TESSERAKV: KV PAGING VIA CO-RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic workloads increasingly operate over long contexts, causing KV caches to grow rapidly and making memory bandwidth a major bottleneck during generation. KV paging [1] alleviates this cost by retaining the cache while retrieving only the pages needed at each decoding step. There are two critical challenges in KV paging: how keys are grouped into pages and how pages are scored for retrieval. Existing methods construct pages using heuristics, and score those pages using a separate criterion. These two decisions should instead be derived from a common objective: whether the keys in a page are likely to be retrieved together. We propose TesseraKV, which derives both page construction and page scoring from *co-retrieval*: keys should share a page when their attention logits tend to vary together under the query distribution of a head. This common principle yields a covariance-weighted geometry for partitioning keys and a spread-aware score for estimating the attention mass of each page. Because the decoding-query distribution is unavailable when the cache is constructed, we further study how to estimate this geometry from queries available beforehand. We find that prefill queries provide a useful but task-dependent signal for decoding-query geometry, while incorporating queries from the downstream instruction substantially improves the co-retrieval structure relevant to partitioning. Across a range of LLMs, long-context tasks, and agentic workloads, TesseraKV consistently improves downstream generation quality over state-of-the-art KV retrieval methods at matched KV budgets. For example, on CLAW-EVAL agentic tasks, TesseraKV achieves an overall success rate of 70.32, improving over the previous state of the art at 42.58 by 27.74 points.

1 INTRODUCTION

Agentic workloads have become a central use of large language models. Unlike traditional chat interaction, where each turn contributes comparable numbers of prompt and response tokens, agentic workloads accumulate context through tool outputs, file contents, and search results far exceeding token generation. Across turns, the consumed context therefore grows steadily while the number of generated tokens remains relatively flat. As a result, the KV cache expands rapidly, and reading from it increasingly dominates the decoding latency. Figure 1 shows that input and output token counts follow distinctly different trends across turns in agentic workloads and chat interaction. Over a SWE-BENCH (Jimenez et al., 2024) trajectory, the median context read grows by a factor of 100 from the first to the last turn. Summed over a session, an agent reads more than 200 context tokens for every token it generates, compared with $2.3\times$ for open-ended chat (ShareGPT, 2023). Similar trends are observed in TERMINALBENCH (Merrill et al., 2026) and CLAW-EVAL (Ye et al., 2026).

To address this challenge, recent studies have proposed methods for computing sparse attention. One line of work uses *eviction*: tokens scored as unimportant are discarded from the KV cache (Li et al., 2024; Zhang et al., 2023; Park et al., 2026). Once discarded, these tokens cannot be recalled in the future. In a long trajectory, this can be dangerous: a token that appears irrelevant at one point may become crucial for a later turn. An alternative is *retrieval*, which preserves the full KV cache while selectively accessing only the portions relevant to the current query (Tang et al., 2024; Liu et al., 2025a; Xiao et al., 2026). Since scoring every key individually would defeat the purpose of retrieval, keys are grouped into pages that are fetched as a unit. We use the term *KV paging* to emphasize the

[1] We use *KV paging* to denote the partitioning of the KV cache into pages that serve as units of retrieval. This term refers to how keys are organized and scored for retrieval, and is different from memory-management mechanisms like PagedAttention, which organize the KV cache into blocks for efficient memory allocation.

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.

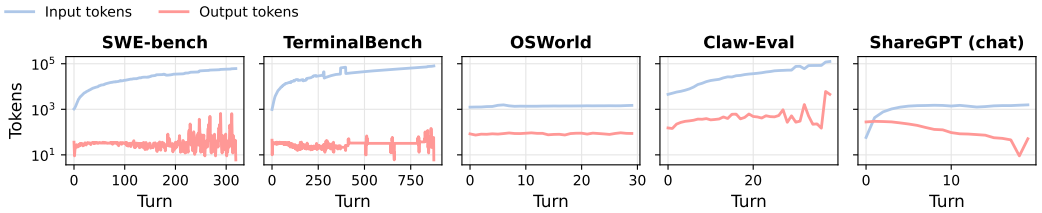

Figure 1: Input and output tokens per turn across four agentic workloads and open-ended chat. Lines show median tokens at each turn over active sessions. Agentic workloads accumulate input tokens faster across turns while output token counts remain relatively stable, in contrast to open-ended chat.

two coupled decisions in this process: *partition*, which determines how the keys are grouped, and *gate*, which determines how those pages are scored for retrieval.

Existing methods make these two decisions based on heuristics and treat them independently. Position-based approaches group contiguous tokens (Tang et al., 2024; Chen et al., 2024), assuming that attention is local; similarity-based approaches group keys with high cosine similarity (Liu et al., 2025a; Wu et al., 2026), assuming that semantically close keys are retrieved together. These approaches characterize key relationships independently of the queries that determine retrieval. In contrast, we characterize relationships between keys through how their attention logits co-vary under the query distribution, directly capturing their tendency to be retrieved together.

We refer to this tendency as *co-retrieval* and use it to jointly design the partition and gate. We show that co-retrieval induces a covariance-weighted geometry over key vectors: the resulting partition minimizes co-retrieval spread within each page, while the gate accounts for the residual spread that no finite partition can eliminate. We highlight the following contributions of TesseraKV:

- We introduce *co-retrieval*, a novel objective that directly optimizes whether keys can be retrieved together under a given query distribution. From this objective, we derive an efficient k-means formulation for page partitioning and a closed-form solution for the gate.
- We design metrics and estimators for predicting the query distribution at decode-time. We propose a simple *query-mix* estimator that combines queries from the context with queries generated from the downstream instruction, effectively closing the gap to the reference on the co-retrieval metric.
- We conduct extensive experiments across a range of LLMs and show that TesseraKV substantially outperforms SoTA methods. TesseraKV achieves an overall success rate of 70.32 on CLAW-EVAL, improving over the previous state of the art at 42.58 by 27.74 points.

2 Background and Related Work

Long-Context Models Context windows have grown from a few thousand tokens to millions of tokens. Recent long-context models increasingly adopt hybrid architectures (at al., 2026; Qwen Team, 2026; Team et al., 2026; Meta Superintelligence Lab, 2026; Cohere & Cohere Labs, 2026; Cohere et al., 2025), interleaving local layers based on either sliding-window attention or gated linear recurrence (Yang et al., 2025) with a small number of global attention layers that enable long-range understanding and retrieval. These global layers are often free of positional encoding (Yang et al., 2026; Gelberg et al., 2026). Consequently, the cost of serving long inputs is concentrated in these layers. Their dominant cost is the KV cache, which has shape seq $\times$ heads $\times$ d per layer and must be read in full at every decoding step, making memory bandwidth the primary bottleneck to generation.

Architectural design reduces two of the three dimensions of the KV cache. Sharing KV representations across query heads (Shazeer, 2019; Ainslie et al., 2023) reduces the head dimension. The feature dimension can be reduced either architecturally, such as MLA and low-rank approaches (Liu et al., 2024; Wong et al., 2025a; Wang et al., 2025), or by lowering the precision of the KV elements stored in the cache (Zandieh et al., 2026; Muller et al., 2026).

KV Cache Eviction Eviction methods compress the sequence axis at inference time by permanently discarding low-scoring KV entries. Attention-based methods score keys from the attention matrix

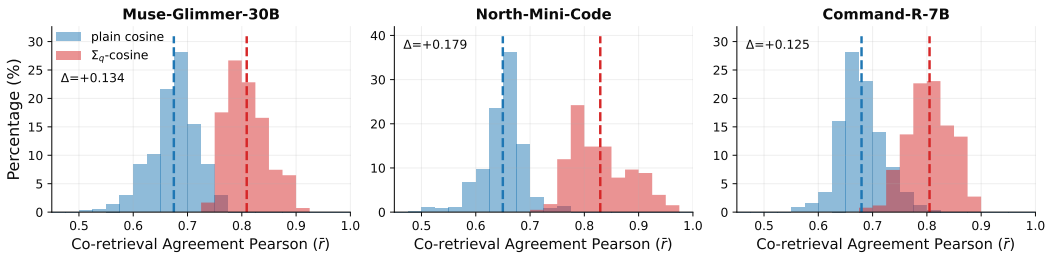

Figure 2: Co-retrieval agreement between predicted and empirical attention-logit correlations for three models. Each panel shows the distribution of per-head Pearson correlations $\bar{r}$ across attention heads, averaged over six domains from SlimPajama (Shen et al., 2023). Distributions concentrated further to the right indicate higher Pearson correlations and thus more accurate prediction of co-retrieval.

(Li et al., 2024; Zhang et al., 2023; Cai et al., 2024; Liu et al., 2026), while key-based methods use geometric criteria such as distance from local centroids or from the key manifold (Park et al., 2026; Datta et al., 2026). *Expected Attention* (Devoto et al., 2025) is most closely related to our work: it models head queries as Gaussian and scores each key by its expected attention under that distribution. Eviction leaves the decoding path unchanged but is irreversible. A dropped key required by a later query is unavailable rather than merely deprioritised, and performance degrades at tight budgets.

KV Cache Retrieval Retrieval methods avoid permanent eviction by retaining the KV cache and grouping keys into *pages* that are fetched as a unit. The cache may be offloaded to CPU memory or another memory tier. At each decoding step, they select only the pages relevant to the current query, leaving the remaining cache available for future steps. Hence this introduces two design decisions in the choice of the *partition* and the *gate*. QUEST (Tang et al., 2024) and ArkVale (Chen et al., 2024) partition keys by position and score a page using an upper bound on the largest logit it may contain. ClusterKV (Liu et al., 2025a) and LouisKV (Wu et al., 2026), in contrast, partition keys by content using cosine similarity and score each page from its centroid.

Both partitioning strategies rely on a heuristic assumption: tokens that are similar in key space or position are likely to be retrieved together. We instead characterize co-retrieval through how attention logits vary together under each head's query distribution, and use this relationship to derive both the partition objective and the page-scoring rule.

3 TESSERAKV

We first formalize this co-retrieval principle by modeling queries as a distribution in Section 3.1, then derive the partition solution in Section 3.2 and a corresponding gating strategy in Section 3.3. Since the distribution of queries at decode time is unavailable when pages are constructed, we discuss how to estimate its covariance from the context and downstream instruction in Section 3.4.

3.1 KEY RETRIEVAL IN AUTOREGRESSIVE TRANSFORMERS

We consider a single attention head in a decoder-only transformer (Vaswani et al., 2017), with head dimension d. At decoding step t, the query $\boldsymbol{q}_t \in \mathbb{R}^d$ scores each cached key $\boldsymbol{k}_i \in \mathbb{R}^d$ through

$$a_{ti} = \frac{\exp(\ell_{ti})}{\sum_{j \le t} \exp(\ell_{tj})}, \qquad \ell_{ti} = \frac{\boldsymbol{q}_t^\top \boldsymbol{k}_i}{\sqrt{d}}, \tag{1}$$

where ℓ_{ti} is the attention logit and a_{ti} is the normalized attention weight.

Co-retrieval. We say that a query $\boldsymbol{q}_t$ *retrieves* a key $\boldsymbol{k}_i$ when the attention score a_{ti} is large. Retrieval is therefore a property of the pair $(\boldsymbol{q}_t, \boldsymbol{k}_i)$ instead of the key alone, and a page is well formed when its members are retrieved by the same queries. Since we construct the pages independently of the any particular query at inference time, it is useful to characterize co-retrieval through a query distribution. We therefore model the queries arriving at a given head as draws of a random vector $\boldsymbol{q}$ with covariance $\boldsymbol{\Sigma}_{\mathrm{q}} = \mathrm{Cov}(\boldsymbol{q}) \in \mathbb{R}^{d \times d}$, and defer the question of how $\boldsymbol{\Sigma}_{\mathrm{q}}$ is estimated to Section 3.4.

