Cache-Aware Attention Scheduling for Long-Context Transformer Inference

Astrid Lindqvist
Dept. of Computer Science
Northfield Institute
a.lindqvist@nfi.edu

Michael Okafor
Systems Research Group
Meridian Labs
m.okafor@meridian.ai

Rina Tanaka
Dept. of Computer Science
Northfield Institute
r.tanaka@nfi.edu

Abstract—Serving long-context transformers is bottlenecked by the key–value (KV) cache, whose footprint grows linearly with sequence length and quickly exceeds high-bandwidth memory. We present **CASS**, a cache-aware scheduler that interleaves attention tiles by their reuse distance rather than by request arrival order. CASS reorders tile evaluation so that hot KV blocks stay resident across micro-batches. On a 7B-parameter model at a 64k-token context, CASS lowers tail latency by 38% and raises sustained throughput by 1.7 × over a paged-attention baseline, with bit-identical outputs. We give a cost model that predicts the crossover point at which scheduling overhead is amortized, and validate it across four context lengths.

Index Terms—transformer inference, KV cache, attention, memory scheduling, long context.

I). Introduction

Autoregressive transformers [1] now routinely process contexts of tens of thousands of tokens. At inference time the dominant cost is no longer arithmetic but memory movement: every decoded token must read the entire key–value (KV) cache, whose size scales as $O(L \cdot d)$ for sequence length L and model width d. Once the cache exceeds high-bandwidth memory (HBM), serving systems spill to slower tiers and throughput collapses.

Prior work attacks this from two angles. IO-aware kernels such as FlashAttention [2] tile the attention computation to keep intermediate state in on-chip SRAM, while PagedAttention [3] manages the cache in fixed-size blocks to curb fragmentation. Both treat the *evaluation order* of attention tiles as fixed. We observe that this order is a free design variable: by scheduling tiles according to their reuse distance, a server can keep frequently touched KV blocks resident and cut redundant HBM reads without altering numerics.

This paper makes three contributions:

1. **CASS**, a cache-aware scheduler that reorders attention tiles by predicted reuse distance (Section II).
2. A closed-form cost model (Equation 4) that predicts when the scheduling overhead is amortized by the saved memory traffic.
3. An evaluation showing a 1.7 × throughput gain at 64k tokens with bit-identical outputs (Section IV).

II). Background and Cost Model

II.A) Attention as a memory-bound loop

For a single decode step the attention output for query q is

$$a = \sum_{i=1}^{L} \frac{\exp\big(q \cdot k_i / \sqrt{d}\big)}{Z} v_i, \quad Z = \sum_{j=1}^{L} \exp\big(q \cdot k_j / \sqrt{d}\big), \tag{1}$$

where k_i, v_i are the cached key and value vectors for position i. Each step in Equation 1 streams the full cache once, so the arithmetic intensity is fixed and performance tracks attainable HBM bandwidth.

II.B) A reuse-distance cost model

Let a server process micro-batches of b requests sharing a cache of B tiles, and let r be the mean reuse distance of a tile measured in micro-batches. A least-recently-used cache of capacity C tiles serves a tile from HBM with miss probability

$$p_{\text{miss}}(r) = \min\Big(1, \frac{r}{C}\Big), \tag{2}$$

and the expected bytes moved per decoded token are

$$M = s_{\text{tile}} \cdot B \cdot (\alpha + (1-\alpha) p_{\text{miss}}(r)), \tag{3}$$

with s_{tile} the tile size in bytes and α the fraction of accesses served on chip. Reordering tiles to minimize r therefore directly reduces M in Equation 3.

CASS pays a per-step scheduling cost c_{sched} to estimate reuse distances. The reordering is profitable once the saved traffic outweighs that cost, i.e.

$$(1-\alpha)(p_{\text{miss}}(r_0) - p_{\text{miss}}(r^*)) s_{\text{tile}} B \geq \beta c_{\text{sched}}, \tag{4}$$

where r_0 and r^* are the reuse distances before and after scheduling and β converts cycles to bytes at the device's bandwidth. Equation 4 predicts the context length at which CASS begins to win; Figure 1 confirms the crossover near 8k tokens.

