CSATTENTION: CENTROID-SCORING SPARSE ATTENTION FOR REUSABLE-PREFIX LONG-CONTEXT LLM SERVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context LLM serving increasingly relies on long, *reusable prefixes*, such as cached retrieval contexts, tool schemas, agent memory, and domain-specific system prompts. In this *write-once, read-many* regime, dense attention over the accumulated KV cache becomes a dominant decode-time bottleneck, while aggressive sparse attention often loses accuracy because online queries and cached keys follow different distributions. We propose *CSAttention*, a training-free sparse attention method designed for reusable-prefix long-context inference. CSAttention shifts expensive retrieval work from online decoding to an offline prefix-build stage: it clusters prefill queries in subspaces, precomputes centroid-to-key partial scores, and stores bounded Top-L lookup tables for each query centroid. At decode time, each new query selects its nearest query centroids, fetches the corresponding shortlists, aggregates centroid scores by key index, and performs attention only over the selected KV entries. This formulation does not remove the linear memory growth of long-context KV cache; instead, it creates a decode-stationary auxiliary index whose capacity is fixed after a reusable prefix is built and whose online work is regular and bounded by the shortlist budget. Extensive experiments on LongBench and LongBench-v2 show that CSAttention preserves near-full-attention accuracy at 95% sparsity across Llama, Qwen, and Mistral backbones. Compared with strong sparse-attention baselines, CSAttention achieves higher recall and downstream accuracy under matched sparse budgets, and delivers up to 4.24$\times$ decode-throughput speedup and up to 3.80$\times$ end-to-end speedup in long-prefix, moderate-to-long generation regimes, while making its TTFT, memory, and batch-scaling trade-offs explicit.

1 INTRODUCTION

Long-context serving often reuses retrieval contexts, system prompts, tool schemas, or agent memories across generations (Lu et al., 2024; Jin et al., 2024; Gao et al., 2024; Lee et al., 2025). Prefix caching materializes the KV cache once for many decode calls. This *write-once, read-many* regime can justify extra prefix-build work that reduces repeated long-KV reads.

Grouped-query and multi-query attention reduce KV heads (Shazeer, 2019; Ainslie et al., 2023), and PagedAttention improves cache allocation and sharing (Kwon et al., 2023). Neither identifies which historical positions a decode query can skip. Every generated token may still read long KV, so a reusable prefix lets us pay for a selector once and reuse it across generations.

Sparse attention touches fewer KV entries per token, but at high sparsity it must find the *right* ones. Heavy-hitter retention, partial KV fetching, and retrieval from key blocks, hashes, or compressed keys reduce work (Zhang et al., 2023; Ribar et al., 2024; Tang et al., 2024; Chen et al., 2024; Zhang et al., 2025a), yet selection errors can discard task evidence. Relevance is determined by queries, whose distribution can differ from cached keys. RetrievalAttention already builds a query-aware key index from prefill query–key relations (Liu et al., 2024). We ask whether a reusable prefix instead allows scoring work to be *compiled into a cacheable artifact*, making online selection regular and bounded.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

At 95% sparsity, a missed fact cannot be recovered by better attention over retained tokens. The selector must preserve rare, distant evidence without another full-context search.

CSAttention is a training-free answer. It clusters *prefill queries* in subspaces and precomputes Top-L lists of original key IDs and signed centroid-to-key partial scores. At decode time, each query routes to a centroid per subspace; fetched shortlists are reduced by key ID to select KV for ordinary attention. The full KV cache remains intact, while expensive centroid-to-key scoring is materialized with the reusable prefix rather than repeated online. A recent-token window and bounded insertion of generated keys protect recent history without changing the fixed query centroids.

Cache hits reuse an immutable prefix table while generated-key updates remain request-local. Cold construction increases time to first token (TTFT), so the target is prefix reuse or sufficiently long one-off generations.

At 5% active prefix KV, CSAttention matches Full Attention on LongBench-v2 (31.1 versus 31.0) and reaches 92.7% Top-32 recall on a fixed subset, versus 87.6% for a matched key-centroid control and 88.9% for RetrievalAttention. It retains near-Full quality across three LongBench backbones and scores 70.2 versus Full's 71.3 on RULER-128K. At 128K, all-GPU decode reaches 4.24$\times$ the same-backend Full baseline, versus Quest's 3.17$\times$ at the same active-KV budget; a 2048-token full request reaches 3.80$\times$ end-to-end. We also report index memory, TTFT, reuse break-even, and CPU–GPU costs.

Our contributions are: (i) reusable query-centroid score tables built over original KV; (ii) bounded signed subspace-score aggregation followed by exact attention over selected KV; and (iii) matched-budget quality and systems evaluation through 128K, including memory–latency–reuse boundaries.

2 Motivation and Problem Setting

A cached prefix of length N_{pre} can serve many generations. The resulting challenge is to spend prefix-build work on an artifact that makes high-sparsity decoding both accurate and cheap. Unlike a one-off prompt, each cache hit reuses the same historical keys and the same precomputed metadata. Thus, the relevant serving cost is not merely the first request's latency: construction can be amortized over repeated decode calls, while each call still receives its own evolving suffix.

2.1 High-sparsity Attention Requires High Recall

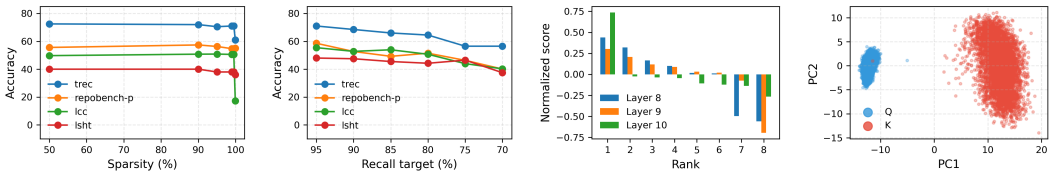
Long-context attention often concentrates on a few historical tokens. We specify the final active-KV budget as $K_{\text{sel}} = \lceil \rho N_{\text{pre}} \rceil$ relative to the cached prefix, even as generated history grows. Figure 1(a) shows that Llama-3.1-8B-Instruct remains accurate near 95% sparsity when the selected tokens match those with high attention. Figure 1(b) shows accuracy falling as Top-K recall drops. The central issue is therefore retrieval quality, not merely how few tokens are scored.