Writing $\ell_i(\boldsymbol{q}) = \boldsymbol{q}^\top \boldsymbol{k}_i/\sqrt{d}$ for the logit of key $\boldsymbol{k}_i$ under such a query, we say that two keys are *co-retrieved* to the extent that their logits fluctuate together:

$$\rho_{ij} \;=\; \frac{\mathrm{Cov}\big(\ell_i(\boldsymbol{q}), \ell_j(\boldsymbol{q})\big)}{\sqrt{\mathrm{Var}\big(\ell_i(\boldsymbol{q})\big)\,\mathrm{Var}\big(\ell_j(\boldsymbol{q})\big)}} \;=\; \frac{{\boldsymbol{k}_i}^\top \boldsymbol{\Sigma}_{\mathrm{q}} \boldsymbol{k}_j}{\sqrt{{\boldsymbol{k}_i}^\top \boldsymbol{\Sigma}_{\mathrm{q}} \boldsymbol{k}_i}\,\sqrt{{\boldsymbol{k}_j}^\top \boldsymbol{\Sigma}_{\mathrm{q}} \boldsymbol{k}_j}} \;=\; \cos\big(\widetilde{\boldsymbol{k}}_i, \widetilde{\boldsymbol{k}}_j\big), \qquad \widetilde{\boldsymbol{k}}_i := {\boldsymbol{\Sigma}_{\mathrm{q}}}^{1/2} \boldsymbol{k}_i. \tag{2}$$

where the expectations are taken over $\boldsymbol{q}$. The co-retrieval correlation is thus a cosine similarity with covariance-weighted keys, which measures angles in the geometry induced by $\boldsymbol{\Sigma}_{\mathrm{q}}$. Setting $\boldsymbol{\Sigma}_{\mathrm{q}} = \boldsymbol{I}$ recovers the plain cosine similarity adopted in prior work (Liu et al., 2025a; Wu et al., 2026), grouping keys by cosine similarity. *When queries are not isotropic, the two measures can disagree in their ordering: a pair of keys with high cosine similarity in the raw space may be co-retrieved far less often than a pair that appears distant, and grouping by raw similarity therefore misplaces keys.*

To validate this, we estimate an empirical covariance $\widehat{\boldsymbol{\Sigma}_{\mathrm{q}}}$ from the prefill queries of the first 16k tokens of each SlimPajama document, and measure how well each similarity predicts the logit correlation observed on a held-out set of queries. Figure 2 shows that the covariance-weighted similarity is substantially more accurate than the plain cosine similarity (see Appendix C for details).

While the correlation captures whether two keys tend to receive high or low logits together, it discards the magnitude of these fluctuations. To derive a partition objective, we therefore also consider the corresponding logit covariance.

$$d\,\mathrm{Cov}\big(\ell_i(\boldsymbol{q}), \ell_j(\boldsymbol{q})\big) = {\boldsymbol{k}_i}^\top \boldsymbol{\Sigma}_{\mathrm{q}} \boldsymbol{k}_j \;=\; \langle \widetilde{\boldsymbol{k}}_i, \widetilde{\boldsymbol{k}}_j \rangle. \tag{3}$$

Appendix C.1 evaluates the agreement between predicted and empirical logit covariances. We next derive a partition objective from these logit relationships.

3.2 Partitioning the Cache by Co-Retrieval

Partition. We partition the KV cache into M pages $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_M\}$, each tagged by its key centroid $\boldsymbol{c}_p = |\mathcal{P}_p|^{-1} \sum_{i \in \mathcal{P}_p} \boldsymbol{k}_i$. At each decoding step, attention is restricted to the pages with the highest centroid scores under a fixed KV-cache budget, while unselected pages remain resident in HBM but are not accessed.

Partition objective. Because a page is retrieved as a whole, its members should be retrieved together. We therefore seek a partition in which each member's logit fluctuates together with its page centroid's logit, minimizing the variance of their difference across queries. For $\boldsymbol{k}_i \in \mathcal{P}_p$, write its logit as

$$\ell_{ti} = \underbrace{\frac{\boldsymbol{q}_t^T \boldsymbol{c}_p}{\sqrt{d}}}_{g_{tp}} + \underbrace{\frac{\boldsymbol{q}_t^T (\boldsymbol{k}_i - \boldsymbol{c}_p)}{\sqrt{d}}}_{\delta_{ti}}, \tag{4}$$

where g_{tp} is the page score and δ_{ti} is the member's deviation from it. Under the query model in Equation (2), the deviation has variance

$$\mathrm{Var}_t(\delta_{ti}) = \frac{(\boldsymbol{k}_i - \boldsymbol{c}_p)^T \boldsymbol{\Sigma}_{\mathrm{q}} (\boldsymbol{k}_i - \boldsymbol{c}_p)}{d} = \frac{\|\widetilde{\boldsymbol{k}}_i - \widetilde{\boldsymbol{c}}_p\|^2}{d}, \tag{5}$$

where $\widetilde{\boldsymbol{k}}_i = {\boldsymbol{\Sigma}_{\mathrm{q}}}^{1/2} \boldsymbol{k}_i$ and $\widetilde{\boldsymbol{c}}_p = {\boldsymbol{\Sigma}_{\mathrm{q}}}^{1/2} \boldsymbol{c}_p$. Since the transformation is linear, $\widetilde{\boldsymbol{c}}_p$ is the mean of the transformed keys in page p. Summing the variances over all pages gives the partition objective

$$J(\mathcal{P}) := d \sum_{p=1}^{M} \sum_{i \in \mathcal{P}_p} \mathrm{Var}_t(\delta_{ti}) = \sum_{p=1}^{M} \sum_{i \in \mathcal{P}_p} \|\widetilde{\boldsymbol{k}}_i - \widetilde{\boldsymbol{c}}_p\|^2. \tag{6}$$

We therefore apply **Euclidean k-means** to the transformed keys $\widetilde{\boldsymbol{k}}_i$ to minimize this objective. The objective also admits a pairwise interpretation,

$$\sum_{i \in \mathcal{P}_p} \|\widetilde{\boldsymbol{k}}_i - \widetilde{\boldsymbol{c}}_p\|^2 = \frac{1}{2|\mathcal{P}_p|} \sum_{i,j \in \mathcal{P}_p} \|\widetilde{\boldsymbol{k}}_i - \widetilde{\boldsymbol{k}}_j\|^2 = \frac{d}{2|\mathcal{P}_p|} \sum_{i,j \in \mathcal{P}_p} \mathrm{Var}_t\big(\ell_{ti} - \ell_{tj}\big), \tag{7}$$

so minimizing the objective reduces the variance of logit differences between keys in the same page, promoting co-retrieval. We analyze its overhead in Figure 7.

Logit correlation versus covariance. Expanding the pairwise variance in Equation (7) gives $d\,\mathrm{Var}_t(\ell_{ti} - \ell_{tj}) = \sigma_i^2 + \sigma_j^2 - 2\sigma_i\sigma_j\rho_{ij}$, where $\sigma_i^2 = \boldsymbol{k}_i{}^T\boldsymbol{\Sigma}_{\mathrm{q}}\boldsymbol{k}_i$. Thus, the objective accounts for both logit correlation and fluctuation scale. Partition optimizing logit correlation can be obtained using **Spherical k-means** on normalized transformed keys. In practice, we observe optimizing logit covariance performs better empirically and is our default. Appendix E.2 details the ablation.

Although the partition minimizes within-page spread, a finite number of pages generally leaves nonzero variation. We next derive a centroid-score correction that accounts for this residual spread.

3.3 Spread-Corrected Page Scoring

The centroid logit, $g_{tp} = \frac{\boldsymbol{q}_t^T\boldsymbol{c}_p}{\sqrt{d}}$, provides a natural score for page p. However, the centroid does not capture the variation of the individual keys within the page. To quantify the effect of within-page variation on page retrieval, we consider the average unnormalized attention mass of page p,

$$U_t(p) := \frac{1}{|\mathcal{P}_p|}\sum_{i\in\mathcal{P}_p}\exp(\ell_{ti}) = \exp(g_{tp})\frac{1}{|\mathcal{P}_p|}\sum_{i\in\mathcal{P}_p}\exp(\delta_{ti}), \tag{8}$$

The centroid is the mean of the keys, $\sum_{i\in\mathcal{P}_p}\delta_{ti} = 0$ for all possible query. Applying Jensen's inequality:

$$\frac{1}{|\mathcal{P}_p|}\sum_{i\in\mathcal{P}_p}\exp(\delta_{ti}) \geq \exp\left(\frac{1}{|\mathcal{P}_p|}\sum_{i\in\mathcal{P}_p}\delta_{ti}\right) = 1 \tag{9}$$

If we estimate a page's mass using only the centroid, this mass would be $\exp(g_{tp})$, which is a smaller estimation than the real $U_t(p)$. To estimate this factor without scoring individual keys, we model each member's deviation across the query distribution as $\delta_{ti} \sim \mathcal{N}(0, {s_i}^2)$, where ${s_i}^2 = \mathrm{Var}_t(\delta_{ti})$. Then

$$\mathbb{E}_t[\exp(\delta_{ti})] = \exp\left(\frac{{s_i}^2}{2}\right), \qquad {s_i}^2 = \frac{(\boldsymbol{k}_i - \boldsymbol{c}_p)^T\boldsymbol{\Sigma}_{\mathrm{q}}(\boldsymbol{k}_i - \boldsymbol{c}_p)}{d}. \tag{10}$$

Approximating the individual spreads by their page mean,

$${s_p}^2 = \frac{1}{|\mathcal{P}_p|}\sum_{i\in\mathcal{P}_p}{s_i}^2 = \frac{1}{|\mathcal{P}_p|}\sum_{i\in\mathcal{P}_p}\frac{\|\widetilde{\boldsymbol{k}}_i - \widetilde{\boldsymbol{c}}_p\|^2}{d}, \tag{11}$$

and substituting back to $U_t(p)$ gives

$$\log U_t(p) \approx g_{tp} + \frac{{s_p}^2}{2}. \tag{12}$$

This gives the spread-corrected gate

$$\mathrm{score}_t(p) = \frac{\boldsymbol{q}_t^T\boldsymbol{c}_p}{\sqrt{d}} + \frac{{s_p}^2}{2}. \tag{13}$$

The spread correction complements the partition objective: k-means minimizes within-page spread, while the gate accounts for residual spread that remains under a finite page budget. The spread is computed from the resulting partition with minimal overhead. In practice, we use a linear bias, $\kappa' s_p$, which is less sensitive to coefficient choice than the quadratic form. See Appendix E.2 for details.

3.4 Estimating the Query Covariance

The partition must be fixed before the decoding queries are generated, hence $\boldsymbol{\Sigma}_{\mathrm{q}}$ must be estimated from queries available beforehand. Prior work models the queries of a head as Gaussian and estimates its moments from the prefill queries of the same head (Devoto et al., 2025; Liu et al., 2025b). Figure 2 shows that covariance estimated from prefill queries predicts the co-retrieval structure of held-out decoding queries.

In a long-context downstream task, however, the prefill and decoding queries need not come from the same distribution. The prefill queries are dominated by the document, whereas the decoding queries are generated in response to the downstream instruction. Their covariances can therefore differ, in which case the prefill estimate induces the wrong geometry for both the partition and the gate. This section quantifies the resulting gap and the extent to which it can be closed.

Metrics. We separate two questions: how well an estimator $\widehat{\boldsymbol{\Sigma}}$ approximates the covariance of the decoding queries characterized by *covariance agreement* , and how much its error affects *co-retrieval agreement* . Let $\boldsymbol{\Sigma}^\star$ denote the *reference* covariance, computed from the decoding queries themselves. *covariance agreement*

$$\mathcal{A}(\widehat{\boldsymbol{\Sigma}}) \;=\; \frac{\langle \widehat{\boldsymbol{\Sigma}}, \boldsymbol{\Sigma}^\star \rangle_F}{\|\widehat{\boldsymbol{\Sigma}}\|_F \, \|\boldsymbol{\Sigma}^\star\|_F}, \tag{14}$$

is the cosine similarity between the two covariance matrices under the Frobenius inner product. This metric is invariant to overall scale, as is the resulting partition.

