0.057	298	147	2.03×
4	0.032	0.058	1056	582	1.82×
16	0.058	0.070	2317	1910	1.21×
64	0.187	0.218	2870	2470	1.16×
128	0.364	0.416	2957	2585	1.14×
256	0.720	0.806	2987	2669	1.12×

Both backends are firmly memory-bound. FlashInfer peaks at 2,987 GB/s (89% of 3.35 TB/s) while Triton peaks at 2,669 GB/s (80%). The gap narrows from 2× at bs=1 (launch-overhead dominated) to 1.12× at bs=256 (bandwidth-saturated).

FlashInfer approaches 3,000 GB/s at $L_{kv} = 8192$, saturating HBM at 89.6% of peak.

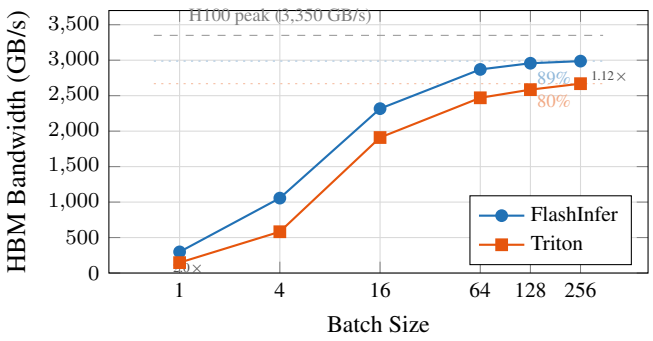

Figure 1: Decode bandwidth vs. batch size (Llama-3-8B, L_{kv}=2048). Both backends approach HBM saturation; FlashInfer peaks at 89% of H100 peak. The gap narrows from 2× at bs=1 to 1.12× at bs=256 as launch overhead becomes negligible relative to streaming KV reads.

Table 3: Decode performance, Llama-3-8B GQA, bs=64, varying L_{kv}.

L_{kv}	FI (ms)	Tri (ms)	FI BW	Tri BW	Speedup
256	0.037	0.065	1859	1053	1.76×
512	0.059	0.069	2273	1950	1.17×
1024	0.102	0.113	2648	2389	1.11×
2048	0.187	0.217	2875	2476	1.16×
4096	0.360	0.409	2984	2624	1.14×
8192	0.716	0.790	3000	2720	1.10×

3.2 Decode: NCU Analysis

The occupancy paradox. FlashInfer uses 183 registers/thread (2.4× Triton's 76), yielding only 12.2% active warps vs. 35.4%. Yet FlashInfer achieves 84% DRAM throughput vs. 76%. This shows that *memory access pattern quality dominates occupancy* for bandwidth-bound kernels: FlashInfer's fused design with coalesced, pipelined accesses extracts more bandwidth per warp than Triton's higher-occupancy two-phase approach.

Two-phase overhead. Triton splits attention into Stage 1 (KV scatter-gather, 217 μs) and Stage 2 (softmax reduction, 10 μs). Stage 2 adds 4.4% overhead but achieves 73.8% occupancy with only 30 registers, a well-optimized reduction kernel.

L2 irrelevance at scale. Both backends show <1% L2 hit rate. The KV cache at bs=64 is $64 \times 2048 \times 8 \times 128 \times 2 = 256$ MB, far exceeding the 50 MB L2. GQA's 4:1 head sharing provides no cache benefit at this scale.

3.3 Prefill: TFLOPS Scaling

FlashInfer peaks at **552 TFLOPS** (55.8% of 989.4 T peak). Triton peaks at 209 TFLOPS (21.1%). The 2.6× gap is consistent across all configurations and *widens* with sequence length, suggesting a fundamental rather than incidental difference.

Table 4: NCU metrics, decode, bs=64, $L_{kv} = 2048$.

Metric	FlashInfer	Triton (Stage 1)
Kernel	`BatchPrefill..`	`_fwd_grouped..`
DRAM util.	**84%**	76%
SM util.	17%	28%
Regs/thread	**183**	76
Active warps	12.2%	35.4%
Tensor insts	2.16M	2.10M
L2 hit rate	0.4%	1.0%
Duration	191 μs	217 μs

Table 5: Prefill performance, Llama-3-8B GQA. TFLOPS uses corrected formula ($4BS^2Hd/2$).