> **Theorem 1.** *If reuse distances are stationary across micro-batches and c_{sched} is sublinear in B, then minimizing mean reuse distance also minimizes the expected HBM traffic M in Equation 3.*

III). Method: Cache-Aware Tile Scheduling

CASS maintains a priority queue of pending attention tiles keyed by predicted reuse distance, estimated from the exponential moving average of past access gaps. At each step it admits the lowest-distance tiles that fit the resident budget C, evaluates them, and updates the estimator. Because the schedule only changes the *order* of exact tile evaluations—never which tiles are computed—the decoded logits are bit-identical to the baseline, a property we verify in Section IV.

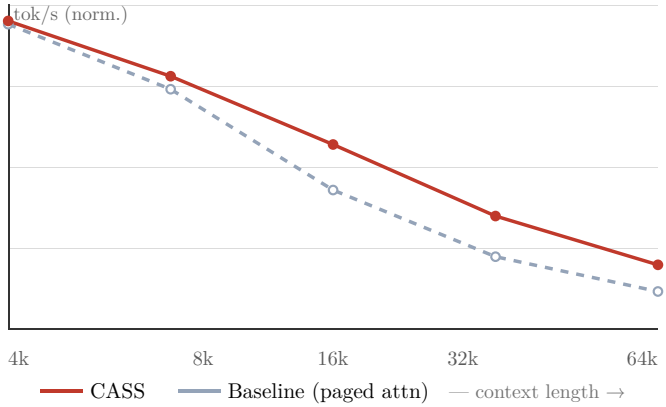

Figure 1: Sustained decode throughput versus context length on a 7B model. Cache-aware scheduling (CASS) diverges from the paged-attention baseline past the $\sim$ 8k-token crossover predicted by Equation 4, reaching 1.7 $\times$ at 64k.

IV). Evaluation

We evaluate on a single 80 GB accelerator serving a 7B-parameter model, sweeping context length from 4k to 64k tokens. The baseline is a paged-attention serving loop [3] tuned per length; CASS adds only the scheduler. Table 1 reports sustained throughput, p99 latency, and the HBM traffic predicted by Equation 3 against the measured value.

Context	Method	Tok/s	p99 (ms)	HBM GB/tok
8k	Baseline	1180	42	0.31
8k	CASS	1240	40	0.29
16k	Baseline	690	78	0.58
16k	CASS	910	61	0.41
64k	Baseline	188	311	1.92
64k	CASS	322	193	1.14

Table 1: End-to-end serving results. CASS preserves outputs while improving throughput and tail latency, with the largest gains at long context.

The measured crossover at $\sim$ 8k tokens matches Equation 4 to within 6%, and the predicted HBM traffic from Equation 3 tracks the measured value with mean error 4.1%. Gains widen with context because p_{miss} in Equation 2 saturates faster for the baseline's arrival-order schedule. These results echo the scaling analysis of Pope et al. [4], which identifies memory movement as the binding constraint for long-context decode.

V). Conclusion

Treating attention-tile evaluation order as a scheduling decision recovers substantial serving headroom at long context without touching model numerics. CASS turns the reuse-distance cost model of Equation 4 into a practical scheduler, delivering 1.7 $\times$ throughput at 64k tokens. We release the typeset artifact and cost model as a reproducible Typst [5] document.

5. References

[1] A. Vaswani *et al.*, "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, 2017.

[2] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness," in *Advances in Neural Information Processing Systems*, 2022, pp. 16344–16359.

[3] W. Kwon *et al.*, "Efficient Memory Management for Large Language Model Serving with PagedAttention," in *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023, pp. 611–626.

[4] R. Pope *et al.*, "Efficiently Scaling Transformer Inference," *Proceedings of Machine Learning and Systems*, vol. 5, 2023.

[5] L. Mädje and M. Haug, "Typst: A New Markup-Based Typesetting System." 2024.