To compute the *co-retrieval agreement*, for each head, we use $\widehat{\boldsymbol{\Sigma}}$ in place of $\boldsymbol{\Sigma}_{\mathrm{q}}$ in Equation (2) to obtain predicted correlations $\hat{\rho}_{ij}$. We then compute the empirical correlations of the logits ℓ_{ti} and ℓ_{tj} across the decoding queries and report the Pearson correlation between the predicted and empirical correlations over all key pairs. This measures how well the estimated covariance captures the pairwise logit relationships on which co-retrieval depends. An estimator can therefore have high covariance agreement yet low co-retrieval agreement if its error is concentrated in directions along which the keys vary, and vice versa.

Estimators. We compare two estimators against $\boldsymbol{\Sigma}^\star$. *Prefill* follows prior work and uses the sample covariance of all prefill queries for the head, centered by their mean. *Query-mix* separates queries produced by the document from those produced by the question or downstream instruction that follows it, and forms the convex combination

$$\widehat{\boldsymbol{\Sigma}}_\lambda \;=\; (1-\lambda)\,\boldsymbol{\Sigma}_{\mathrm{ctx}} + \lambda\,\boldsymbol{\Sigma}_{\mathrm{qst}}, \qquad \lambda \in [0,1], \tag{15}$$

Before mixing, we rescale the question covariance to match the trace of the context covariance, so that λ controls the relative contribution of the two covariance structures. We evaluate them in Section 4.2.

GQA. Under multi-head attention, each query head has its own keys, so the partition is constructed independently for each head using its covariance $\boldsymbol{\Sigma}_{\mathrm{q}}$. Grouped query attention instead shares a single set of keys among G query heads, each with its own covariance $\boldsymbol{\Sigma}_{\mathrm{q}}{}^{(g)}$. The group therefore also needs to share a single partition to preserve bandwidth savings.

For each query head, the covariance induces a distance between keys through the transformed-key geometry of Equation (3). We combine these head-specific geometries by averaging their pairwise squared distances, which is equivalent to using the average covariance.

$$(\boldsymbol{k}_i - \boldsymbol{k}_j)^\top \overline{\boldsymbol{\Sigma}}\,(\boldsymbol{k}_i - \boldsymbol{k}_j) \;=\; \frac{1}{G}\sum_{g=1}^{G} (\boldsymbol{k}_i - \boldsymbol{k}_j)^\top \boldsymbol{\Sigma}_{\mathrm{q}}{}^{(g)}\,(\boldsymbol{k}_i - \boldsymbol{k}_j), \qquad \overline{\boldsymbol{\Sigma}} = \frac{1}{G}\sum_{g=1}^{G} \boldsymbol{\Sigma}_{\mathrm{q}}{}^{(g)}, \tag{16}$$

We therefore construct a single partition by running k-means under $\overline{\boldsymbol{\Sigma}}$, clustering keys according to the geometry shared across the query heads. Algorithms 1 and 2 state the procedure. We assess the accuracy cost of sharing a partition across query heads in Appendix E.1.

4 Experiments

4.1 Setup

Baselines We compare TesseraKV against two families of KV-cache methods. Eviction methods discard keys during pre-fill and cannot recover them, including SnapKV (Li et al., 2024), KeyDiff (Park et al., 2026) and AgentKV (Liu et al., 2026) . Retrieval methods retain the full cache and select a subset at each decoding step, including Quest (Tang et al., 2024), ArkVale (Chen et al., 2024) and LouisKV (Wu et al., 2026). All baselines are run at the same key budget and the same average page size $c = 64$. More details about the setup are available in Appendix D.

Models and benchmarks We evaluate Muse-Glimmer-30B (Meta Superintelligence Lab, 2026), North-Mini-Code-30B (Cohere & Cohere Labs, 2026), and Command-R-7B (Cohere et al., 2025), whose full-attention layers have no positional encoding and whose GQA group widths are 16, 8, 4, respectively. Paging applies only to these layers with sliding-window layers remain unchanged.

Table 1: TesseraKV outperforms majority of the baselines on RULER with 8K and 16K context length. We show average score with decreasing KV budget across baselines. FullCache denotes attention over the entire KV cache. Best results for each compression ratio are displayed in bold.

Model		RULER-8K						RULER-16K					
	KV budget B	16	32	64	128	256	Avg.	16	32	64	128	256	Avg.
Muse-Glimmer-30B	FullCache			94.61			94.61			94.52			94.52
	KeyDiff	17.50	22.32	27.34	34.41	43.82	29.08	16.06	21.01	27.35	32.54	40.20	27.43
	Quest	20.13	24.78	31.92	43.93	59.44	36.04	16.41	20.68	27.02	37.37	51.37	30.57
	ArkVale	20.72	26.29	35.12	47.87	63.30	38.66	16.58	20.94	28.14	38.21	53.43	31.46
	LouisKV	39.27	56.79	74.58	86.36	93.91	70.18	35.39	48.11	61.12	81.27	91.19	63.42
	TesseraKV	**44.38**	**68.94**	**84.53**	**92.45**	**94.40**	**76.94**	**41.70**	**65.52**	**80.54**	**90.51**	**93.79**	**74.41**
North-Mini-Code-30B	FullCache			94.62			94.62			91.19			91.19
	KeyDiff	9.50	14.86	31.25	51.44	60.20	33.45	8.28	8.70	27.09	45.46	56.99	29.30
	Quest	12.23	20.18	35.48	48.02	64.83	36.15	12.12	19.14	32.83	44.16	57.62	33.17
	ArkVale	12.82	20.28	34.52	52.50	68.66	37.75	12.61	18.21	30.54	46.49	61.10	33.79
	LouisKV	23.36	51.06	74.75	84.93	91.62	65.14	14.53	20.26	61.74	79.18	86.72	52.49
	TesseraKV	**34.85**	**60.72**	**76.72**	**88.19**	**92.70**	**70.64**	**36.43**	**57.83**	**72.45**	**83.66**	**89.26**	**67.93**
Command-R-7B	FullCache			89.64			89.64			85.95			85.95
	KeyDiff	12.15	17.88	29.10	48.90	58.79	33.36	10.04	15.82	22.58	40.64	54.72	28.76
	Quest	12.91	19.26	35.71	47.24	57.09	34.44	9.98	14.82	24.84	37.16	49.26	27.21
	ArkVale	12.12	17.55	27.46	45.09	58.01	32.05	9.88	14.90	20.29	33.43	46.95	25.09
	LouisKV	**36.24**	54.30	72.24	83.47	**88.18**	66.88	26.59	39.64	60.75	74.00	83.81	56.96
	TesseraKV	35.60	**56.99**	**74.32**	**85.09**	88.13	**68.03**	**33.09**	**53.15**	**67.66**	**79.84**	**84.17**	**63.58**

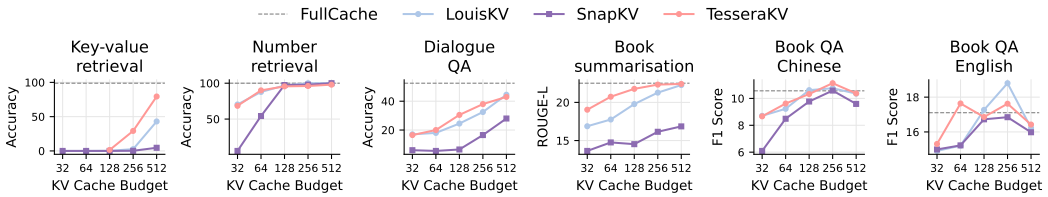

Figure 3: Performance comparison of TesseraKV with LouisKV, SnapKV, and full-cache attention across KV-cache budgets on INFINITEBENCH using Muse-Glimmer-30B.

For long-context tasks, we use RULER (Hsieh et al., 2024) at 8K and 16K. Its 13 synthetic tasks span needle retrieval, variable tracing, and question answering. To complement these synthetic tasks, we evaluate on INFINITEBENCH (Zhang et al., 2024b), whose contexts average over 100K tokens, and include novels and dialogue scripts. For agentic workflows, we use 155 text-agent tasks from CLAW-EVAL (Ye et al., 2026), spanning communication, web search, finance, productivity.

4.2 COVARIANCE ESTIMATION ACCURACY

We ask whether queries available at partition time reliably proxy the decoding distribution, and whether the answer depends on the task. We consider two regimes: a balanced subset of RULER, which retrieves from synthetic contexts, and INFINITEBENCH book summarisation, which reads natural prose. For each, we compute the reference covariance $\mathbf{\Sigma}^\star$ from decoding queries and compare it with three pre-decoding $\widehat{\mathbf{\Sigma}}$ estimators: *Prefill*, *query-mix* with λ, and *identity* ($\boldsymbol{I}$) – *no covariance*. See details in Appendix D.1.

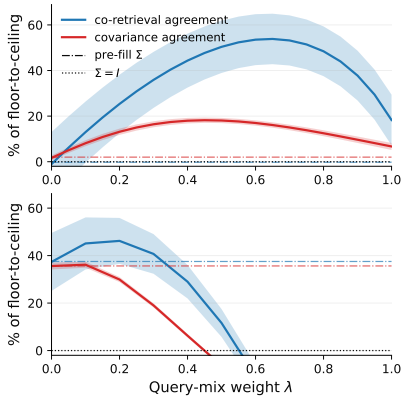

Figure 4: Both metrics against the query-mix weight λ, as a share of the range from $\mathbf{\Sigma}_q = \boldsymbol{I}$ to the reference $\mathbf{\Sigma}^\star$. Signal carried by flips with the task: pre-fill sits on the floor for RULER (top) and up for INFINITE BENCH (bottom).

On RULER, *query-mix* improves both metrics, although they favor different query-mix weights. Co-retrieval agreement recovers 57.3% of the distance from $\boldsymbol{I}$ (0.453) to the reference (0.579), peaking at $\lambda = 0.65$; $\mathcal{A}$ recovers 18.2% of the corresponding distance from $\boldsymbol{I}$ (0.383) to the reference (1), peaking at $\lambda = 0.45$. Neither endpoint helps: the context-only estimate performs slightly below the identity, while the question-only estimate falls back toward it. The prefill estimator is likewise indistinguishable from the identity, indicating that the gain

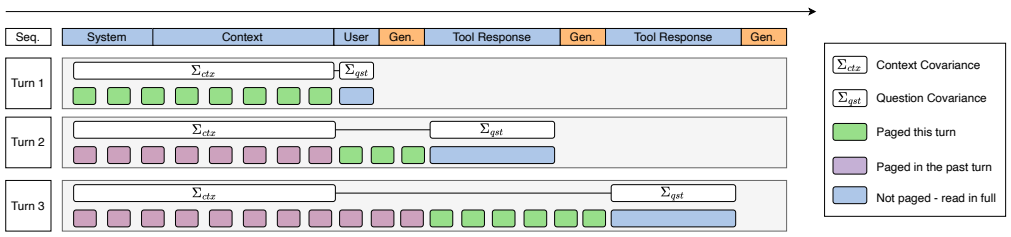

Figure 5: TesseraKV across the turns of an agentic workload. Each turn partitions only the tokens appended since the last turn and keeps the pages earlier turns built; its own question stays unpaged and is read in full. Σ_{ctx} is estimated once from the opening context, Σ_q once per turn.

Table 2: Evaluation on CLAW-EVAL using a 131,072-token context window across 155 agentic tasks, with three trials per task on Muse-Glimmer-30B. A task is considered successful only if all three trials exceed the rubric threshold. Results report task success rates by difficulty. KV budget B denotes the number of cached tokens retrieved per KV head at each decoding step. *Overall* is task-weighted performance (Easy: 69, Medium: 46, Hard: 40). Bold indicates the best result in each column.

	Easy				Medium				Hard				Overall			
KV budget B	64	128	256	512	64	128	256	512	64	128	256	512	64	128	256	512
FullCache	73.91				86.96				70.00				76.77			
Arkvale	24.64	42.03	66.67	72.46	2.17	34.78	69.57	80.43	5.00	17.50	57.50	60.00	12.90	33.55	65.16	71.61
LouisKV	28.99	47.83	**73.91**	69.57	17.39	45.65	73.91	**84.78**	10.00	30.00	45.00	62.50	20.65	42.58	66.45	72.26
TesseraKV	**59.42**	**71.01**	**73.91**	**75.36**	**63.04**	**82.61**	**80.43**	82.61	**42.50**	**55.00**	**62.50**	**65.00**	**56.13**	**70.32**	**72.90**	**74.84**

comes from combining the two query populations. On INFINITEBENCH, the pattern reverses. The context-only estimate already recovers 37% of the distance to the reference. Adding question queries yields a further improvement, reaching 50% at $\lambda = 0.2$, but becomes harmful thereafter; at $\lambda = 0.6$, performance falls below the identity.