2.2 Why Key-centric Retrieval Becomes Fragile

Key-side indexes are natural for KV retrieval, but online relevance is determined by queries. At a small budget, key-side representatives can dilute rare important tokens or fail to align with the query distribution. Figure 1(d) illustrates this query–key distribution mismatch, motivating routing through query space. In particular, a representative that is useful for grouping keys need not preserve the ordering induced by an unseen query. At a tight Top-K budget, small ranking errors can remove the few positions carrying an answer even when most retrieved positions look semantically plausible.

2.3 From Query-centric Retrieval to Offline Centroid Scoring

RetrievalAttention uses prefill queries to build an attention-aware nearest-neighbor index over *keys* (Liu et al., 2024). CSAttention instead clusters prefill queries in subspaces and materializes centroid-to-key partial scores once. Decode routes to one centroid per subspace and reads bounded shortlists, avoiding repeated centroid-to-key scoring or online key-graph traversal. This is a distinct storage-for-computation choice, not the first query-informed retrieval method.



(a)Accuracy vs. sparsity. (b)Accuracy vs. recall. (c)Subspace contribution. (d)Q/K distribution shift.

Figure 1: **Motivation.** (a) Accuracy remains stable when true important tokens are retained at high sparsity. (b) Accuracy tracks Top-K recall. (c) Subspace contributions are uneven, motivating partition-and-aggregate scoring. (d) Query and key embeddings occupy different distributions, making purely key-centric retrieval fragile.

It aggregates signed evidence across m head subspaces. For query q_i and key k_j,

$$\langle q_i, k_j \rangle = \sum_{b=1}^{m} \langle q_i^{(b)}, k_j^{(b)} \rangle. \tag{1}$$

Figure 1(c) shows uneven subspace contributions. Uniform weights keep the online path regular; negative partial scores lower the reduce-by-key sum rather than being discarded. Centroid scores serve only retrieval; final attention uses original Q/K/V. The signed sum is important because evidence from one favorable subspace is not automatically sufficient: a key that also conflicts strongly in another subspace should lose rank. Conversely, a key supported by several selected lists accumulates their contributions before the final KV budget is imposed.

3 CSATTENTION

3.1 OVERVIEW AND PREFIX-CACHE LIFECYCLE

On a *cold prefix build*, CSAttention clusters the prefill *queries* and stores centroid-to-key score tables alongside the original KV cache; it neither clusters nor compresses K/V. A *prefix-cache hit* reuses the immutable prefix KV and table snapshot. Generated-suffix KV and table updates are request-local, so divergent continuations cannot alter one another's state. Promoting a continuation to a reusable prefix requires a corresponding new or updated artifact.

Centroids and list capacities are *decode-stationary*: maintenance can replace entries but cannot enlarge a list. With $L = \lceil \alpha N_{\text{pre}} \rceil$, storage still grows with prefix length. The matched-budget and 128K headline setting uses $C = 64$, $m = 8$, $\alpha = 0.10$, a final KV ratio $\rho = 0.05$, and a 32-token recent-history window; experiments identify any changes. This per-prefix storage and build cost is amortized over subsequent decoding.

3.2 OFFLINE PREFIX BUILD: QUERY-CENTROID SCORING

Consider one indexed KV-head group g in a layer. Let Q_g contain its assigned prefill query vectors and K_g, V_g its original KV tensors; under grouped-query attention, $M_g = |Q_g|$ need not equal the N_{pre} key positions. One table bank is stored per indexed KV-head group. Suppressing g, split Q and K into m corresponding subspaces of dimensions d_b, with $\sum_b d_b = d$, $Q^{(b)} \in \mathbb{R}^{M_g \times d_b}$, and $K^{(b)} \in \mathbb{R}^{N_{\text{pre}} \times d_b}$. Cosine k-means on the *prefill queries* yields $\mathcal{C}^{(b)} = \{c_1^{(b)}, \ldots, c_C^{(b)}\}$; normalization is used for clustering and routing, not native attention scoring.

For each centroid $c_j^{(b)}$, CSAttention computes signed partial scores against all keys in the same subspace:

$$s_{j,i}^{(b)} = \langle c_j^{(b)}, k_i^{(b)} \rangle, \qquad i \in \{1, \ldots, N_{\text{pre}}\}. \tag{2}$$

It then stores the Top-L key indices and their corresponding scores:

$$\mathcal{T}_j^{(b)} = \text{TopL}_{i=1}^{N_{\text{pre}}} \left\{ \left(i, s_{j,i}^{(b)} \right) \right\}, \tag{3}$$

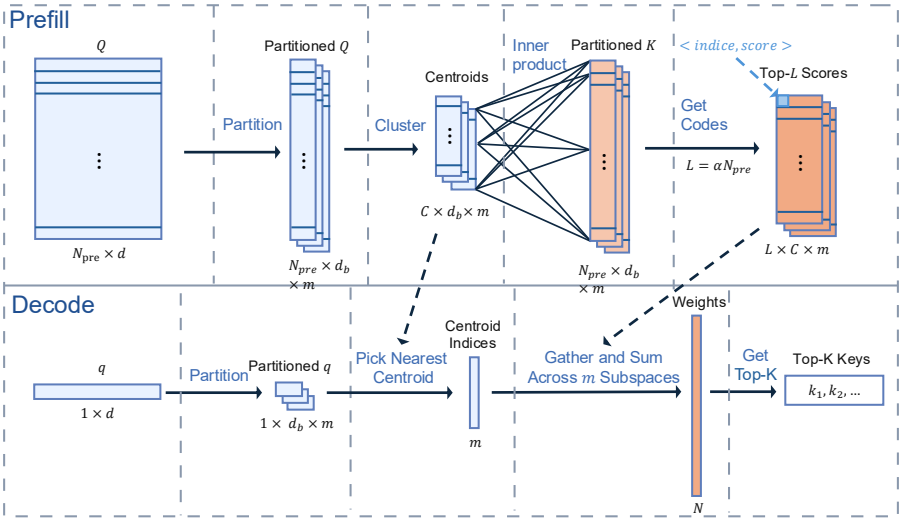


Figure 2: **CSAttention overview.** During prefix build, prefill queries are clustered in subspaces and converted into centroid-to-key Top-L score tables over original key IDs. During decoding, each query selects nearest fixed query centroids, aggregates signed partial scores by key ID, reserves recent-history slots within the final Top-K budget, and attends to the selected original KV entries.

where Top-L ranks pairs by score. Each entry holds an original key index and a low-precision signed score used only for retrieval; final attention uses original Q/K/V and native scaling.