BS	S	FI (ms)	Tri (ms)	FI TF	Speedup
1	512	0.024	0.040	88	1.63×
1	1024	0.033	0.069	264	2.12×
1	2048	0.089	0.214	387	2.41×
1	4096	0.283	0.730	485	2.58×
4	2048	0.307	0.753	448	2.45×
4	4096	0.995	2.698	**552**	2.71×
16	2048	1.238	2.822	444	2.28×
16	4096	4.299	10.531	512	2.45×

3.4 Prefill: NCU Root Cause Analysis

3.4.1 Finding: TMA vs. Global Loads (Not "Cache Hit Rate")

A naive reading of NCU's L1 sector counters yields: FlashInfer 97% L1 hit rate, Triton 0.1%. **This is misleading.**

FlashInfer's Hopper kernel uses TMA (Tensor Memory Accelerator) [9], a dedicated hardware unit that performs asynchronous bulk tensor copies directly from HBM/L2 into shared memory, *bypassing the L1 global load data path entirely*. The 108K L1 global load sectors are residual metadata accesses (indptr arrays, scalar parameters), not QKV tensor data. The 37.8M sectors in Triton represent the *actual* QKV working set flowing through L1 via standard `ld.global` instructions.

The correct comparison is at L2: FlashInfer achieves 86.4% hit rate vs. Triton's 72.0%. FlashInfer's cooperative grid launch (132 blocks = 1 per SM) creates a controlled L2 working set; Triton's 2048-block grid generates 83% more L2 misses (9.9M vs. 5.4M sectors) due to wave-quantization thrashing.

This distinction is architecturally fundamental. TMA access is not "better caching" but a *different hardware data path* unavailable to Triton's compiler, which generates standard `ld.global` PTX.

3.4.2 Finding: No Register Spills (Occupancy-Only Bottleneck)

Triton's prefill kernel compiles to 255 registers, the `sm_90` hardware maximum. A natural hypothesis is that registers spill to local memory, adding latency. We tested this at three levels:

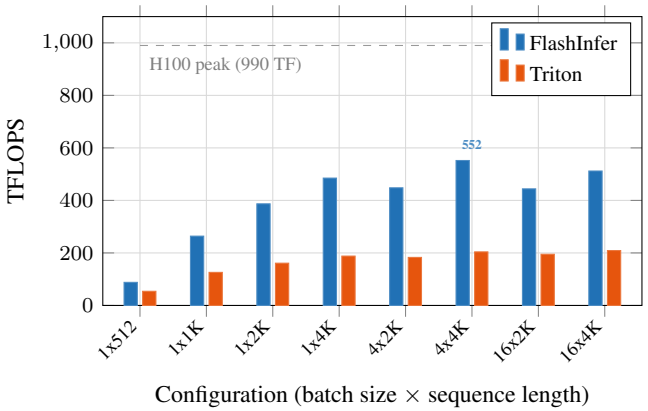

Figure 2: Prefill compute throughput (Llama-3-8B GQA). FlashInfer achieves 2.1–2.7× higher TFLOPS across all configurations, peaking at 552 TFLOPS (56% of peak). The gap widens with sequence length due to FlashInfer's TMA hardware loads and cooperative grid launch (Section 3.4).

Table 6: NCU metrics, prefill, bs=4, $S = 2048$.

Metric	FlashInfer	Triton
Kernel	`PrefillWith...`	`_fwd_kernel`
SM util.	**52.0%**	32.7%
DRAM util.	13.1%	9.3%
Regs/thread	168	**255** (max)
Active warps	14.1%	12.5%
Grid size	132 (cooperative)	2048
L1 global ld sectors	108K	37.8M
L2 hit rate	**86.4%**	72.0%
Tensor insts	2.23M	2.23M
LOCAL mem (CUBIN)	**0 bytes**	**0 bytes**
Duration	368 μs	909 μs

1. **NCU local memory counters**: The local memory wavefront counters (`l1tex..mem_local_op_{ld,st}.sum`) returned `n/a` on all Hopper kernels (known NCU limitation on sm_90).

2. **PTX inspection**: Zero `.local` directives, zero `st.local` / `ld.local` instructions in the compiled PTX.

3. **CUBIN resource dump** (`cuobjdump --dump-resource-usage`): `LOCAL:0` for `_fwd_kernel`.

Verdict: Triton's 255-register kernel does *not* spill. The compiler fits all live variables within the register file at the cost of maximal register allocation. The performance impact is purely occupancy: fewer warps per SM means fewer opportunities to hide memory latency.

Reducing Triton's register count (e.g., via different tiling or explicit `num_stages` tuning) could improve occupancy and partially close the prefill gap, but the TMA access pattern difference (Section 3.4.1) would remain.