This contrast reflects the different roles of the context and question in the two tasks. In RULER, the question determines what needs to be retrieved, so question queries carry the relevant signal while the synthetic context contributes little. In book summarisation, the decoding queries are directed at the document itself, so the context carries much more of the relevant signal. The two metrics also need not move together: an estimator can be a poor approximation of the full covariance while still capturing the geometry relevant to partitioning if its error lies primarily in directions along which the keys vary little. The same analysis across additional tasks is reported in Appendix E.4.

4.3 MAIN RESULT

This section presents the main evaluation results of TesseraKV across long-context retrieval and understanding tasks, followed by agentic workloads. We conclude with profiling results that quantify the hardware efficiency gains and the overhead of estimating the query covariance.

Long-Context Tasks In Table 1, we compare TesseraKV against eviction and retrieval baselines on RULER-8K and 16K. Retrieval methods including Quest, Arkvale, and LouisKV, generally outperform eviction methods like KeyDiff, showing that permanently discarding KV entries can degrade accuracy.

Notably, LouisKV groups keys by cosine similarity, corresponding to the identity (no-covariance) geometry considered in our analysis. TesseraKV instead uses the query covariance to adapt the key geometry to the co-retrieval structure of the head. It outperforms all baselines in most settings. On RULER-16K with Muse-Glimmer-30B, TesseraKV achieves an average accuracy of 74.41, improving over LouisKV's 63.42 by 10.99 points, averaged across KV budgets. We present the INFINITEBENCH results in Figure 3. TesseraKV matches or outperforms LouisKV in six tasks at almost all budgets, with the largest gains on book summarisation. This result is consistent with our covariance analysis: for natural-language documents, the context itself provides a strong signal for the decoding-query geometry, making it useful for constructing the partition.

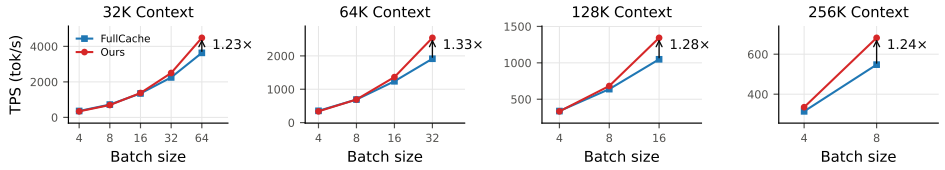

Figure 6: Decoding throughput against batch size at four context lengths up to device memory limit.

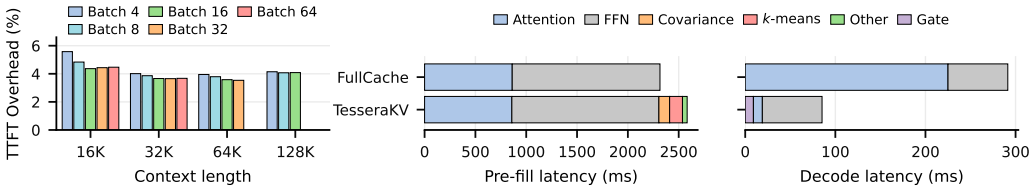

Figure 7: Time-to-first-token overhead and profiled latency components of TesseraKV relative to full-cache attention. *Left:* TTFT overhead across context lengths and batch sizes. *Middle and right:* prefill and decode component times of one global attention layer at 64k context and batch 32. Decode times are summed over 512 steps. Component times are summed without asynchronous overlap.

Agentic Workloads We evaluate on CLAW-EVAL in Table 2, extending TesseraKV to multi-turn tasks by estimating Σ_{ctx} once from the opening context and re-estimating only Σ_q each turn. We partition only newly appended tokens while retaining pages from earlier turns, avoiding re-partitioning the full context as shown in Figure 5. TesseraKV substantially outperforms ArkVale and LouisKV at tight KV-cache budgets, achieving an overall success rate of 70.32 versus 33.55 and 42.58 at $B = 128$, respectively. From 128 to 512, its performance remains relatively stable, with only a small gap to full cache, while both ArkVale and LouisKV degrade more substantially. At 512, the methods narrow the gap, although Hard tasks remain below full-cache accuracy. Comparisons with full repartitioning and AgentKV are reported in Appendix F.

Inference Throughput Figure 6 reports decoding throughput across batch sizes and context lengths for 512 output tokens. At the largest batch size supported by both configurations on a B200 GPU with SGLang and FlashAttention 4 , TesseraKV achieves 1.23–1.33× full-cache throughput across 32k–256k contexts. At batch size 8, TesseraKV retains 97.4% of its throughput as context grows from 8k to 256k, whereas full-cache throughput falls by 26.5%. Our implementation reduces paging overhead through batched matrix multiplication for page scoring, fused Triton kernels for page selection and address packing, and sparse attention. Implementation details are provided in Appendix G.1.

The left panel of Figure 7 reports TTFT. Relative to full-cache attention, TesseraKV adds around 3.5–4.0% from 32k to 128k, with little variation across batch sizes. The remaining panels profile one layer with global attention at 64k context and batch size 32. When component times are summed without asynchronous overlap, page selection and sparse attention together take one-twelfth the time of full attention during decoding. During prefill, covariance estimation accounts for 4% of the summed component time. Additional breakdowns appear in Appendix G.2.

4.4 ABLATION STUDY

To isolate the contributions of the covariance-based partition and gate, we cross three partition choices (positional, $\mathbf{\Sigma}_{\mathrm{q}} = \boldsymbol{I}$, and the *query-mix*) with three gates (none, the spread measured under $\boldsymbol{I}$, and under $\mathbf{\Sigma}_{\mathrm{q}}$) in Appendix E.3. The covariance-based setting generally performs best in both components and jointly. We further ablate sharing a partition across GQA groups, the form and weight of the spread correction, and the query-mix weight across workloads. Details are provided and Appendix E.

5 CONCLUSION

We propose TesseraKV, a KV paging method that formulates partitioning and page scoring around the co-retrieval structure of attention. TesseraKV reduces KV bandwidth requirements during decoding for long-context and agentic workloads, while achieving state-of-the-art performance.

AI DISCLOSURE

In this work, we used generative AI tools to assist with polishing the writing and editing code development, and launching experiments at large scale. We did not use generative AI tools to generate research ideas or synthetic datasets, or to formulate mathematical claims. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, 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.

GLM-5-Team at al. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Renze Chen, Zhuofeng Wang, Beiquan Cao, Tong Wu, Size Zheng, Xiuhong Li, Xuechao Wei, Shengen Yan, Meng Li, and Yun Liang. Arkvale: Efficient generative llm inference with recallable key-value eviction. *Advances in Neural Information Processing Systems*, 37:113134–113155, 2024.

Cohere and Cohere Labs. North Mini Code 1.0. Hugging Face model repository, June 2026. URL https://huggingface.co/CohereLabs/North-Mini-Code-1.0. Apache-2.0. Accessed 8 September 2026.

Team Cohere, Arash Ahmadian, Marwan Ahmed, Jay Alammar, Milad Alizadeh, Yazeed Alnumay, Sophia Althammer, Arkady Arkhangorodsky, Viraat Aryabumi, Dennis Aumiller, et al. Command a: An enterprise-ready large language model. *arXiv preprint arXiv:2504.00698*, 2025.

Debajyoti Datta, Trishala Neeraj, Bibek Paudel, Vyom Sharma, and Subhabrata Mukherjee. Manifoldkv: Training-free kv cache compression via euclidean outlier detection. *arXiv preprint arXiv:2602.08343*, 2026.

Alessio Devoto, Maximilian Jeblick, and Simon Jégou. Expected attention: Kv cache compression by estimating attention from future queries distribution. *arXiv preprint arXiv:2510.00636*, 2025.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Yoav Gelberg, Koshi Eguchi, Takuya Akiba, and Edoardo Cetin. Extending the context of pre-trained llms by dropping their positional embedding. In *International Conference on Learning Representations*, volume 2026, pp. 60170–60197, 2026.

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? *arXiv preprint arXiv:2404.06654*, 2024.

Tao Ji, Bin Guo, Yuanbin Wu, Qipeng Guo, Lixing Shen, Zhan Chen, Xipeng Qiu, Qi Zhang, and Tao Gui. Towards economical inference: Enabling deepseek's multi-head latent attention in any transformer-based llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33313–33328, 2025.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

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. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

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 on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6: 87–100, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 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, 2025a.

James Liu, Pragaash Ponnusamy, Tianle Cai, Yoon Kim, Ben Athiwaratkun, et al. Training-free activation sparsity in large language models. In *International Conference on Learning Representations*, volume 2025, pp. 98302–98322, 2025b.

Taowen Tony Liu, Jeffrey TH Wong, Can Xiao, Bowen Yang, Hao Mark Chen, and Yiren Zhao. Agentkv: Phase-aware kv eviction for agentic llms. *arXiv preprint arXiv:2609.14872*, 2026.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026.

Meta Superintelligence Lab. Muse Glimmer 30B. Hugging Face model repository, August 2026. URL `https://huggingface.co/meta-models/Muse-Glimmer-30B`. Apache-2.0. Accessed 8 September 2026.

Lorenz K Muller, Philippe Bich, Chiara Boretti, Hyun-Min Chang, Jiawei Zhuang, and Lukas Cavigelli. Kvarn: Variance-normalized kv-cache quantization mitigates error accumulation in reasoning tasks. *arXiv preprint arXiv:2606.03458*, 2026.

Junyoung Park, Dalton Jones, Matthew Morse, Raghavv Goel, Mingu Lee, and Christopher Lott. Keydiff: Key similarity-based kv cache eviction for long-context llm inference in resource-constrained environments. *Advances in Neural Information Processing Systems*, 38:5983–6019, 2026.

Qwen Team. Qwen3.8-Flash-Next: A new architecture, towards ultimate cost-efficiency, August 2026. URL `https://qwen.ai/blog?id=qwen3.8-flash-next`.

Valeria Ruscio, Umberto Nanni, and Fabrizio Silvestri. What are you sinking? a geometric approach on attention sink. *Advances in Neural Information Processing Systems*, 38:123482–123521, 2026.

ShareGPT. Sharegpt vicuna unfiltered. `https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered`, 2023. Dataset.

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

Zhiqiang Shen, Tianhua Tao, Liqun Ma, Willie Neiswanger, Zhengzhong Liu, Hongyi Wang, Bowen Tan, Joel Hestness, Natalia Vassilieva, Daria Soboleva, et al. Slimpajama-dc: Understanding data combinations for llm training. *arXiv preprint arXiv:2309.10818*, 2023.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations*, volume 2025, pp. 19299–19319, 2025.

Tianxin Wei, Ting-Wei Li, Zhining Liu, Xuying Ning, Ze Yang, Jiaru Zou, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Dongqi Fu, et al. Agentic reasoning for large language models. *arXiv preprint arXiv:2601.12538*, 2026.

Jeffrey TH Wong, Cheng Zhang, Xinye Cao, Pedro Gimenes, Christos-Savvas Bouganis, George A Constantinides, Wayne Luk, and Yiren Zhao. A3: an analytical low-rank approximation framework for attention. *arXiv preprint arXiv:2505.12942*, 2025a.

Jeffrey TH Wong, Cheng Zhang, Louis Mahon, Wayne Luk, Anton Isopoussu, and Yiren Zhao. On the existence and behavior of secondary attention sinks. *arXiv preprint arXiv:2512.22213*, 2025b.

Jeffrey TH Wong, Zixi Zhang, Junyi Liu, and Yiren Zhao. Team of thoughts: Efficient test-time scaling of agentic systems through orchestrated tool calling. *arXiv preprint arXiv:2602.16485*, 2026.

Wenbo Wu, Qingyi Si, Xiurui Pan, Ye Wang, and Jie Zhang. Louiskv: Efficient kv cache retrieval for long input-output sequences. In *International Conference on Learning Representations*, volume 2026, pp. 65401–65421, 2026.