This builds reusable, bounded tables rather than searching historical queries online; query awareness itself is not our novelty.

3.3 Online Decode: Centroid Routing and Sparse Accumulation

At decode step t, for each incoming query q_t, CSAttention first splits it into subspace vectors $q_t^{(b)}$. For each subspace, it chooses the nearest query centroid:

$$\hat{j}_b = \arg\max_{j\in\{1,\dots,C\}} \cos\left(q_t^{(b)}, c_j^{(b)}\right). \tag{4}$$

The method then fetches the corresponding lists $\mathcal{T}_{\hat{j}_b}^{(b)}$ across all subspaces. Candidate key scores are accumulated by key index:

$$\hat{s}_{t,i} = \sum_{b:i\in \mathrm{IDs}(\mathcal{T}_{\hat{j}_b}^{(b)})} s_{\hat{j}_b,i}^{(b)}. \tag{5}$$

Signed partial scores can raise or lower a key's accumulated proxy score when it appears in a selected list. Let $K_{\mathrm{sel}} = \lceil \rho N_{\mathrm{pre}} \rceil$, $\mathcal{I}_t = \bigcup_b \mathrm{IDs}(\mathcal{T}_{\hat{j}_b}^{(b)})$, and $\mathcal{W}_t$ be the $\min\{32, K_{\mathrm{sel}}\}$ most recent historical key IDs. We select

$$\mathcal{S}_t = \mathcal{W}_t \cup \mathrm{Top}_{K_{\mathrm{sel}}-|\mathcal{W}_t|}(\mathcal{I}_t \setminus \mathcal{W}_t;\ \hat{s}_{t,\cdot}), \tag{6}$$

where $\mathrm{Top}_r(A; s)$ returns up to r highest-scored IDs. Hence $|\mathcal{S}_t| \le K_{\mathrm{sel}}$: recent tokens consume slots *within* the reported 5% budget. Standard attention then uses the selected original K/V.

The routed lists nominate candidates, not extra attention positions: their union is reduced to at most ρN_{pre} selected IDs. Signed partial scores can demote a key that looks strong in another subspace. This remains a retrieval proxy: exact attention over selected original K/V cannot recover an absent key. We therefore report both Top-32 recall and task accuracy.

3.4 Incremental Maintenance for Long Generation

New keys are absent from prefix tables, so the 32-token window protects them initially. For later retrieval, each new key is scored against *every* fixed centroid in each subspace. Its (original key ID,

signed score) pair replaces a request-local list's tail only if its score exceeds the current L-th score. Lists remain bounded, the prefix snapshot stays unchanged, and keys leaving the window remain retrievable through successful insertions; the query codebook is never reclustered.

3.5 Complexity and Memory Trade-off

With $L = \lceil \alpha N_{\text{pre}} \rceil$, list storage per indexed group is $O(mC\alpha N_{\text{pre}})$ and centroid storage is $O(Cd)$. Thus, auxiliary memory remains linear in prefix length, controlled by (m, C, α).

At decode time, nearest-centroid routing costs $O(Cd)$ per routed query, fetching and reducing its m selected lists costs $O(m\alpha N_{\text{pre}})$, and sparse attention over at most $K_{\text{sel}} = \lceil \rho N_{\text{pre}} \rceil$ selected entries costs $O(K_{\text{sel}} d)$. Generated-key maintenance also scores each new key against the fixed mC centroids and conditionally updates bounded lists. Dense attention over N_t available positions costs $O(N_t d)$ per step. CSAttention retains linear dependence on prefix length, but reduces the online attention workload and makes retrieval regular and bounded by configured shortlist and keep ratios. It is an amortized systems trade-off for reusable-prefix serving, not an asymptotic elimination of context-length dependence.

3.6 Execution Modes

All-GPU mode. When HBM is sufficient, CSAttention stores the KV cache and lookup tables on GPU. Decode uses device-side centroid routing, list fetch, reduce-by-key accumulation, Top-K selection, and sparse attention. This mode avoids PCIe transfer and is best suited to 8B-class models or moderate auxiliary table sizes.

CPU↔GPU mode. When HBM is insufficient, CSAttention stores the KV cache and lookup tables in CPU DRAM. CPU workers perform shortlist processing, reduce-by-key accumulation, Top-K selection, and table maintenance; pinned-memory asynchronous copies stage the selected KV entries rather than the full cache to the GPU. Host work and GPU kernels can execute concurrently where their dependencies permit, and we report raw host work separately from the residual latency exposed after overlap. This does not assume that a future autoregressive query is available before the preceding token is generated. The mode supports larger contexts or backbones, at the cost of extra host storage, transfer, and scheduling complexity.

4 Experiments

Table 1: Main accuracy at 5% active prefix KV. Left: LongBench macro scores on three backbones. Right: matched-budget LongBench-v2 quality on all 503 Llama-3.1-8B-Instruct examples; the full comparison is in Appendix 10.

(a) LongBench macro.

Method	Llama	Qwen	Mistral
Full	52.41	52.30	49.92
CSAttention	**52.04**	**52.25**	**49.92**
PQCache	49.79	50.59	45.49
H_2O	45.11	48.04	37.45
SparQ	41.79	43.65	35.53
MagicPig	43.10	46.60	44.37

(b) LongBench-v2, Llama-3.1-8B.