3.4.3 Finding: Cooperative Grid vs. Wave Quantization

FlashInfer launches exactly 132 thread blocks (one per SM) using CUDA cooperative launch semantics. This eliminates wave-quantization effects and enables persistent-kernel-style execution where each block processes multiple tiles sequentially with full shared memory and register file utilization.

Triton launches 2,048 blocks (15.5× more), dispatched in multiple waves. Each wave evicts the previous wave's L2 residency, explaining the 14-point L2 hit rate gap (86.4% vs. 72.0%).

4 The MLA Reconstruction Bottleneck

Having established confidence in our profiling infrastructure, we turn to our central question: how much of MLA's attention-layer time is spent on reconstruction?

Multi-head Latent Attention [4] compresses KV cache to a low-rank latent $c_t \in \mathbb{R}^{d_{\text{lora}}}$ with $d_{\text{lora}} = 512$. During decode, full-dimensional K and V are *reconstructed* via weight-absorbed batch matrix multiplications (BMMs):

$$\text{BMM1: } Q'_h = Q_h W_h^{\text{kc}}, \quad W_h^{\text{kc}} \in \mathbb{R}^{d_{\text{nope}} \times d_{\text{lora}}}$$
$$\text{BMM2: } O_h = A_h W_h^{\text{vc}}, \quad W_h^{\text{vc}} \in \mathbb{R}^{d_{\text{lora}} \times d_v}$$

where $h = 1, \ldots, H$ heads. These are batched GEMMs with batch dimension H, independent of KV sequence length.

4.1 Cost Breakdown: DeepSeek-V3 Architecture

We profile reconstruction BMMs separately from FlashInfer MLA attention on DeepSeek-V3-scale shapes ($H = 128$, $d_{\text{nope}} = 128$, $d_{\text{lora}} = 512$, $d_v = 128$).

Table 7: MLA reconstruction overhead per layer, DeepSeek-V3 shapes, $L_{kv} = 2048$. Reconstruction cost is independent of KV length.

BS	Recon (μs)	Attn (μs)	Recon %	Model (ms)
1	35.6	23.0	**60.7%**	2.17
4	35.4	27.2	56.6%	2.16
16	34.2	40.4	45.8%	2.08
64	34.1	83.6	29.0%	2.08
128	38.4	154.4	19.9%	2.34
256	66.9	293.6	18.6%	4.08
512	109.4	584.5	15.8%	6.68

Key finding: At batch size 1, reconstruction BMMs consume **60.7%** of attention-layer time (35.6 μs vs. 23.0 μs for the attention kernel itself). Across 61 layers, this totals **2.17 ms per token**, a fixed per-query overhead invisible to standard attention benchmarks.

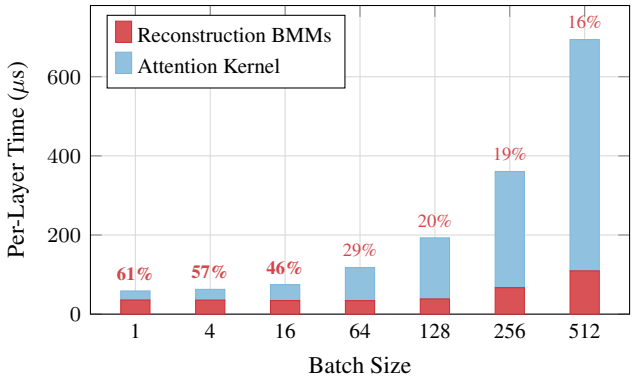

Figure 3: MLA attention-layer time decomposition per layer (DeepSeek-V3 shapes, L_{kv}=2048). Reconstruction BMMs (red) dominate at small batch sizes (**61% at bs=1**) and remain significant at bs=512. Reconstruction cost is nearly constant (∼35 μs), independent of KV length; attention scales linearly.

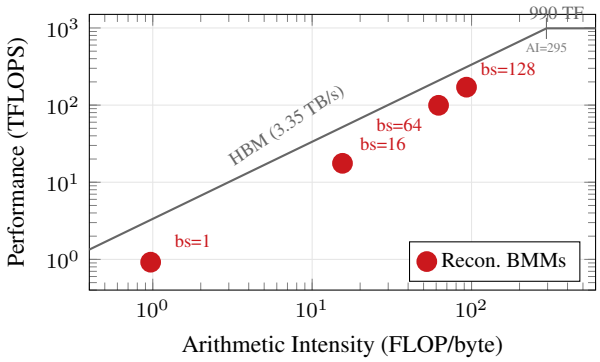

Figure 4: H100 roofline with MLA reconstruction BMMs. All operating points fall on the memory-bound slope, well below the compute ceiling (crossover at AI=295). Even at bs=128 (AI=93), reconstruction achieves only 55% of the memory ceiling.