Can Xiao, Sukmin Cho, Junbong We, Zhixiong Niu, Jianyi Cheng, Yiren Zhao, Youngjin Kwon, Yongqiang Xiong, Rui Ma, and Junyi Liu. Oasiskv: Scaling in-decode kv cache beyond hbm with lookahead sparse prefetching. *arXiv preprint arXiv:2608.08097*, 2026.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

Jiawei Xu, Arief Koesdwiady, Sisong Bei, Yan Han, Baixiang Huang, Dakuo Wang, Yutong Chen, Zheshen Wang, Peihao Wang, Pan Li, et al. Rethinking the value of multi-agent workflow: A strong single agent baseline. *arXiv preprint arXiv:2601.12307*, 2026.

Bowen Yang, Bharat Venkitesh, Dwaraknath Gnaneshwar Talupuru, Hangyu Lin, David Cairuz, Phil Blunsom, and Acyr Locatelli. Rope to nope and back again: A new hybrid attention strategy. *Advances in Neural Information Processing Systems*, 38:64133–64157, 2026.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, volume 2025, pp. 29687–29707, 2025.

Bowen Ye, Rang Li, Qibin Yang, Yuanxin Liu, Linli Yao, Hanglong Lv, Zhihui Xie, Chenxin An, Lei Li, Lingpeng Kong, et al. Claw-eval: Towards trustworthy evaluation of autonomous agents. *arXiv preprint arXiv:2604.06132*, 2026.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. Turboquant: Online vector quantization with near-optimal distortion rate. In *International Conference on Learning Representations*, volume 2026, pp. 56418–56439, 2026.

Cheng Zhang, Jianyi Cheng, George A Constantinides, and Yiren Zhao. Lqer: Low-rank quantization error reconstruction for llms. *arXiv preprint arXiv:2402.02446*, 2024a.

Cheng Zhang, Jeffrey TH Wong, Can Xiao, George Constantinides, and Yiren Zhao. Qera: an analytical framework for quantization error reconstruction. In *International Conference on Learning Representations*, volume 2025, pp. 12531–12560, 2025.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, et al. ∞ bench: Extending long context evaluation beyond 100k tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15262–15277, 2024b.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö Arık. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237, 2024c.

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.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, et al. Latent collaboration in multi-agent systems. *arXiv preprint arXiv:2511.20639*, 2025.

A NOTATION

Table 3 collects the vectors, matrices and sets used in this paper, and Table 4 the dimensions and scalars.

Table 3: Notation of vectors, matrices, mappings, and sets in this paper.

Notation	Description
$\boldsymbol{q}$	A decoding query of a head; $\boldsymbol{q}_t$ is the query of step t
$\boldsymbol{k}_i$	Key of cached position i
$\boldsymbol{c}_p$	Centroid of page p, the mean of its keys in the raw key space
$\widetilde{\boldsymbol{k}}_i$	Transformed key, $\widetilde{\boldsymbol{k}}_i = {\boldsymbol{\Sigma}_{\mathrm{q}}}^{1/2}\boldsymbol{k}_i$
$\widetilde{\boldsymbol{c}}_p$	Transformed centroid, also the mean of $\{\widetilde{\boldsymbol{k}}_i\}_{i\in\mathcal{P}_p}$
$\boldsymbol{\Sigma}_{\mathrm{q}}$	Query covariance of a head, Section 3.1
${\boldsymbol{\Sigma}_{\mathrm{q}}}^{1/2}$	Unique symmetric positive semidefinite square root of $\boldsymbol{\Sigma}_{\mathrm{q}}$
ϕ	Key transform $\phi(\boldsymbol{k}) = {\boldsymbol{\Sigma}_{\mathrm{q}}}^{1/2}\boldsymbol{k}$; uses $\overline{\boldsymbol{\Sigma}}^{1/2}$ for GQA
$\boldsymbol{\Sigma}_{\mathrm{ctx}}$	Covariance of the queries produced by the context block
$\boldsymbol{\Sigma}_{\mathrm{qst}}$	Covariance of the queries produced by the question block
$\widehat{\boldsymbol{\Sigma}}_{\lambda}$	Query-mix estimator of $\boldsymbol{\Sigma}_{\mathrm{q}}$, Equation (15)
$\boldsymbol{\Sigma}^{\star}$	Reference covariance estimated from the decoding queries
$\overline{\boldsymbol{\Sigma}}$	Covariance pooled over a KV group, one per KV head, Equation (16)
$\boldsymbol{I}$	Identity matrix; $\boldsymbol{\Sigma}_{\mathrm{q}} = \boldsymbol{I}$ recovers plain cosine similarity
π	Partition map, from a key index to a page index
$\mathcal{P}$	The partition $\{\mathcal{P}_1, \ldots, \mathcal{P}_M\}$
$\mathcal{P}_p$	Page p, the set of key indices it holds
$\mathcal{R}$	Resident keys, the first n_{s} positions
$\mathcal{C}$	Clustered keys, the positions the partition is built over
$\mathcal{O}$	Open tail: the question and what has been generated, always read
$\mathcal{S}$	Key slots admitted by the budget walk at the current step
$\mathcal{W}$	Working set actually attended over, $\mathcal{R} \cup \mathcal{S} \cup \mathcal{O}$

Table 4: Notation of dimensions and scalars in this paper.

Notation	Description
d	Head dimension, shared by the query and the key
n	Context keys per layer and KV head
G	GQA group width: query heads per KV head, indexed by g
M	Number of pages the clustered keys are partitioned into
c	Target page size, $M = \max(\lceil \lvert\mathcal{C}\rvert / c \rceil, 1)$
B	Retrieval budget for clustered keys per KV head per decoding step
n_{s}	Length of the resident prefix held as attention sinks
λ	Query-mix weight of Equation (15)
κ'	Coefficient of the linear spread correction $\kappa' s_p$
ℓ_{ti}	Attention logit of key i under the query of step t
a_{ti}	Normalised attention score of key i at step t
g_{tp}	Centroid logit of page p under query t, Equation (4)
δ_{ti}	Deviation of member i from its page's centroid score, Equation (4)
$U_t(p)$	Average unnormalized attention mass of page p at step t
${s_p}^2$	Mean within-page logit residual variance, Equation (11)
s_p	Square root of the mean residual variance used in the linear gate
γ_p	Page score aggregated across the query heads of a KV group
ρ_{ij}	Co-retrieval correlation of keys i and j, Equation (2)
$\mathcal{A}$	Covariance agreement of an estimator, Equation (14)
J	Partition objective, Equation (6)

B ALGORITHM

Algorithm 1 details the construction of partitions and Algorithm 2 details the page retrieval process. The partition is built once per prefill, for each paged layer and KV head, at the end of the question block; the gate is evaluated at each decoding step by query heads. Empty pages after the final Lloyd assignment are skipped when computing centroids and spreads and masked during decoding.

Algorithm 1 TesseraKV pre-fill: constructing the partition.

Input: Context keys $\boldsymbol{K} \in \mathbb{R}^{n \times d}$; context queries $\{\mathbf{Q}_{\mathrm{ctx}}^{(g)}\}_{g=1}^{G}$; question queries $\{\mathbf{Q}_{\mathrm{qst}}^{(g)}\}_{g=1}^{G}$; query-mix weight λ; target page size c; sinks n_{s}
Output: Partition π, centroids $\{\boldsymbol{c}_p\}$, dispersions $\{s_p\}$, sizes $\{|\mathcal{P}_p|\}$, slot table

for $g = 1$ **to** G **do** ▷ *per-query-head* query statistics
  $\boldsymbol{\Sigma}_{\mathrm{ctx}}{}^{(g)} \leftarrow \mathrm{Cov}(\mathbf{Q}_{\mathrm{ctx}}^{(g)})$
  $\boldsymbol{\Sigma}_{\mathrm{qst}}{}^{(g)} \leftarrow \mathrm{Cov}(\mathbf{Q}_{\mathrm{qst}}^{(g)}) \cdot \mathrm{tr}\, \boldsymbol{\Sigma}_{\mathrm{ctx}}{}^{(g)} / \mathrm{tr}\, \boldsymbol{\Sigma}_{\mathrm{qst}}{}^{(g)}$ ▷ trace-match to $\boldsymbol{\Sigma}_{\mathrm{ctx}}{}^{(g)}$
  $\boldsymbol{\Sigma}_{\mathrm{q}}{}^{(g)} \leftarrow (1-\lambda)\, \boldsymbol{\Sigma}_{\mathrm{ctx}}{}^{(g)} + \lambda\, \boldsymbol{\Sigma}_{\mathrm{qst}}{}^{(g)}$
end for
$\overline{\boldsymbol{\Sigma}} \leftarrow \frac{1}{G} \sum_{g=1}^{G} \boldsymbol{\Sigma}_{\mathrm{q}}{}^{(g)}$ ▷ the shared metric of Section 3.4: average
$\phi(\boldsymbol{k}) \leftarrow \overline{\boldsymbol{\Sigma}}^{1/2} \boldsymbol{k}$

$\mathcal{R} \leftarrow \{1, \ldots, n_{\mathrm{s}}\}$; $\mathcal{C} \leftarrow \{n_{\mathrm{s}}+1, \ldots, n\}$ ▷ sinks are excluded from partition
$M \leftarrow \max\big(\lceil |\mathcal{C}|/c \rceil,\, 1\big)$
$\pi \leftarrow \textsc{Lloyd}\big(\{\phi(\boldsymbol{k}_i)\}_{i \in \mathcal{C}},\, M\big)$ ▷ 10 iterations, Euclidean k-mean

for $p = 1$ **to** M **do**
  $\boldsymbol{c}_p \leftarrow \frac{1}{|\mathcal{P}_p|} \sum_{i \in \mathcal{P}_p} \boldsymbol{k}_i$
  ${s_p}^2 \leftarrow \frac{1}{|\mathcal{P}_p|} \sum_{i \in \mathcal{P}_p} \frac{(\boldsymbol{k}_i - \boldsymbol{c}_p)^\top \overline{\boldsymbol{\Sigma}}\, (\boldsymbol{k}_i - \boldsymbol{c}_p)}{d}$
  store $\mathcal{P}_p$'s indices in slots by *decreasing* position
end for
return π, $\{\boldsymbol{c}_p\}$, $\{s_p\}$, $\{|\mathcal{P}_p|\}$, slot table ▷ $\overline{\boldsymbol{\Sigma}}$ and ϕ are discarded

C CO-RETRIEVAL ON NATURAL TEXT

Figure 2 compares two ways of predicting whether a pair of keys has correlated attention across decoding queries: the plain cosine used by prior work, and the $\boldsymbol{\Sigma}_{\mathrm{q}}$-weighted cosine of Equation (2). This section describes how each quantity in the figure is computed.

Data. We use six SlimPajama domains (Shen et al., 2023) — ArXiv, Books, C4, CommonCrawl, GitHub, and Wikipedia — with 41 to 45 documents per model, each truncated to its first 16k tokens. Measurements are taken on the global full-attention layers only.

Query populations. A document is passed through the model once. The queries emitted while reading the 16k-token context give the estimate $\widehat{\boldsymbol{\Sigma}}$, the covariance of those context queries. The model then decodes 256 further tokens, and the queries at those positions are held out: they never enter $\widehat{\boldsymbol{\Sigma}}$ and are the only queries used to evaluate the predictions.

The measured quantity. For each (document, layer), we draw 8,192 key pairs (i, j) uniformly from the cached keys, with $i \neq j$. For each pair, the *empirical co-retrieval* is the correlation of their attention logits across the held-out queries, $\rho_{ij}^{\mathrm{emp}} = \mathrm{Corr}_t(\ell_{ti}, \ell_{tj})$, computed separately for each query head. We compare this quantity with two predictors. The plain cosine is ${\boldsymbol{k}_i}^\top \boldsymbol{k}_j / (\|\boldsymbol{k}_i\| \|\boldsymbol{k}_j\|)$, which Equation (2) reduces to when $\boldsymbol{\Sigma}_{\mathrm{q}} = \boldsymbol{I}$. The $\boldsymbol{\Sigma}_{\mathrm{q}}$-weighted cosine is Equation (2) evaluated at $\widehat{\boldsymbol{\Sigma}}$.