Method	Overall	Easy	Hard	CoT
Full	31.0	35.4	28.3	36.5
CSAttention	**31.1**	34.7	29.1	36.2
PQCache	29.8	33.3	27.7	32.9
Quest	23.9	25.0	23.2	30.6
ShadowKV	30.4	34.3	28.0	32.6
HashAttention	30.0	34.3	27.3	34.9

4.1 Setting

Benchmarks and models. We evaluate Llama-3.1-8B-Instruct, Qwen3-8B, and Mistral-7B-Instruct-v0.3 on LongBench (Bai et al., 2024a), which covers question answering, summarization, retrieval, few-shot, and code tasks. We report both macro and task-level scores so that regressions are visible. Controlled quality comparisons use Llama-3.1-8B-Instruct on all 503 LongBench-v2 (Bai et al., 2024b) examples; RULER (Hsieh et al., 2024) further probes 128K contexts. Accuracy uses deterministic greedy decoding and the respective benchmark metrics.

Table 2: Task-level LongBench scores for the original three-model evaluation. MQA-E/Z: MultiFieldQA English/Chinese; NarQA: NarrativeQA; M-News: MultiNews; P-Ret: Passage Retrieval; G-Rep: GovReport.

Method	MQA-E	MQA-Z	NarQA	M-News	Musiq	Trec	Samsum	TrivQA	P-Ret	Hotpot	G-Rep	LCC	LSHT	VCSum	Avg
Llama-3.1-8B-Instruct															
Full	55.54	62.87	29.91	27.16	30.89	72.50	43.75	91.65	100.0	56.16	35.26	64.89	46.00	17.16	52.41
CSAttention	56.02	62.01	30.46	26.38	31.11	71.50	44.16	91.95	99.50	55.94	33.60	63.33	45.00	17.63	52.04
PQCache	52.96	57.57	30.14	16.67	28.69	71.00	40.01	91.82	99.00	55.22	34.01	60.52	43.00	16.38	49.79
H_2O	40.17	40.01	29.21	23.94	28.08	62.00	41.10	90.32	97.00	53.36	28.32	57.96	23.50	16.50	45.11
SparQ	39.56	34.32	26.96	21.78	28.48	47.00	42.11	89.26	87.00	51.83	25.21	55.42	21.00	15.18	41.79
MagicPig	48.78	53.16	25.86	14.50	19.20	70.00	42.00	65.05	96.00	38.50	23.39	61.11	38.00	7.83	43.10
Qwen3-8B															
Full	53.67	63.37	26.05	24.88	36.18	71.50	44.30	88.54	100.0	59.40	33.35	69.13	47.50	14.31	52.30
CSAttention	53.21	63.74	26.18	24.60	37.05	72.00	45.10	88.62	100.0	59.18	32.63	68.89	46.00	14.29	52.25
PQCache	51.98	60.35	26.98	21.90	36.90	72.00	42.80	84.10	100.0	58.73	33.09	61.01	44.00	14.39	50.59
H_2O	50.10	57.90	26.01	23.99	34.20	61.00	44.50	85.00	98.50	54.12	29.99	63.01	31.00	13.21	48.04
SparQ	45.32	50.07	25.88	21.03	31.90	59.00	43.90	80.20	91.00	47.93	27.62	48.32	25.50	13.48	43.65
MagicPig	52.11	57.32	26.31	18.94	28.08	58.00	44.22	87.90	98.50	51.88	24.32	55.67	40.00	9.12	46.60
Mistral-7B-Instruct-v0.3															
Full	50.21	53.19	27.74	26.57	26.50	70.00	46.30	89.04	97.00	51.08	34.22	64.32	47.00	15.68	49.92
CSAttention	49.92	52.94	25.56	27.06	26.10	70.50	45.91	90.59	97.00	49.34	32.88	63.98	46.00	16.44	49.92
PQCache	45.57	39.59	22.57	26.04	22.30	71.00	42.18	88.62	89.00	35.22	29.68	64.01	46.00	15.01	45.49
H_2O	37.26	30.43	21.07	25.33	17.01	63.00	41.98	84.77	52.00	31.56	22.92	59.91	31.00	6.04	37.45
SparQ	31.51	31.77	19.62	21.86	15.63	61.00	41.68	84.10	42.00	29.69	25.67	53.01	34.00	5.81	35.53
MagicPig	45.87	38.91	26.01	23.34	21.42	71.00	45.02	90.15	95.00	34.98	31.29	55.04	29.00	14.08	44.37

Baselines and controls. Alongside Full Attention, the original LongBench comparison includes PQCache, H_2O, SparQ, and MagicPig; the matched-budget comparison adds Quest, SnapKV, PyramidKV, ShadowKV, HashAttention, and RetrievalAttention. Our key-centric control keeps the subspace and centroid counts, shortlist and final KV budgets, reduction, recent-window rule, maintenance, and sparse-attention backend fixed, changing only whether prefill queries or keys are clustered. RetrievalAttention is query-informed; its equivalent compatibility implementation supports quality comparisons, not speed claims.

Configuration and fairness. The default selector uses $\rho = 5\%$ active prefix KV, $m = 8$ subspaces, $C = 64$ centroids per subspace, shortlist ratio $\alpha = 0.10$, per-token routing, and a 32-token recent window *inside* the final active-KV budget. All matched sparse-quality methods access 5% of prefix KV; LongBench-v2 quality uses 503 examples, whereas R@32 uses a fixed 50-example subset. Within each throughput comparison, hardware, checkpoint, precision, and timing boundary are shared. The CPU↔GPU sweep explicitly labels its other keep ratios and search periods; decode-only throughput excludes construction, while TTFT and end-to-end latency include it. Further implementation and timing details appear in Appendix A.2.

4.2 Main Accuracy under High Sparsity

Table 1 compares the three LongBench backbones and the latest matched-budget LongBench-v2 results. CSAttention's macro scores differ from Full by -0.37, -0.05, and 0.00 for Llama, Qwen, and Mistral. Its 31.1 versus 31.0 LongBench-v2 score under deterministic decoding is best described as *matching* Full. Table 2 exposes task-level variation: on NarrativeQA, Llama and Qwen improve slightly, whereas Mistral drops from 27.74 to 25.56. Table 3 tests a 128K context: CSAttention retains 70.2 versus Full's 71.3 overall score, but retrieval and tracking declines preclude a uniformly lossless claim.