4.2 Roofline Analysis: Memory-Bound at All Batch Sizes

Table 8: Arithmetic intensity and achieved bandwidth for reconstruction BMMs. H100 roofline crossover: AI = 295 (990 TFLOPS / 3.35 TB/s).

BS	AI (BMM1)	BW_1	AI (BMM2)	BW_2
1	0.97	954	0.97	952
16	15.5	1133	15.5	1137
64	62.1	1592	62.1	1607
128	93.0	1837	93.0	2114
256	93.0	1741	93.0	1770

Arithmetic intensity peaks at 93 (bs=128+), well below the crossover point of 295. **All reconstruction BMMs are memory-bound.** Achieved bandwidth ranges from 952 GB/s to 2,114 GB/s (28–63% of HBM peak), leaving significant room for kernel optimization.

Critical implication: Because reconstruction is memory-bound and weight-dominated (weight matrix is $128 \times 128 \times 512 \times 2 = 16$ MB per BMM, fixed regardless of batch size), reducing weight precision directly reduces the memory bandwidth bottleneck.

5 INT4 Mitigation and the L2 Cache Barrier

FP16 reconstruction BMMs are memory-bound at all practical batch sizes, and their weight matrices dominate data transfer. Reducing weight precision should therefore yield proportional speedups, assuming the kernel remains memory-bound after quantization. We evaluate INT4 weight-only quantization of W^{kc} and W^{vc}, addressing two questions: does it preserve quality, and does it deliver the expected performance gain?

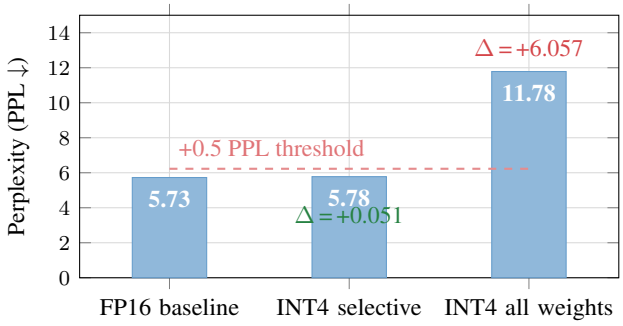

Figure 5: Wikitext-2 perplexity under INT4 quantization (DeepSeek-V2-Lite). Selective INT4 of reconstruction weights (`kv_b_proj`) adds only +0.051 PPL, well within the +0.5 quality threshold. Naive INT4 of all linear weights severely degrades quality.

5.1 Perplexity Evaluation

We evaluate on DeepSeek-V2-Lite (15.7B parameters) using wikitext-2 test set (308K tokens, stride-512 sliding window, max length 2048). Three configurations:

Table 9: Wikitext-2 perplexity under INT4 quantization. `kv_b_proj` is the reconstruction weight (W^{kc}/W^{vc} combined).

Configuration	PPL	Δ	Weights quantized
FP16 baseline	5.727	—	0
INT4 selective	5.777	**+0.051**	27 (kv_b_proj)
INT4 all linear	11.784	+6.057	5,209

Selective INT4 quantization of reconstruction weights adds only +0.051 perplexity, an order of magnitude below the typical 0.5 quality threshold. Naive INT4 of all linear weights, by contrast, more than doubles perplexity.

5.2 Why Reconstruction Weights Are Quantization-Friendly

The W^{kc} and W^{vc} matrices are *projection weights* that map between a compressed 512-dimensional latent space and 128-dimensional head spaces. Several factors explain their quantization robustness:

1. **Low-rank structure**: The latent space was trained to be compressible; the projection matrices inherit smooth spectral properties.
2. **Redundancy**: With $H = 128$ heads each projecting from the same 512-dim latent, per-head reconstruction errors are averaged across heads before affecting output.
3. **Post-softmax attenuation**: BMM2 operates on attention-weighted values; quantization noise is attenuated by the softmax distribution's sparsity.