Algorithm 2 TesseraKV decoding step: selecting the working set.

Input: Queries $\{\boldsymbol{q}_g\}_{g=1}^{G}$; centroids $\{\boldsymbol{c}_p\}$, dispersions $\{s_p\}$, sizes $\{|\mathcal{P}_p|\}$, slot table; budget B; coefficient κ'; sink set $\mathcal{R}$; open tail $\mathcal{O}$
Output: Working set $\mathcal{W}$

for $g = 1$ **to** G, $p = 1$ **to** M **do**
  $\gamma_{g,p} \leftarrow \dfrac{\boldsymbol{q}_g \cdot \boldsymbol{c}_p}{\sqrt{d}} + \kappa' s_p$
end for
$\gamma_{g,p} \leftarrow -\infty$ **for all** p **with** $|\mathcal{P}_p| = 0$

$\gamma_p \leftarrow \frac{1}{G} \sum_{g=1}^{G} \left[\mathrm{softmax}_{p'}(\gamma_{g,p'})\right]_p$ ▷ group-consistent reduce: a page must be wanted by *many* heads

$(o_1, \ldots, o_M) \leftarrow$ the page indices ordered by decreasing γ_p ▷ $\mathcal{P}_{o_j}$ is the j-th ranked page
spent $\leftarrow 0$; $\mathcal{S} \leftarrow \emptyset$
for $j = 1$ **to** M **do** ▷ budget walk in *keys*
  $t \leftarrow \min\big(|\mathcal{P}_{o_j}|,\ \max(B - \textit{spent},\ 0)\big)$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{\text{first } t \text{ slots of } \mathcal{P}_{o_j}\}$; *spent* $\leftarrow$ *spent* $+ t$
end for

$\mathcal{W} \leftarrow \mathcal{R} \cup \mathcal{S} \cup \mathcal{O}$ ▷ $\mathcal{O}$ is the question plus what has been generated: read in full
return $\mathcal{W}$ ▷ written to the request's index table

The score. Within each (document, layer, head), we compute the Pearson correlation between a predictor and ρ^{emp} across the 8,192 pairs, yielding one agreement $\bar{r}$ per head per document. These values are averaged over documents before forming the histogram, so each unit in Figure 2 is a (layer, head) pair: 416 units for Muse-Glimmer-30B, 384 for North-Mini-Code-30B, and 256 for Command-R-7B.

C.1 Predicting Attention-Logit Covariance

We repeat the analysis of Figure 2 for attention-logit covariance, which retains the magnitude of joint logit fluctuations. For each key pair, Equation (3) predicts covariance as $\boldsymbol{k}_i^{\top} \widehat{\boldsymbol{\Sigma}} \boldsymbol{k}_j / d$; the identity baseline predicts $\boldsymbol{k}_i^{\top} \boldsymbol{k}_j / d$. We compare both with the empirical covariance over the same held-out decoding queries used for the correlation analysis. For each document, layer, and head, we compute the Pearson correlation between predicted and empirical covariances across sampled key pairs, then average this value over documents. This measures how well each predictor distinguishes key pairs with different covariances, rather than its calibration in absolute units.

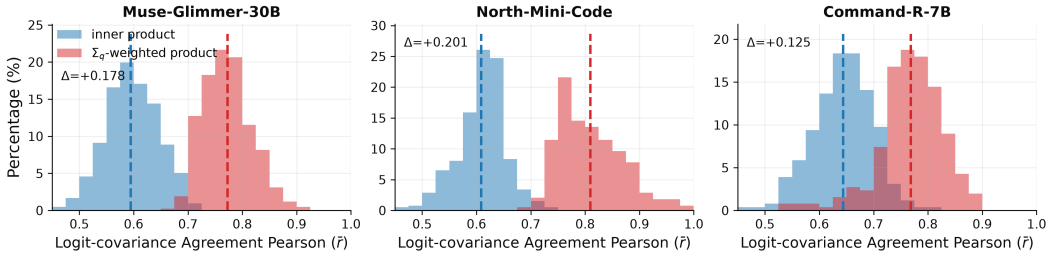

Figure 8: Agreement between predicted and empirical attention-logit covariances on SlimPajama. Each panel shows the distribution over (layer, head) pairs of Pearson correlations $\bar{r}$ across sampled key pairs, averaged over documents from six domains. The predictors use either the prefill query covariance in equation 3 or the identity matrix, which gives the scaled raw inner product. Larger $\bar{r}$ indicates better agreement; dashed lines mark the mean for each predictor.

Figure 8 shows higher agreement with the estimated query covariance than with the identity baseline in all three models. The mean gains are 0.178, 0.201, and 0.125 for Muse-Glimmer-30B, North-Mini-Code-30B, and Command-R-7B, respectively. The corresponding mean gains for logit correlation are 0.134, 0.179, and 0.125. Thus, the benefit of query covariance persists when the magnitude of logit co-fluctuation is retained.

D Detailed Experimental Setup

Configuration. We estimate query covariance from context and question queries using Equation (15), retain the first four positions as attention sinks (Xiao et al., 2024; Ruscio et al., 2026; Wong et al., 2025b), and share one partition per KV group under $\overline{\boldsymbol{\Sigma}}$. All experiments use 10 k-means iterations. Table 5 lists the query-mix weight and gate coefficient for each benchmark; their effects are examined in Section 4.2, Appendix E.2 and Appendix E.4. CLAW-EVAL uses the same configuration as INFINITEBENCH.

For the multi-turn CLAW-EVAL comparison, we extend LouisKV by treating the accumulated context at each turn as a new prefill and rebuilding its partition. This follows the same repartitioning schedule as the TesseraKV$_{\text{Repartition}}$ variant.

Table 5: Configuration per benchmark. λ is the query-mix weight of Equation (15) and κ' the coefficient of the linear spread bonus variant of Equation (13).

Benchmark	Arm	λ	κ'	Gate	k-means
RULER 8K / 16K	TesseraKV	0.5	6	linear	euclidean
INFINITEBENCH	TesseraKV	0.2	4	linear	euclidean
CLAW-EVAL	TesseraKV	0.2	4	linear	euclidean

CLAW-EVAL task specification. The benchmark ships 300 tasks: 161 text agent tasks (`T*`), 38 that drive a simulated user (`C*`), and 101 multimodal ones (`M*`). We evaluate the `T*` family less the six tasks carrying an attachment (`T056`, `T057`, `T058`, `T072` attach an image; `T091` and `T096` a text or CSV file), leaving 155.

D.1 Estimator quality

The sweeps in Figure 4 evaluate Muse-Glimmer-30B on the 13 *NoPE* layers. The unit of measurement is a *(sequence, layer) cell*; Algorithm 3 defines how the two metrics are computed within each cell.

Algorithm 3 One (sequence, layer) cell. T is the number of decode steps that sequence produced, P is the number of sampled key pairs, and a ranges over the query heads.

$\widehat{\boldsymbol{\Sigma}}_\lambda \leftarrow \big((1-\lambda)\,\boldsymbol{\Sigma}_{\text{ctx}} + \lambda\,\boldsymbol{\Sigma}_{\text{qst}}\big)$, trace-matched pre-fill ▷ Equation (15), from this sequence's own
draw P pairs (i, j), $i \neq j$, uniformly over the cached keys ▷ once per cell
for each pair (i, j) and query head a **do**
  $\rho^{\text{emp}}_{ij} \leftarrow \text{Corr}_{t \le T}\left(\ell_{ti},\, \ell_{tj}\right)$ ▷ over the decode queries
  $\hat{\rho}_{ij} \leftarrow \boldsymbol{k}_i{}^\top \widehat{\boldsymbol{\Sigma}}_\lambda \boldsymbol{k}_j \,/\, \sqrt{\boldsymbol{k}_i{}^\top \widehat{\boldsymbol{\Sigma}}_\lambda \boldsymbol{k}_i \cdot \boldsymbol{k}_j{}^\top \widehat{\boldsymbol{\Sigma}}_\lambda \boldsymbol{k}_j}$ ▷ Equation (2)
end for
co-retrieval agreement $\leftarrow \text{mean}_a \,\text{Corr}_{(i,j)}\left(\hat{\rho},\, \rho^{\text{emp}}\right)$ ▷ over the P pairs
$\mathcal{A} \leftarrow \text{mean}_a \left\langle \widehat{\boldsymbol{\Sigma}}_\lambda, \boldsymbol{\Sigma}^\star \right\rangle_F \big/ \left(\|\widehat{\boldsymbol{\Sigma}}_\lambda\|_F \|\boldsymbol{\Sigma}^\star\|_F\right)$

For each cell, we sample $P = 8{,}192$ KV key pairs once and reuse them across all decode steps. The T decode steps provide the samples for estimating empirical correlations. We use 128 decode tokens per document on INFINITEBENCH and 5–15 on RULER, depending on answer length, following KVPress (Devoto et al., 2025). Both metrics are computed in a single pass over the same captures. We sample ten sequences from each of RULER's 13 tasks and ten documents from book summarisation.

For each sequence, the 13 layers of the model share the same document and captured queries. We first average across layers to obtain one value per sequence, then report ± 1 standard error across sequences.

For each layer and query head, we estimate the reference covariance $\mathbf{\Sigma}^\star$ from decoding queries, centred by their pooled mean, across the rows of the task. On RULER, this pools all 130 rows, yielding approximately 1,328 queries per head. On book summarisation, it pools the 10 documents, yielding 7,361 queries per head.

Because $\mathbf{\Sigma}^\star$ is itself a finite-sample estimate, the two metrics have different interpretations. For *co-retrieval agreement*, the target is each row's own empirical correlation rather than $\mathbf{\Sigma}^\star$; a covariance estimated from other rows can therefore outperform $\mathbf{\Sigma}^\star$ and exceed 100%. In contrast, *covariance agreement* $\mathcal{A}$ measures agreement directly with $\mathbf{\Sigma}^\star$ and is bounded by 1.

Sequence Geometry. Using n sequences raises the question of whether the results depend on the sampled sequences. We therefore conduct the transfer experiment in Figure 9: each cell estimates $\mathbf{\Sigma}_{\mathrm{q}}$ from one task's rows and evaluates it on another's. The *diagonal* thus measures within-task transfer, where the covariance is estimated from one set of rows and evaluated on disjoint rows from the same task.

The results depend strongly on the query source. When estimated from decoding queries, the diagonal recovers 89%–102% of the identity-to-reference range for co-retrieval agreement and 88%–99% for covariance agreement. In contrast, covariances estimated from pre-fill queries recover only 19%–45%. Thus, pre-fill geometry varies substantially across rows within a task, whereas decoding geometry is relatively stable. A covariance estimated from disjoint rows therefore remains close to one estimated on the evaluated rows, suggesting that decoding-query geometry is primarily a property of the task rather than the particular sample.

The off-diagonal entries measure cross-task transfer and show a different pattern. Among the four natural-text tasks, covariances transferred from another task recover 30%–88% of the identity-to-reference range. In contrast, using KV RETRIEVAL as the source is consistently harmful, yielding -68% to -37%, below the identity baseline. Thus, synthetic-filler retrieval and natural-prose reading induce distinct decoding-query subspaces, such that a covariance learned from one can degrade performance on the other.

E ABLATION STUDY

E.1 THE GQA GROUP

Table 6: Per-query-head versus shared partitioning on a balanced subset of RULER-16k, averaged over five KV budgets and 130 samples per cell.

Model	G	Per query head	Shared, $\overline{\mathbf{\Sigma}}$	Margin
Muse-Glimmer-30B	16	79.28	75.14	$+4.14$
North-Mini-Code-30B	8	72.46	69.37	$+3.09$
Command-R-7B	4	69.51	67.63	$+1.88$

GQA requires a single key partition for each query-head group. To quantify the cost of this constraint, Table 6 compares our shared partition, constructed from the averaged covariance $\overline{\mathbf{\Sigma}}$ in Equation (16), with an otherwise identical per-head construction. We fix the covariance estimates in both cases, isolating the effect of partition sharing.