4.3 Mechanism Study: Query-centric vs. Key-centric Retrieval

Cached *query*-centroid scores yield a better selector than a matched key-centroid control. Both use the same subspace and centroid counts, shortlist and final KV budgets, reduction, recent-window rule, maintenance, and sparse-attention backend; only the clustered object changes. Table 4 shows

Table 3: RULER-128K on Llama-3.1-8B-Instruct; CSAttention accesses 5% of prefix KV.

Method	Retrieval	Tracking	Aggregation	QA	Overall
Full	90.1	22.8	33.0	58.0	71.3
CSAttention	88.7	20.9	34.0	56.9	70.2

Table 4: Controlled LongBench-v2 comparison at 5% active prefix KV. Quality uses 503 examples; R@32, Full top-32 recall, uses a fixed 50-example subset. CoT denotes CoT-2048.

Method	R@32	Overall	Easy	Hard	CoT
Full	100.0	31.0	35.4	28.3	36.5
CSAttention	**92.7**	**31.1**	34.7	29.1	36.2
Key-centric control	87.6	29.8	33.4	27.2	34.7
RetrievalAttention	88.9	30.0	32.6	28.7	33.1
PQCache	83.9	29.8	33.3	27.7	32.9

gains of 5.1 R@32 points and 1.3 LongBench-v2 points. RetrievalAttention is a query-informed baseline, not a key-only strawman: it builds an approximate-nearest-neighbor key index from prefill attention. Our reusable signed centroid-to-key scores avoid rescoring historical keys at decode time. Its quality-compatibility backend is excluded from speed comparisons.

4.4 Decode Throughput

Figure 3 reports CPU↔GPU and all-GPU deployment sweeps. At 128K, the CPU↔GPU $\rho = 20\%$, search-every-eight-steps point reaches $8.26\times$, $7.85\times$, $17.9\times$, and $1.33\times$ the throughput of PQCache, MagicPig, SparQ, and H_2O; these are not matched-5% comparisons. The all-GPU 5%/per-step curve reaches $4.24\times$ Full. Table 5 provides a direct same-backend check at matched 5% active KV: Quest reaches $3.17\times$ and CSAttention $4.24\times$ against the same FlashAttention Full reference. A separate A100 timing run attributes 18.69%, 33.35%, 4.62%, and 43.34% of default 128K decode time to routing, shortlist gathering/reduction, maintenance, and sparse attention (Appendix 12); its absolute tok/s is not combined with the sweep.

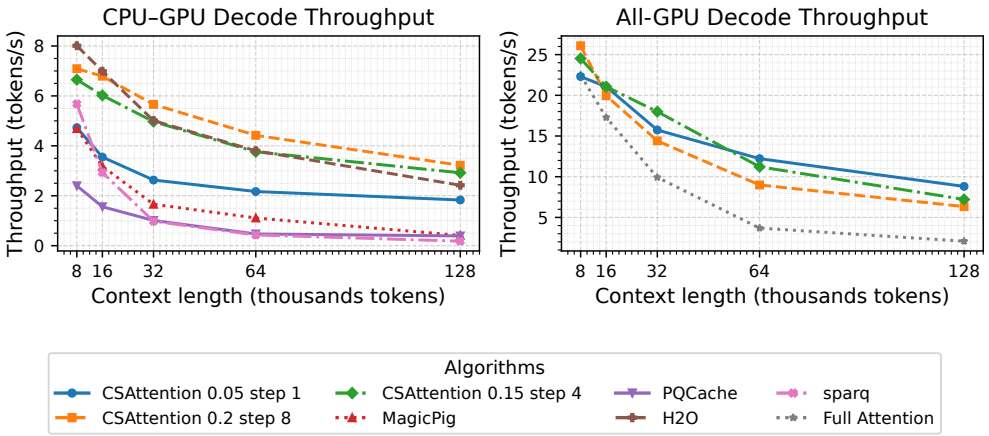


Figure 3: Decode throughput in CPU↔GPU (left) and all-GPU (right) modes. Legends give active-KV fraction and search period. The 128K CPU↔GPU ratios use 20%/step-8; all-GPU $4.24\times$ uses 5%/step-1. Timing boundaries differ across panels.

Table 5: 128K all-GPU decode at matched 5% active prefix KV; Quest and CSAttention share a sparse backend and FlashAttention Full reference. Excludes construction.

Method	Active prefix KV	Speedup vs. Full
Full	100%	1.00×
Quest	5%	3.17×
CSAttention	5%	4.24×

4.5 End-to-end Latency and TTFT Trade-off

CSAttention pays a one-time prefix-build cost to lower amortized decoding cost. Table 6 reports TTFT and full-request latency together. At 128K, TTFT rises from 11.5 to 14.0 seconds, yet one-off latency breaks even near 120 generated tokens and D2048 end-to-end speedup reaches 3.80×. Table 7 quantifies reuse: D32 needs seven, six, or four requests sharing a 32K, 64K, or 128K prefix, respectively; at D512, one continuation repays construction at all three lengths.

Table 6: Latency trade-off. TTFT and end-to-end (E2E) include construction; E2E also includes prefill and decoding. D_{BE} is the approximate one-request break-even length in tokens.

(a) TTFT (s) and break-even.

Prefix	Full	CS	Extra	D_{BE}
32K	3.1	3.8	0.7	220
64K	5.9	7.1	1.2	180
128K	11.5	14.0	2.5	120

(b) E2E speedup versus Full.

Prefix	D128	D512	D1024	D2048
32K	0.88×	1.15×	1.45×	1.80×
64K	0.92×	1.35×	1.85×	2.50×
128K	1.00×	1.70×	2.60×	3.80×

Table 7: Minimum requests sharing one prefix for CSAttention to beat Full end-to-end; 1 means a single continuation repays construction.