5.3 Theoretical Speedup

For memory-bound operations, speedup from weight quantization equals the ratio of total data transferred:

$$\text{Speedup} = \frac{\text{Weight}_{\text{FP16}} + \text{Activation}_{\text{FP16}}}{\text{Weight}_{\text{INT4}} + \text{Scale/ZP} + \text{Activation}_{\text{FP16}}} \quad (1)$$

Table 10: Theoretical INT4 speedup for reconstruction BMMs. Weight bytes dominate at small batch sizes.

BS	Wt (MB)	Act (MB)	INT4 eq. (MB)	Speedup
1	16.00	0.03	4.00	3.94×
4	16.00	0.13	4.00	3.89×
16	16.00	0.50	4.00	3.67×
64	16.00	2.00	4.00	3.00×
128	16.00	4.00	4.00	2.50×
256	16.00	8.00	4.00	2.00×

At batch size 1 (latency-critical single-query serving), the roofline predicts **3.94× speedup**, which would reduce the 2.17 ms full-model reconstruction overhead to ∼0.55 ms.

5.4 Measured Kernel Performance

We implemented a custom batched W4A16 Triton kernel that fuses the head dimension into the grid, avoiding 128 separate kernel launches. The kernel dequantizes INT4 weights to FP16 in registers and uses tensor core `tl.dot` for the matmul.

The INT4 kernel is 2× slower than cuBLAS FP16, not 3.9× faster. It is, however, 30× faster than a per-head FP16 loop (0.073 ms vs. 2.19 ms), confirming that the batched grid approach itself is effective. The bottleneck is competing with cuBLAS, not the batching strategy. Preliminary NCU profiling indicates the INT4 kernel is compute-bound rather than memory-bound, suggesting dequantization overhead may offset the memory savings; we analyze this in detail in Section 5.5.

Table 11: Measured INT4 kernel performance vs. cuBLAS FP16 `torch.bmm`. "FP16 loop" = 128 individual `torch.mm` calls.

BS	FP16 bmm (ms)	INT4 Tri (ms)	FP16 loop (ms)	Ratio INT4/FP16
1	0.036	0.073	2.194	0.49×
4	0.037	0.073	2.203	0.50×
16	0.036	0.082	2.187	0.44×
64	0.036	0.129	2.215	0.28×
128	0.040	0.187	2.211	0.21×
256	0.070	0.302	2.223	0.23×

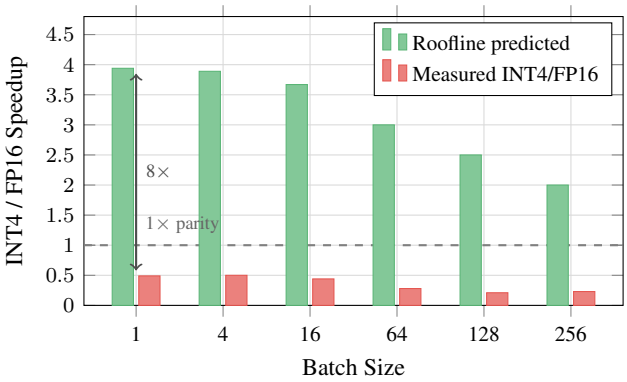

Figure 6: The L2 cache barrier. Roofline analysis predicts 2–3.9× INT4 speedup assuming HBM-bound operation. Measured performance shows 0.2–0.5×: the INT4 kernel is *slower* than cuBLAS FP16. The 16 MB reconstruction weight fits in H100's 50 MB L2, and INT4 dequantization shifts the kernel to compute-bound, doubly invalidating the roofline prediction.

5.5 The L2 Cache Barrier: Why Roofline Fails for Small-Matrix MLA Reconstruction

The 8× gap between roofline-predicted (3.9×) and measured (0.49×) performance has two primary contributing factors and one amplifier. The roofline fails twice: it assumes data is served from HBM (it is not), and it assumes the INT4 kernel remains memory-bound (it does not).

(1) L2 cache residency invalidates the HBM-bound assumption. The standard roofline model [12] assumes data is served from HBM at 3.35 TB/s. But the total reconstruction weight per BMM is $128 \times 128 \times 512 \times 2 = 16$ MB, comfortably within H100's 50 MB L2 cache. After first access, `torch.bmm` serves weights from L2 at multi-terabyte-per-second effective bandwidth (≈5–12 TB/s depending on access pattern), far exceeding HBM's 3.35 TB/s. Reducing weight precision from FP16 to INT4 saves HBM bandwidth that *was never the bottleneck*.