However, this per-head construction provides shouldn't be viewed as an alternative. It reads 2.2–$4.8\times$ more keys per group and stores G times as many centroids. The accuracy gap is 4.14, 3.09, and 1.88 points for $G = 16$, 8, and 4, respectively. Thus, sharing partitions incurs a modest accuracy cost that decreases as fewer query heads share the same keys.

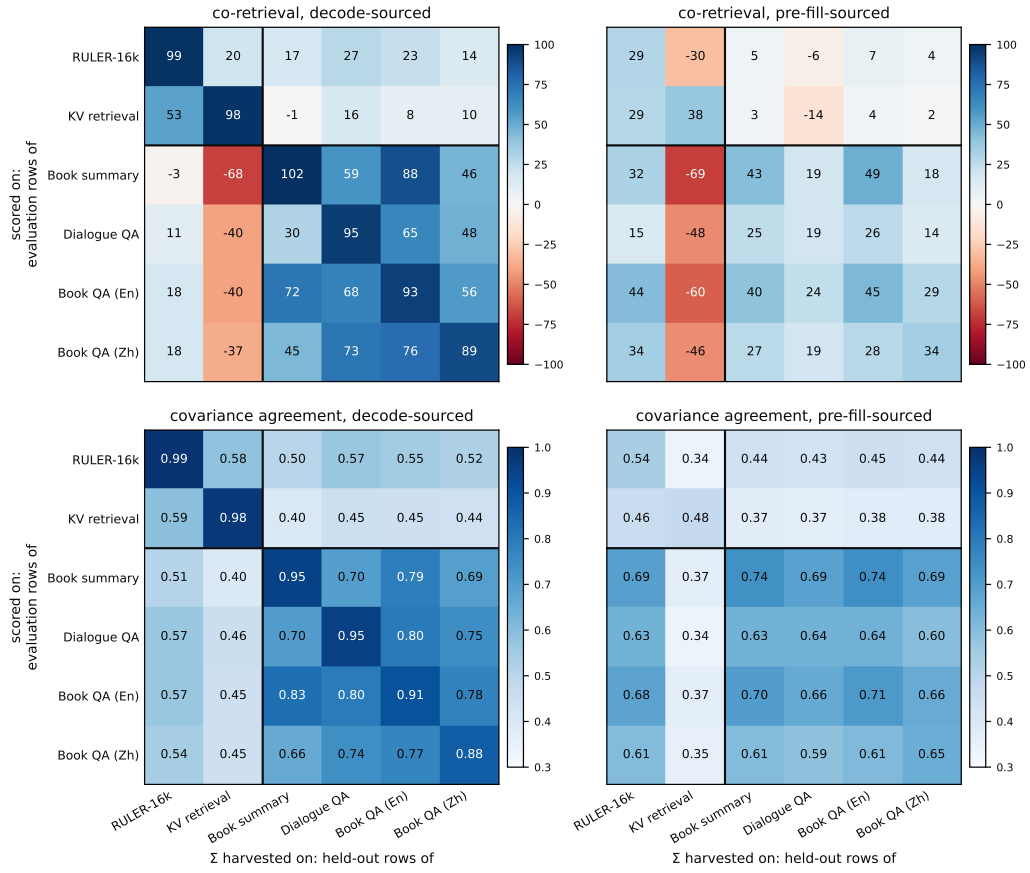

Figure 9: Transfer between tasks. Each cell estimates an offline $\mathbf{\Sigma}_{\mathrm{q}}$ from the column task and evaluates it on the row task. **(Top)** Co-retrieval agreement, normalized to the evaluated task's range from $\mathbf{\Sigma}_{\mathrm{q}} = \boldsymbol{I}$ to its reference; 0 indicates no improvement over Identity, while values may exceed 100%. **(Bottom)** Covariance agreement. The left and right columns use decoding and pre-fill queries, respectively. The diagonal denotes the held-out within-task setting, where $\mathbf{\Sigma}_{\mathrm{q}}$ is estimated from an independent sample of the same task.

E.2 Ablations of Gate Strength, Form, and Partition Metric

Ablation Analysis. The spread correction involves two design choices: the functional form of the bias and its strength. Equation (13) derives a quadratic bias proportional to ${s_p}^2/2$, while our deployed gate uses a linear bias $\kappa' s_p$. We also compare partitioning by covariance and correlation, using Euclidean and spherical k-means, respectively, following Section 3.2. Figure 10 evaluates these choices across gate strengths on a retrieval benchmark (a balanced subset of RULER-16K) and a natural-prose benchmark (a subset of InfiniteBench Book Summarization). Table 7 averages the four configurations to isolate the effect of gate strength.

Table 7: Mean score over the four (partition metric, gate form) methods of Figure 10, at each coefficient. Averaging over the methods removes the crossings visible in the individual curves and shows the effect of the swept coefficient. Best per row in bold.

Benchmark	0	2	4	6	8	10
RULER	68.94	73.26	77.17	**77.62**	75.19	73.61
InfiniteBench summ.	18.52	19.10	**20.20**	19.67	19.58	19.42

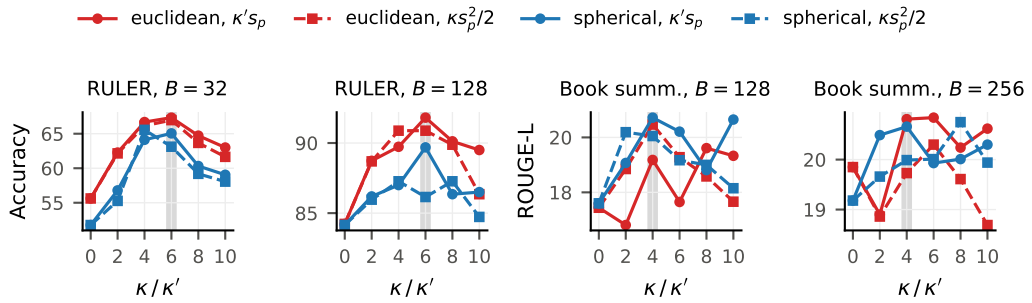

Figure 10: Spread-gate performance as a function of gate coefficient on a balanced subset of RULER-16K and INFINITEBENCH Book Summarization, at two KV-cache budgets per benchmark. Line style denotes gate form and color denotes partition metric. Solid curves use the linear bias $\kappa' s_p$; dashed curves use the quadratic bias $\kappa {s_p}^2/2$. Both coefficients take the values shown on the horizontal axis, with zero disabling the gate. Grey bands mark the benchmark-specific linear coefficients used in Table 5.

Gate strength. Performance follows an inverted-U trend, with the preferred coefficient depending on the workload. Relative to no correction, the best coefficient improves performance by 8.7 points on RULER and 1.7 points on Book Summarization, averaged over the four configurations. The larger variation on RULER indicates that the spread correction is more consequential for synthetic retrieval than for natural prose. On RULER, a coefficient of 6 outperforms 4 in 6 of 8 methods, whereas Book Summarization favors 4 in 5 of 8 methods; the difference is approximately 0.5 points in either direction. We therefore use $\kappa' = 6$ for RULER and $\kappa' = 4$ for Book Summarization in Table 5. This task dependence mirrors that of the query-mix weight (Section E.4).

Gate form. The linear and quadratic forms perform similarly at their respective per-cell optima, with the linear form achieving the higher score in 5 of 8 cells. Neither form therefore consistently dominates over the range of spreads encountered in our experiments. The distinction is more apparent without per-cell retuning: increasing the swept coefficient to 10 causes a smaller performance drop for the linear form in 7 of 8 cells, by 1.3 points on average relative to each form's optimum. Since the coefficient is fixed offline and reused across requests and models, this greater robustness motivates our use of the linear form.

Partition metric. Logit covariance-based partitioning outperforms logit correlation-based partitioning on RULER in all 24 combinations of budget, gate form, and coefficient, by 3.14 points on average. On Book Summarization, the two metrics differ by less than 0.5 points on average, with spherical clustering slightly ahead in more cells. This difference is consistent with the role of key magnitude: spherical k-means normalizes the transformed keys and removes magnitude information, whereas Euclidean clustering retains it. Magnitude is more informative when retrieval depends on a small number of distinctive keys and less informative when keys are more uniformly distributed, as in natural prose. We therefore use covariance-based partitioning throughout.

E.3 THE COVARIANCE AND THE GATE, CROSSED

Here we disentangle the roles of partition and gate geometry. In Table 8, we cross three partition choices with three gate choices: a contiguous positional partition, Euclidean k-means under $\mathbf{\Sigma}_{\mathrm{q}} = \boldsymbol{I}$, or Euclidean k-means under the query mix of Equation (15) with $\lambda = 0.5$; and no gate, a gate with $\kappa' = 6$ using plain distance, or the corresponding $\mathbf{\Sigma}_{\mathrm{q}}$-weighted spread. All differences are paired by sequence and tested with a sign test.

Partitioning and gating both help. Replacing the contiguous partition with k-means improves performance by 22.7–32.5 points across all budgets. Within the k-means partitions, adding the spread correction provides a further improvement of up to 15.5 points. Thus, both partitioning and gating contribute substantially to retrieval quality.

Table 8: Crossing partition geometry and spread correction on RULER-16K across KV-cache budgets. Rows vary the partition: a contiguous positional partition, Euclidean k-means under $\mathbf{\Sigma}_q = \boldsymbol{I}$, or Euclidean k-means under the query mix of Equation (15) with $\lambda = 0.5$. Columns vary the gate: none, $\kappa' = 6$ with spread measured by plain distance, or with spread measured under $\mathbf{\Sigma}_q$. Results are averaged over 650 sequences (50 per task); full attention scores 95.08.

	$B = 16$			$B = 32$			$B = 64$			$B = 128$			$B = 256$		
	no gate	$\boldsymbol{I}$	$\mathbf{\Sigma}_q$	no gate	$\boldsymbol{I}$	$\mathbf{\Sigma}_q$	no gate	$\boldsymbol{I}$	$\mathbf{\Sigma}_q$	no gate	$\boldsymbol{I}$	$\mathbf{\Sigma}_q$	no gate	$\boldsymbol{I}$	$\mathbf{\Sigma}_q$
positional	19.70	20.34	20.97	25.32	26.91	33.65	40.99	44.75	57.04	51.78	57.08	67.58	63.34	68.23	76.76
$\mathbf{\Sigma}_q = \boldsymbol{I}$	36.44	41.85	**45.21**	47.12	59.33	65.62	61.76	74.52	77.71	80.94	86.59	87.36	91.06	93.34	93.90
$\mathbf{\Sigma}_q$, $\lambda = 0.5$	35.42	41.63	43.69	50.66	62.38	**66.14**	69.53	79.08	**81.37**	84.76	89.72	**90.48**	93.11	**94.27**	94.18

$\mathbf{\Sigma}_q$ improves both components. Using $\mathbf{\Sigma}_q$ for either partitioning or spread correction generally improves performance over $\mathbf{\Sigma}_q = \boldsymbol{I}$, with the strongest results obtained when $\mathbf{\Sigma}_q$ is used for both. Using $\mathbf{\Sigma}_q$ consistently improves performance by up to 6.9 points over $\mathbf{\Sigma}_q = \boldsymbol{I}$ in both components, with significant gains from $B = 32$ onward.

The two effects overlap. Using $\mathbf{\Sigma}_q$ in either the partition or the gate recovers much of the gain from using it in both. Thus, the benefits of query-induced geometry in partitioning and gating are complementary but not directly additive: applying $\mathbf{\Sigma}_q$ to both remains strongest, but the two improvements provide overlapping ranking information. At $B = 256$, all effects beyond partitioning are below one point.

E.4 The Query Mix Across Tasks

We extend the two-workload analysis in Figure 4 to six workloads, using the same normalization: each metric is reported as a fraction of the task's range from $\mathbf{\Sigma}_q = \boldsymbol{I}$ to its reference. We use ten RULER sequences per task and ten documents per INFINITEBENCH task, and additionally show a held-out $\mathbf{\Sigma}_q$ for each task. The two question-answering tasks are excluded because their answers are only one to six tokens long, providing too few decode steps to estimate correlations reliably; this leaves 110 sequences.