Prefix	D32	D64	D128	D512	D2048
32K	7	4	2	1	1
64K	6	3	2	1	1
128K	4	2	1	1	1

4.6 Memory Footprint

CSAttention adds an index to the original KV cache rather than compressing it. Table 8 restores the model/prefix capacity comparison. At 128K for Llama-3.1-8B, roughly 20 GiB of model state, 16.0 GiB of KV, and 9.6 GiB of tables total 45.6 GiB. On an 80 GiB A100, base-artifact arithmetic allows two *distinct* CSAttention prefixes (71.2 GiB) versus three Full-Attention prefix KVs (68.0 GiB), before workspace and request-local generated state; many continuations of one prefix share its immutable table. Exact 8K chunked construction reduces measured peak GPU allocation from 55.452 to 41.856 GiB at 128K with 1.05× TTFT and 1.01× D512 E2E latency (Appendix 11).

Table 8: Memory in GiB: model state, full KV, and score tables. A100 80 GiB fit is an artifact estimate, excluding workspace and concurrency.

Model / prefix	KV	Table	Combined	Fits?
Llama-3.1-8B / 64K	8.0	4.8	32.8	Yes
Llama-3.1-8B / 128K	16.0	9.6	45.6	Yes
Llama-3.1-8B / 256K	32.0	19.2	71.2	Yes
Qwen3-8B / 128K	18.0	10.8	48.8	Yes
Llama-70B / 64K	20.0	12.01	180.01	No

4.7 Large-batch Throughput and Long-generation Robustness

Table 9 reports batched serving and larger-model CoT stress tests. Against PQCache, CSAttention's throughput advantage rises from $1.38\times$ at batch 1 to $1.67\times$ at batch 16. At a 2048-token cap it remains close to Full on Llama-70B and Qwen3-32B, with varying task buckets. At 4096 and 8192 generated tokens, a separate long-context diagnostic shows no abrupt drift-induced collapse within the tested subsets; the Full–CS gap widens from 0.3 to 1.0 point as active samples fall from 209 to 74 (Appendix 14).

Table 9: Batch and long-generation stress tests. Left: decode throughput (tok/s). Right: LongBench-v2 CoT quality at a 2048-token cap. PQ: PQCache; CS: CSAttention.

(a) Batch throughput.

Batch	PQ	CS	CS/PQ
1	8.1	11.2	1.38×
4	19.7	28.2	1.43×
8	27.5	42.6	1.55×
16	34.1	57.0	1.67×

(b) CoT-style quality.

Model	Method	Overall	Easy	Hard	Long
Llama-70B	Full	36.5	38.6	35.2	27.6
Llama-70B	CS	36.4	39.1	34.8	25.9
Qwen3-32B	Full	49.2	53.1	46.8	47.2
Qwen3-32B	CS	48.1	51.1	46.2	49.4

5 Related Work

Sparse selection and vector retrieval. Prior methods use sparse patterns, heavy-hitter retention, or partial KV fetching (Xiao et al., 2024; Fu et al., 2025; Zhu et al., 2024; Zhang et al., 2023; Ribar et al., 2024). Quest prunes key blocks (Tang et al., 2024); PQCache indexes product-quantized keys (Zhang et al., 2025a); MagicPig and HashAttention use hashing (Chen et al., 2024; Desai et al., 2024). RetrievalAttention builds an attention-aware key index using prefill query–key relations (Liu et al., 2024). CSAttention likewise exploits prefill queries, but caches signed query-centroid-to-key scores so decode uses centroid routing, bounded list reads, and reduce-by-key rather than online key-vector search.

KV compression and clustering. SnapKV and PyramidKV shrink prompt KV (Li et al., 2024; Cai et al., 2024); ClusterKV clusters keys (Liu et al., 2025), and ClusterAttn uses attention clustering for KV compression (Zhang et al., 2025b). CSAttention retains original K/V and adds score tables: its budget constrains *active* KV access, not storage. We therefore account for auxiliary memory explicitly.

Reusable-prefix serving. Prefix caching serves retrieval-augmented generation, agents, and repeated prompts (Lu et al., 2024; Jin et al., 2024; Gao et al., 2024; Lee et al., 2025; Kwon et al., 2023). CSAttention caches tables with prefix KV; construction is amortized across cache hits but increases cold-request TTFT, which our evaluation measures.

6 Conclusion and Limitations

CSAttention turns prefill-query structure into reusable centroid-to-key score tables. Bounded routing and signed shortlist aggregation select original KV entries for standard attention, yielding near-Full quality at a small active budget and faster long-prefix decoding. Its benefit is conditional rather than free: cold requests pay index construction and increased TTFT; distinct cached prefixes each require an auxiliary table whose memory scales linearly with prefix length. Some retrieval/tracking and task-level scores decline, and the long-generation diagnostic only covers up to 8192 generated tokens. The intended operating point is a reused long prefix or a sufficiently long generation that amortizes construction. Future work includes reducing table memory and integrating the index with production prefix-cache managers.

AI USE STATEMENT

Generative AI assisted literature discovery, experimental planning, and manuscript editing. The authors remain responsible for verifying originality, citations, code, results, and all claims.

REPRODUCIBILITY STATEMENT

The main text and appendix specify the method, default configuration, KV-budget accounting, evaluation protocols, and systems timing boundaries.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, 2023.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*, 2024b.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, et al. Magicpig: Lsh sampling for efficient llm generation. *arXiv preprint arXiv:2410.16179*, 2024.

Aditya Desai, Shuo Yang, Alejandro Cuadron, Matei Zaharia, Joseph E Gonzalez, and Ion Stoica. Hashattention: Semantic sparsity for faster inference. *arXiv preprint arXiv:2412.14468*, 2024.

Tianyu Fu, Haofeng Huang, Xuefei Ning, Genghan Zhang, Boju Chen, Tianqi Wu, Hongyi Wang, Zixiao Huang, Shiyao Li, Shengen Yan, et al. Moa: Mixture of sparse attention for automatic large language model compression. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025.

Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. {Cost-Efficient} large language model serving for multi-turn conversations with {CachedAttention}. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pp. 111–126, 2024.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling*, 2024.