We validate this directly by scaling the weight matrix across the L2 boundary. Holding H=128 and d_{nope}=128 fixed, we sweep d_{lora} from 256 to 4096, producing FP16 weight sizes from 8 MB to 128 MB. Figure 7 shows the result: the INT4/FP16 time ratio drops from 1.91× at 8 MB to 1.08× at 128 MB, with a clear knee at 40–48 MB as weights begin to exceed L2 capacity.

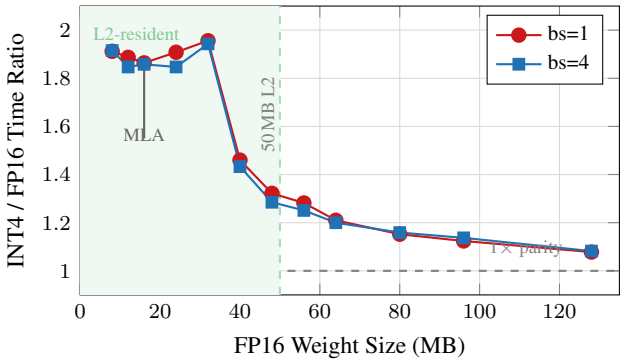

Figure 7: Weight-size scaling across the L2 boundary. INT4/FP16 time ratio vs. FP16 weight size. Below 50 MB (green, L2-resident), INT4 is ∼1.9× slower. Above 50 MB, the ratio drops sharply toward parity as FP16 falls back to HBM. At MLA's 16 MB operating point (arrow), FP16 benefits fully from L2 residency.

At MLA's operating point (16 MB), INT4 is 1.86× slower; at 128 MB (well past L2), the gap nearly closes.

NCU profiling across the same sweep confirms the mechanism. The FP16 cuBLAS kernel (`nvjet`, TMA-based) is DRAM-bound: DRAM utilization scales from 35% at 8 MB to 83% at 128 MB, while SM utilization stays below 15%. The INT4 Triton kernel is the opposite: SM utilization scales from 33% to 79% (dequantization dominates), while DRAM utilization stays below 23%. INT4 reads exactly 4× fewer DRAM bytes as expected, but converting that bandwidth savings into latency reduction requires the kernel to not be compute-bound, which it is.

This creates a paradox specific to MLA: the same low-rank compression that makes reconstruction weights small enough to be a latency concern *also* makes them small enough to be L2-resident, defeating the primary motivation for weight quantization. Standard LLM linear layers (e.g., FFN projections with weights in the hundreds of MB) do not exhibit this behavior.

(2) INT4 dequantization shifts the bottleneck from memory to compute. The NCU data shows that the INT4 kernel is SM-bound (up to 79% SM utilization) rather than DRAM-bound. Bit masking, shifting, signed extension, and type conversion on every packed byte consume the freed bandwidth headroom. Even when weights exceed L2 and FP16 falls back to HBM, INT4 cannot fully exploit its 4× data reduction because dequantization saturates the SMs first.

(3) cuBLAS dispatch advantage and INT4 cache sensitivity. `torch.bmm` dispatches via the `nvjet` kernel family with TMA hardware loads (168 registers, L1 bypassed entirely), while our Triton kernel uses standard `ld.global` with 57 registers. Under concurrent L2 contention, INT4 degrades 3.7× vs. only 1.7× for FP16, indicating that dequantization and irregular packed-byte access patterns increase the INT4 kernel's reliance on cache locality. This asymmetric sensitivity further compounds factors (1) and (2).

When does INT4 help? The L2 cache barrier is not absolute.

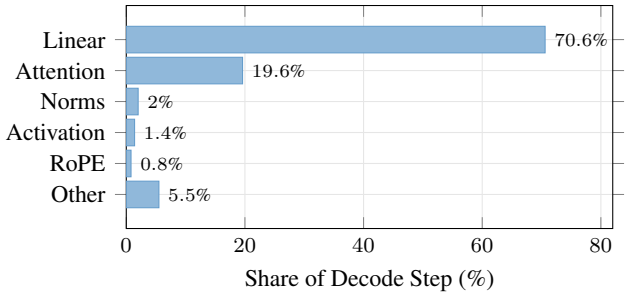

Figure 8: Decode step decomposition (Qwen2.5-7B, bs=64). Linear layers dominate at 71%; attention is ∼20%. For MLA, reconstruction BMMs add to this: at bs=1, reconstruction (2.17 ms) exceeds attention (1.40 ms).