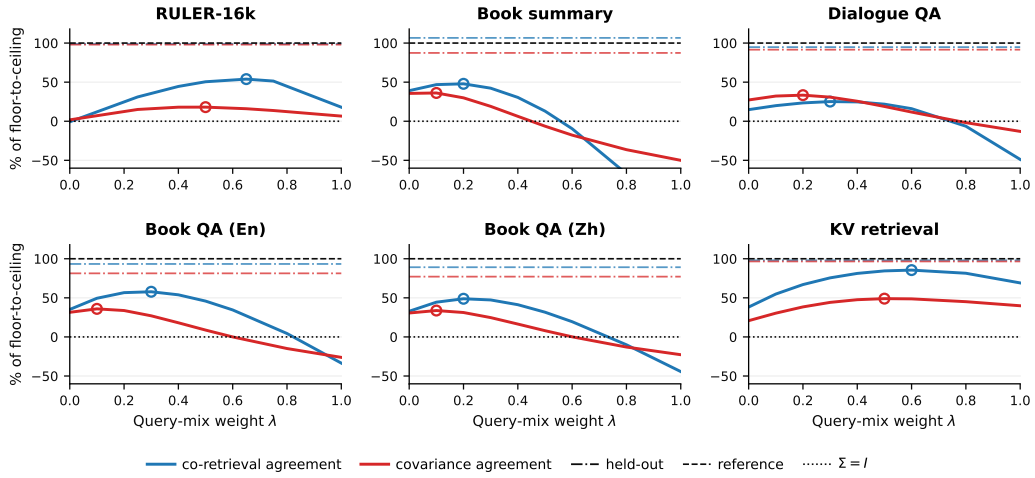

Figure 11: Co-retrieval and covariance agreement across query-mix weights. Both metrics are normalized to the range from $\mathbf{\Sigma}_q = \boldsymbol{I}$ to each task's reference on Muse-Glimmer-30B, with circles marking their optima. The optimal λ tracks context composition rather than length: synthetic retrieval tasks peak near $\lambda = 0.6$, while natural-text tasks peak near $\lambda = 0.2$. This holds even when KV RETRIEVAL uses 124k tokens, matching the book tasks in length and RULER in composition. Held-out covariances are computed offline from rows excluded from evaluation.

The optimal mix is task-dependent and tracks the content of the context rather than its length. The two synthetic retrieval tasks peak near $\lambda = 0.6$, while the four natural-text tasks peak near $\lambda = 0.2$. KV RETRIEVAL helps separate context length from content: despite reaching 124k tokens, it follows the synthetic RULER tasks rather than the natural-text tasks. Thus, the preferred mix differs systematically between synthetic retrieval and natural-prose workloads, motivating benchmark-specific mix weights in Table 5.

F CLAW-EVAL BUDGET AND REPARTITIONING

We report the full repartitioning comparison and results for AgentKV (Liu et al., 2026), an eviction method evaluated at budgets 512, 1024, and 1536.

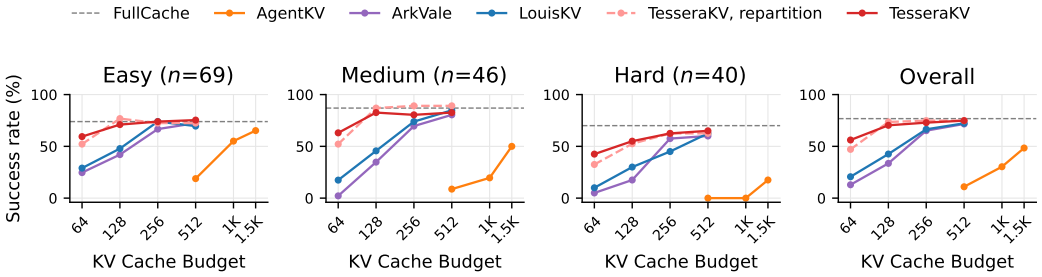

Figure 12: CLAW-EVAL success rate against the key budget, by difficulty tier and task-weighted over all 155 tasks. The dashed line is full attention. *AgentKV* was run at $B \in \{512, 1024, 1536\}$ only and the remaining methods at $B \leq 512$.

Table 9: CLAW-EVAL success rates (%) for incremental partitioning and full repartitioning at each turn. Both variants use the same KV budget B and evaluation protocol as Table 2.

	Easy				Medium				Hard				Overall			
KV budget B	64	128	256	512	64	128	256	512	64	128	256	512	64	128	256	512
$\text{TesseraKV}_{\text{Repartition}}$	52.17	76.81	72.46	72.46	52.17	86.96	89.13	89.13	32.50	52.50	62.50	62.50	47.10	73.55	74.84	74.84
TesseraKV	59.42	71.01	73.91	75.36	63.04	82.61	80.43	82.61	42.50	55.00	62.50	65.00	56.13	70.32	72.90	74.84

Table 9 compares incremental partitioning with rebuilding the partition over the full context at every turn. In overall success rate, incremental partitioning leads by 9.03 points at $B = 64$, trails by 3.23 and 1.94 points at $B = 128$ and 256, and ties at $B = 512$.

In Figure 12, the retrieval methods differ most at tight budgets. At $B = 64$, TesseraKV reaches 56.1%, compared with 20.7% for LouisKV and 12.9% for ArkVale. At $B = 512$, their success rates rise to 74.8%, 72.3%, and 71.6%, respectively, against 76.8% for full attention.

Eviction behaves differently in kind: AgentKV reaches 30.3% at $B = 1024$, below TesseraKV's 56.1% at one-sixteenth of that budget, and solves none of the hard tier until $B = 1536$, three times the largest retrieval budget tested. CLAW-EVAL carries the cache across turns, so a key discarded in one turn is unavailable in the next, whereas the retrieval methods retain the cache and re-select from it at every step.

G ADDITIONAL INFERENCE DETAILS

G.1 INFERENCE IMPLEMENTATION

Our SGLang implementation combines asynchronous index construction during prefill with GPU-resident page selection and sparse attention during decode. KV paging is applied to global-attention layers, while sliding-window layers retain their local attention pattern. The full KV cache remains resident on the GPU; the index stores page summaries and physical token addresses.

Prefill and partition construction. Both our implementation and the dense baseline use FlashAttention 4 for prefill. We accumulate query statistics in FP32 across prefill chunks without retaining the full query tensor. These statistics determine the covariance- weighted key transformation used for clustering. To accelerate Lloyd iterations, we adapt the tiled Triton assignment kernel from `flash-kmeans`, using TF32 matrix multiplication with FP32 accumulation while retaining FP32 features and centroids. Centroid updates and the final spread correction follow our reference implementation. Once a layer's inputs are available, a background host worker dispatches index construction onto a separate CUDA stream, allowing it to overlap with subsequent layers' forward computation. CUDA events enforce data dependencies, and indexed decode waits for construction to complete. Consequently, overlap can hide part of the indexing cost; any unfinished work remains on the critical path.

Page scoring. The decode indexer computes query–centroid scores using batched matrix multiplication with BF16 inputs and outputs. Triton kernels then apply the spread correction, compute per-query-head softmax in FP32, average probabilities within each GQA group, and select pages until the key budget is filled. The optimized selection path extracts candidates in top-k blocks rather than fully sorting all pages, truncating the final selected page when necessary. It packs physical token addresses without copying the selected KV entries. These operations are fused where supported; larger page counts use separate softmax and selection kernels.

Sparse attention. A custom Triton kernel reads the selected keys and values directly from their physical locations in the KV cache, together with the always-attended sink tokens and question/generated tail. It partitions the attention computation across 16 splits, maintains FP32 online-softmax statistics and accumulators, and combines the partial outputs with a subsequent merge kernel. This avoids materializing a compact KV tensor and reduces decode-time KV reads. The dense baseline uses the FlashAttention 4 backend.

G.2 Latency Across Context Lengths and Batch Sizes

Figure 13 extends the breakdown in Figure 7 to 32k context at batch size 64 and 128k at batch size 16. All three configurations use 512 decode steps.

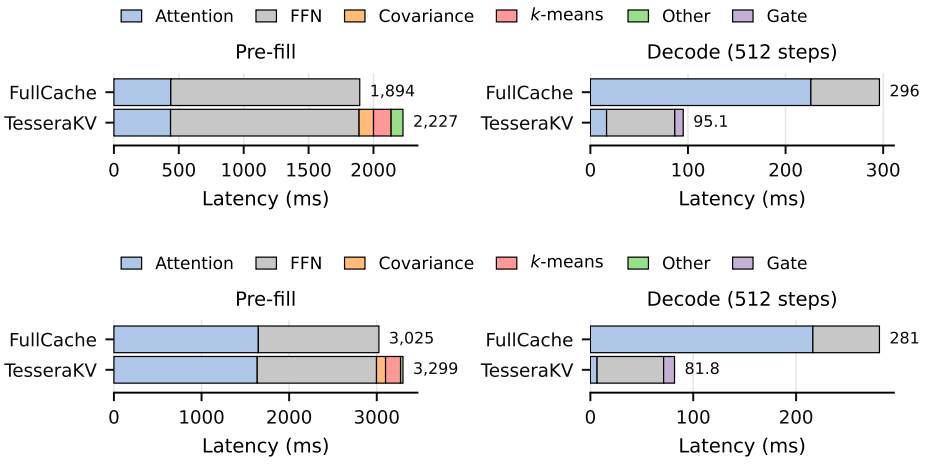

Figure 13: Profiled latency breakdown for one layer with global attention at 32k context and batch size 64 (top) and 128k context and batch size 16 (bottom), with $B = 512$. Bars sum component times without asynchronous overlap; decode totals cover 512 steps.

Prefill attention becomes more expensive at longer contexts, while the feed-forward component changes much less across these configurations.

The added costs remain similar across configurations: covariance estimation takes 111, 106, and 103 ms, while index construction takes 227, 172, and 199 ms at 32k, 64k, and 128k, respectively.

When profiled component times are summed without overlap, the relative increase in prefill time falls from 17.6% to 11.5% and 9.1% as global attention becomes more expensive.

During decoding, the indexer accounts for an increasing share of the profiled paging cost: 33%, 48%, and 61% at 32k, 64k, and 128k, respectively. More pages must be scored as the context grows, while the attention budget B remains fixed. Sparse attention alone is $13.6\times$ to $33.2\times$ faster than full attention, but the gain is $8.7\times$ to $12.4\times$ when page selection is included in the summed component times. Page selection therefore limits the projected decode gain at longer contexts.

These per-layer measurements include the feed-forward block; Figure 6 reports whole-model throughput.

H Limitations

Our method is designed for NoPE global full-attention layers and does not directly support rotary positional embeddings (RoPE). Although NoPE is increasingly used in the global attention layers of hybrid long-context models, some models retain RoPE, such as Qwen-Next (Qwen Team, 2026). The challenge arises because RoPE rotates keys and queries according to their positions, making the covariance geometry position-dependent. One possible extension is to average the rotations over the n future positions, analogous to how Expected Attention (Devoto et al., 2025) obtains a position-averaged second moment. A systematic investigation of this approach is left to future work.

I Relation to Other Axes of Efficient Inference

Reducing KV-cache traffic is one axis in the broader effort to reduce inference cost, and these axes are largely complementary. Weight-side methods reduce model size: post-training quantization lowers parameter precision (Lin et al., 2024; Frantar et al., 2022; Zandieh et al., 2026; Muller et al., 2026; Zhang et al., 2024a; 2025), while factorization methods compress weight matrices directly (Ji et al., 2025; Wang et al., 2025; Wong et al., 2025a). Cache-side methods, the focus of this paper, change how cached keys and values are stored or accessed while leaving model weights unchanged.

Our method also naturally extends to multi-agent workloads (Zhang et al., 2024c; Zou et al., 2025; Wong et al., 2026; Wei et al., 2026; Xu et al., 2026), where multiple tool-using agents maintain long, multi-turn sessions of the kind evaluated in this work. Each agent can independently maintain and page its own KV cache.

On the system side, our implementation keeps the partitioned KV cache in HBM. The same partitioned representation could instead be offloaded to CPU memory, following prior systems for KV-cache offloading (Xiao et al., 2026; Wu et al., 2026). We leave this extension to future work.