Chao Jin, Zili Zhang, Xuanlin Jiang, Fangyue Liu, Shufan Liu, Xuanzhe Liu, and Xin Jin. Ragcache: Efficient knowledge caching for retrieval-augmented generation. *ACM Transactions on Computer Systems*, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Hyungwoo Lee, Kihyun Kim, Jinwoo Kim, Jungmin So, Myung-Hoon Cha, Hong-Yeon Kim, James J Kim, and Youngjae Kim. Shared disk kv cache management for efficient multi-instance inference in rag-powered llms. *arXiv preprint arXiv:2504.11765*, 2025.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, 2024.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, et al. Retrievalattention: Accelerating long-context llm inference via vector retrieval. *arXiv preprint arXiv:2409.10516*, 2024.

Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo. Clusterkv: Manipulating llm kv cache in semantic space for recallable compression. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*, pp. 1–7. IEEE, 2025.

Songshuo Lu, Hua Wang, Yutian Rong, Zhi Chen, and Yaohua Tang. Turborag: Accelerating retrieval-augmented generation with precomputed kv caches for chunked text. *arXiv preprint arXiv:2410.07590*, 2024.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. Sparq attention: bandwidth-efficient llm inference. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 42558–42583, 2024.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: query-aware sparsity for efficient long-context llm inference. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 47901–47911, 2024.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024.

Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. Pqcache: Product quantization-based kvcache for long context llm inference. *Proceedings of the ACM on Management of Data*, 3(3):1–30, 2025a.

Minwei Zhang, Haifeng Sun, Jingyu Wang, Shaolong Li, Wanyi Ning, Qi Qi, Zirui Zhuang, and Jianxin Liao. ClusterAttn: KV cache compression under intrinsic attention clustering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 14451–14473, 2025b.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

Qianchao Zhu, Jiangfei Duan, Chang Chen, Siran Liu, Xiuhong Li, Guanyu Feng, Xin Lv, Huanqi Cao, Xiao Chuanfu, Xingcheng Zhang, et al. Sampleattention: Near-lossless acceleration of long context llm inference with adaptive structured sparse attention. *arXiv preprint arXiv:2406.15486*, 2024.

A Implementation and Evaluation Protocol

A.1 Prefix Artifact and Routing

For a reusable prefix, CSAttention stores the standard KV cache and query-centroid score tables under the same cache key. A cache hit reuses an immutable prefix snapshot; generated-suffix KV and list updates are request-local. Prefix queries are clustered separately in each of m subspaces. Each

centroid list holds original key positions and *signed* partial dot-product scores, not compressed K/V. A decode query routes to one centroid per subspace, sums scores by original key position, selects a bounded KV set, and applies native attention to the gathered original K/V. Each generated key is scored against all fixed centroids and inserted into a request-local list only when its score exceeds the current Top-L tail. The query codebook is not retrained during decoding. The most recent 32 historical tokens occupy slots *inside* the final active-KV budget.

A.2 Fairness and Timing Boundaries

Within each throughput comparison, methods use the same hardware generation, checkpoint, tokenizer, precision, context construction, batch size, decode length, warm-up procedure, and timing boundary. Matched sparse comparisons equalize *active* prefix-KV fraction, not physical cache footprint; Full attends to all KV. The CPU↔GPU deployment sweep includes 5%, 15%, and 20% CSAttention operating points, searching every one, four, and eight decode steps, respectively. Quoted 128K ratios to PQCache, MagicPig, SparQ, and H_2O use the 20%/step-8 point, whereas the all-GPU $4.24\times$ versus Full uses 5%/step-1. Pinned memory and asynchronous transfers are used where host staging is part of a method. Decode-only throughput assumes a built prefix artifact; TTFT and end-to-end latency include its construction.

The key-centric control shares CSAttention's subspace and centroid counts, shortlist length, final Top-K budget, reduce-by-key step, recent-window rule, incremental maintenance, and sparse-attention path. Only the clustered object changes from prefill queries to keys. RetrievalAttention instead uses prefill attention information to build an attention-aware approximate-nearest-neighbor index over keys; its mathematically equivalent compatibility implementation is used for quality, not systems-speed claims. The matched LongBench-v2 quality comparison uses all 503 examples, while R@32 uses a fixed 50-example instrumentation subset. All sparse methods access 5% of prefix KV; the default CSAttention selector sets $\rho = 5\%$, $C = 64$, $m = 8$, and $\alpha = 0.10$, with per-token routing. Accuracy uses deterministic greedy decoding. A small positive score difference from Full is interpreted as matching, not an intrinsic accuracy gain from sparsity.

B Supplemental Accuracy and Matched-Backend Results

Table 10: Complete matched-5% active prefix-KV LongBench-v2 comparison. Quality uses all 503 examples; R@32 uses the fixed 50-example instrumentation subset. Equal active KV does not imply equal stored-KV memory.

Method	R@32	Overall	Easy	Hard	CoT
Full	100.0	31.0	35.4	28.3	36.5
CSAttention	92.7	31.1	34.7	29.1	36.2
Key-centric control	87.6	29.8	33.4	27.2	34.7
RetrievalAttention	88.9	30.0	32.6	28.7	33.1
PQCache	83.9	29.8	33.3	27.7	32.9
Quest	82.4	23.9	25.0	23.2	30.6
SnapKV	84.6	28.8	30.2	28.0	31.4
PyramidKV	83.2	29.4	30.2	29.0	34.6
ShadowKV	86.7	30.4	34.3	28.0	32.6
HashAttention	84.9	30.0	34.3	27.3	34.9

C Deployment Cost and Sensitivity

C.1 Prefix Build and Reuse

Figure 4 shows prefill latency in both execution modes. The table-build cost is included in the main-text TTFT and end-to-end comparisons (Table 6); main-text Table 7 quantifies how many continuations sharing a prefix repay construction at shorter decode lengths.