In production serving, concurrent operations (FFN GEMMs, attention across multiple layers, multi-tenant request batching) contend for L2 capacity and may evict reconstruction weights back to HBM. INT4 reconstruction gains are therefore **deployment-dependent**: negligible in isolated benchmarks, but potentially significant under L2 pressure from high-throughput serving. Fusing reconstruction across multiple layers to exceed L2 capacity, or deploying on GPUs with smaller L2 caches, could also restore the predicted benefit.

6 End-to-End Context

To quantify the system-level impact of reconstruction, we contrast MLA against a standard GQA decode workload.

Table 12: Decode step decomposition, Qwen2.5-7B, bs=64.

Component	Time (μs)	Share
Linear layers (GEMMs)	5,079	70.6%
Attention (FlashInfer)	**1,413**	**19.6%**
Norms (RMSNorm)	145	2.0%
Activation (SiLU)	98	1.4%
RoPE	57	0.8%
Other	399	5.5%

Attention constitutes ∼20% of decode time for GQA models. For MLA models like DeepSeek-V3, reconstruction overhead adds a further component: at bs=1, reconstruction GEMMs (2.17 ms across 61 layers) exceed the attention kernel cost (1.40 ms), making reconstruction the dominant attention-layer bottleneck. Targeted optimization of reconstruction, whether through INT4 quantization or kernel fusion, is therefore higher-impact than further attention kernel tuning.

7 Reproducibility

At bs≥64, results are deterministic (<0.1% CV). At bs=1, ∼3% variance arises from GPU boost clock fluctuations on short (∼28 μs) kernels. Relative speedup ratios are stable across all trials.

Table 13: Reproducibility across 3 independent trials (warmup=20, iters=200).

Config	T1	T2	T3	CV
FI dec bs=1	0.029	0.030	0.030	3.4%
Tri dec bs=1	0.056	0.059	0.057	2.6%
FI dec bs=64	0.187	0.187	0.187	<0.1%
Tri dec bs=64	0.216	0.216	0.216	<0.1%
FI dec bs=256	0.719	0.718	0.718	<0.1%
Tri dec bs=256	0.813	0.813	0.813	<0.1%

8 Discussion

Our analysis reveals a chain of findings: MLA's KV compression shifts the attention-layer bottleneck to reconstruction GEMMs (61% at bs=1); these are memory-bound at all practical batch sizes; INT4 quantization preserves quality but fails to deliver speedup due to L2 cache residency of FP16 weights and dequantization-induced compute saturation of the INT4 kernel. We now discuss the implications.

8.1 Implications for Compiler-Generated Attention

The prefill gap is fundamental to the compiler architecture: Triton's MLIR→PTX pipeline generates `ld.global` instructions because it has no TMA code generation path for attention patterns. Triton 3.x introduced `tl.experimental_descriptor_load` for TMA access, but this requires manual descriptor setup that attention kernels have not adopted. Closing the prefill gap requires either:

1. Triton compiler support for automatic TMA lowering of attention-shaped loads (research problem).

2. Manual TMA integration in the Triton kernel source (engineering, loses portability).

3. A higher-level DSL that targets both TMA and global load paths depending on the GPU architecture.

The decode gap (1.1×) is more tractable: Triton's two-phase split could be fused, and register tuning (`num_warps`, tiling factors) could recover some bandwidth.

8.2 The Occupancy Paradox as Design Principle

FlashInfer's decode kernel shows that for memory-bound workloads, *maximizing bandwidth per warp* outperforms *maximizing warp count*. At 183 registers, FlashInfer has 2.4× fewer active warps than Triton, yet extracts 10% more HBM bandwidth. This is consistent with the principle that memory-bound kernels benefit more from coalesced access patterns and prefetching than from raw occupancy [11].

8.3 MLA: Compression Shifts the Bottleneck

MLA's 7.1× KV traffic reduction makes the attention kernel faster but creates a new bottleneck in the reconstruction GEMMs that were previously negligible. At bs=1, reconstruction costs 61% of attention-layer time. This illustrates a general systems principle: optimizations that reduce data movement can shift workloads into regimes where cache hierarchy, rather than raw bandwidth, determines performance. The implication for MLA specifically is that reconstruction overhead must be co-optimized with attention, not ignored.

8.4 Selective Quantization: Quality Result

Despite the kernel limitations, reconstruction weights tolerate quantization well: +0.051 PPL for selective INT4 vs. +6.057 for naive INT4 of all weights. For future MLA serving systems, **reconstruction weights are the natural first target for aggressive quantization**, pending kernel support.