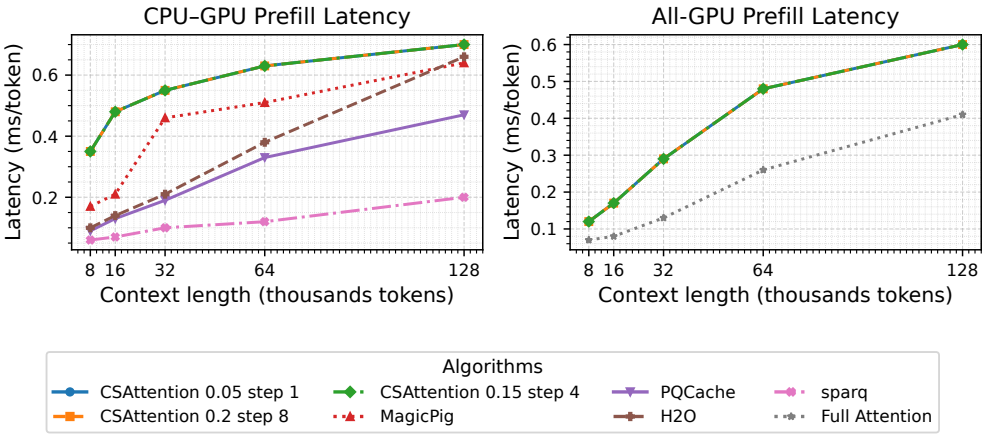


Figure 4: Long-context prefill latency in CPU↔GPU and all-GPU modes. Table construction adds cold-build work but is shared by later cache hits to the same prefix.

C.2 Memory Accounting and Chunked Construction

For prefix length N_{pre}, m subspaces, C centroids per subspace, and list length $L = \lceil \alpha N_{\text{pre}} \rceil$, each indexed KV-head group stores mCL original-key positions and signed scores. Thus the index is $O(mC\alpha N_{\text{pre}})$ auxiliary storage and the original KV remains intact. Multiple continuations of one cached prefix reuse its immutable table, but divergent suffix KV and mutable insertion state are request-local. The distinct-prefix capacity arithmetic in Table 8 excludes runtime workspace, fragmentation, and aggregate concurrent-request state.

Exact two-pass chunked construction first fixes query centroids, then streams prefix keys through the table build. It changes peak build memory, not score-table semantics. Table 11 reports latency ratios relative to a single 128K build, so values above $1\times$ mean slower, not faster.

Table 11: 128K chunked prefix construction. Ratios are relative *latencies* (higher is slower); GPU and CPU columns are peak allocated GiB. E2E D512 includes prefill, build, and 512 decode tokens.

Chunk	Build	TTFT	GPU GiB	CPU GiB	E2E D512
Single 128K	$1.00\times$	$1.00\times$	55.452	5.793 ± 0.011	$1.00\times$
16K	$1.26\times$	$1.02\times$	42.707	5.793 ± 0.008	$1.01\times$
8K	$1.51\times$	$1.05\times$	41.856	5.796 ± 0.008	$1.01\times$

C.3 128K Decode Component Costs

Table 12 is a separate A100 80 GiB instrumented run. The default gives 12.0 ± 0.4 tok/s within this run's implementation and timing boundary; it is not substituted into Figure 3 or mixed with the faster NPU used for a separate memory/parameter sweep. Non-default throughput entries are normalized to the default, not absolute tok/s. Raising ρ shifts the work share toward sparse attention, whereas increasing C or m raises bounded maintenance cost.

Table 12: A100 128K decode-component split. Shares sum to approximately 100% after rounding. The default row gives absolute tok/s; other throughput entries are relative to it.

Sweep	Value	Route %	Shortlist %	Maint. %	Sparse attn. %	Throughput
Default	$\rho = 5\%, C = 64, m = 8$	18.69 ± 0.09	33.35 ± 0.10	4.62 ± 0.03	43.34 ± 0.06	12.0 ± 0.4 tok/s
ρ	10%	15.21 ± 0.04	26.47 ± 0.05	3.67 ± 0.02	54.65 ± 0.06	$0.82\times$
ρ	20%	10.16 ± 0.20	18.60 ± 0.07	2.44 ± 0.03	68.80 ± 0.26	$0.73\times$
C	32	19.45 ± 0.16	32.97 ± 0.05	3.91 ± 0.02	43.67 ± 0.18	$1.08\times$
C	128	19.17 ± 0.03	32.24 ± 0.12	6.16 ± 0.02	42.43 ± 0.16	$0.90\times$
m	4	19.51 ± 0.05	32.54 ± 0.19	3.93 ± 0.01	44.02 ± 0.22	$1.02\times$
m	16	18.64 ± 0.17	33.75 ± 0.21	6.05 ± 0.03	41.56 ± 0.04	$0.96\times$

Table 13: CPU↔GPU 128K, 5% active-KV raw work shares in a separate run from the 20%/step-8 Figure 3 point. The four components divide raw work, not additive wall-clock time; host exposed/raw is residual host latency after overlap divided by raw CPU processing, host gather, and H2D time.

CPU process	Host gather	H2D	GPU sparse attn.	Host exposed/raw
$39.12 \pm 0.73\%$	$4.79 \pm 0.39\%$	$6.44 \pm 0.23\%$	$49.66 \pm 0.90\%$	$21.81 \pm 0.17\%$

D Long-generation Diagnostic

Table 14: Long-context CoT diagnostic. Scores use only active examples in the final 512 generated tokens at each cap; the rows have different sample counts and are not a paired trajectory. Query-centroid cosine is measured in the same window.

Max gen.	Active N	Full	CS	CS–Full	Cosine
2048	209	36.5	36.2	-0.3	0.89
4096	156	37.1	36.4	-0.7	0.84
8192	74	37.4	36.4	-1.0	0.82

The active count falls from 209 to 74 and the gap widens to one point at 8K. These data show no abrupt failure in the tested long-context setup, not stability at untested generation lengths.

E Limitations and Discussion

CSAttention targets long reusable-prefix requests. Short one-off prompts may not repay construction. Tables add $O(mC\alpha N_{\text{pre}})$ storage to the original KV cache and can reduce the number of distinct prefixes fitting on one device. Concurrent continuations require request-local suffix KV and update state beyond the shared immutable artifact; single-continuation measurements do not quantify aggregate concurrent memory or throughput. CPU↔GPU execution and chunked build exchange additional complexity or latency for capacity. At 5% active KV, RULER-128K retrieval and tracking decline relative to Full, and Mistral NarrativeQA loses 2.18 points despite a stable LongBench macro average. The 8K-generation diagnostic covers only active LongBench-v2 examples in this setting, not arbitrary reasoning models or unbounded generation.