8.5 Limitations

- Single GPU; multi-GPU tensor-parallel workloads may show different ratios due to all-reduce overlap.
- GPU clocks not locked; absolute numbers carry ∼3% uncertainty at small batch sizes.
- NCU local memory counters unavailable on Hopper (`n/a`); spill analysis relies on PTX/CUBIN inspection.
- INT4 kernel benchmarked in Triton only; a CUDA-native implementation with INT8 tensor cores may achieve different results.
- Perplexity evaluated on DeepSeek-V2-Lite (15.7B); larger models may show different quantization sensitivity.
- One model architecture per attention type.

9 Related Work

Attention kernels. FlashAttention [3] and FlashAttention-2 [2] introduced tiling-based exact attention with $O(N)$ memory. FlashInfer [13] extends this with Hopper-native TMA support. Triton [10] enables Python-level GPU kernel development. SGLang [14] integrates both backends for LLM serving.

MLA architectures. DeepSeek-V2 [4] introduced Multi-head Latent Attention; DeepSeek-V3 [5] scaled it to 671B parameters with 128 attention heads. While both papers describe the weight-absorbed reconstruction mechanism, neither quantifies its runtime cost relative to the attention kernel.

Performance modeling. The roofline model [12] is the standard framework for reasoning about compute- vs. memory-bound kernels. Our L2 cache barrier finding (Section 5.5) exposes a dual failure mode of single-level roofline analysis:

when working sets fit in L2 the effective bandwidth ceiling shifts from HBM to L2, and when dequantization makes the kernel compute-bound the memory-bound assumption itself breaks. Volkov [11] showed that occupancy is not always the primary performance factor, motivating our analysis of FlashInfer's register-heavy approach.

Weight quantization. GPTQ [6] and AWQ [8] enable post-training INT4 weight quantization for LLM inference. KVQuant [7] applies quantization to the KV cache itself, targeting the memory footprint that MLA addresses architecturally. QuaRot [1] uses rotation to eliminate outliers before 4-bit quantization. All prior quantization work targets standard linear layers (FFN, attention projections), not the reconstruction matrices specific to MLA. To our knowledge, no prior public work quantifies the reconstruction GEMM overhead in MLA or evaluates selective INT4 quantization of these weights.

10 Conclusion

We presented two sets of findings. First, a kernel-level characterization of FlashInfer vs. Triton attention on H100, identifying TMA hardware loads, memory access pattern quality, and cooperative grid launch as the three root causes of the performance gap. Second, we identified MLA's reconstruction GEMMs as a dominant and previously unquantified bottleneck, consuming 61% of attention-layer time at batch size 1 for DeepSeek-V3-scale architectures.

The INT4 quantization results illustrate a subtlety in MLA optimization. **Quality is safe**: selective INT4 adds only +0.051 perplexity. **But speedup is not automatic**: our custom Triton W4A16 kernel achieves 0.49× of cuBLAS FP16 due to two interacting factors: the 16 MB reconstruction weights fit in H100's 50 MB L2 cache (making HBM bandwidth savings irrelevant), and INT4 dequantization shifts the kernel from memory-bound to compute-bound (consuming the freed bandwidth headroom). The gap between roofline-predicted (3.9×) and realized performance is specific to small, L2-resident weight matrices, a regime common in MLA but absent from standard LLM linear layers. Closing this gap requires either CUDA-native INT8 tensor core kernels that avoid the dequantization bottleneck, cross-layer weight fusion beyond L2 capacity, or deployment on GPUs with smaller L2 caches.

All profiling scripts, raw data, and the perplexity evaluation code are publicly available for reproduction.

References

[1] Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024.

[2] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *ICLR*, 2024.

[3] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022.

[4] DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[5] DeepSeek-AI. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[6] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *ICLR*, 2023.

[7] Coleman Hooper, Sehoon Kim, Hiva Mohammadi, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*, 2024.

[8] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. In *MLSys*, 2024.

[9] NVIDIA. Nvidia h100 tensor core gpu architecture. `https://resources.nvidia.com/en-us-tensor-core`, 2022.

[10] Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *MAPL*, 2019.

[11] Vasily Volkov. Better performance at lower occupancy. In *GPU Technology Conference*, 2010.

[12] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

[13] Zihao Ye, Lequn Lai, Ruihang Shao, Wuwei Zheng, and Tianqi Chen. Flashinfer: Efficient and customizable attention engine for llm inference serving. In *MLSys*, 2025.

[14] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Kober, Yineng Shi, et al. Sglang: Efficient execution of structured language model programs. In *NeurIPS*, 2024.