SIEVE: ESTIMATE-AWARE CLUSTERING IN THE QUERY-INDUCED SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse attention uses only a small part of the KV cache at each decoding step, while cluster-based indices rank and estimate groups of keys through their centroids. Existing methods cluster keys by angular similarity, even though attention depends on the inner products between keys and decoding queries. Keys that are close in angle can therefore induce very different logits, causing important keys to be diluted in both retrieval and estimation. We introduce SIEVE, which clusters keys in the score space induced by the query distribution. Its objective minimizes residual logit error under a robust query-induced metric and places additional weight on keys that are likely to become important retrieval targets. We show that this objective controls errors in attention estimation. Across long-context and long-reasoning benchmarks, SIEVE outperforms training-free sparse-attention baselines at equal budget and remains close to full attention while using only a small fraction of the KV cache.

1 INTRODUCTION

Large language models now read and reason over contexts of hundreds of thousands of tokens (Liu et al., 2024a). To avoid recomputing the past, a Transformer stores each processed token's key and value in memory (Vaswani et al., 2017). This Key–Value (KV) cache grows with the context, and every decoding step reads all of it, so at long context it dominates inference memory and latency (Tang et al., 2024). Once the cache exceeds GPU memory, it must be offloaded to CPU memory, whose link to the GPU has a bandwidth about 60 times lower than that of GPU memory (Chen et al., 2025a).

Sparse attention makes offloading practical. For a given query, a small subset of tokens dominates the attention output (Zhang et al., 2023b; Tang et al., 2024), so each decoding step can load only this subset and attend to it exactly (Sun et al., 2024; Liu et al., 2025; Chen et al., 2025a). This turns decoding into a search problem: finding the tokens with the largest logits without scoring the whole cache. Cluster-based indices answer it with a partition of the keys (Liu et al., 2025; Chen et al., 2025a). The clusters are ranked by the logits of their centroids, and the top ones are loaded. RetroInfer further estimates the attention of the next clusters from their centroids and sizes, without loading them (Chen et al., 2025a). The centroid thus stands in for every key of its cluster, both in the ranking and in the estimate.

Existing indices build this partition from the angular distance between keys, a choice inherited from vector search that ignores the queries reading the index. The index must preserve the logit each query assigns to each key, and the angle between two keys says little about it. The angular distance discards the norm of the keys, to which the logit is linear, and weighs all directions equally. The queries of a head, in contrast, place most of their energy on a few directions: the mean query direction alone carries about 64% of it. We show in Section 4.1 that this mismatch dilutes the strongest key of a cluster. The cluster is ranked as if this key had the average logit of the cluster, so it can be excluded in favor of weaker clusters, and the estimate averages the value of this key with those of its neighbors. The fault lies in the geometry of the partition, not in the index.

We therefore introduce SIEVE, which keeps the index and changes the geometry of its partition: two keys belong together when the queries cannot tell their logits apart. SIEVE clusters the keys under the second moment of the queries, which measures the average squared error of each logit, and

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.

guards against the error of estimating this moment by minimizing the worst case over nearby second moments, which admits a closed form. It then reweights the keys that the index must retrieve, along the direction of the queries that attend to them, so that the strongest keys are not diluted. SIEVE operates during index construction, so it adds no cost at decoding time and produces an index that existing retrieval and estimation procedures read as is.

Contributions. Our contributions are threefold:

- We identify how angular clustering dilutes dominant keys, impairing both retrieval and attention estimation, and derive bounds on estimation errors.
- We introduce SIEVE, a clustering objective that combines a robust query-induced metric with a target term that protects important keys.
- SIEVE achieves the best sparse scores on RULER, LongBench, AIME-25, and GPQA Diamond across the evaluated settings, including ties. Ablations confirm the contributions of its components.

This paper unfolds as follows: Section 2 reviews the related work, and Section 3 describes sparse attention with cluster-based indices. Sections 4 and 5 present SIEVE and the numerical results.

2 Related Work

KV cache compression bounds the cache by evicting the tokens judged unimportant, using attention sinks and a recent window (Xiao et al., 2024b), accumulated or recent attention scores (Zhang et al., 2023b; Liu et al., 2023; Oren et al., 2024; Li et al., 2024), budgets allocated across layers and heads (Cai et al., 2024; Feng et al., 2026), the diversity of keys (Park et al., 2026), or a model of future queries (Devoto et al., 2025). Eviction is irreversible, whereas the tokens that matter change from one decoding step to the next (Tang et al., 2024). Quantization (Liu et al., 2024b) reduces the size of every token and is orthogonal to our work.

KV cache retrieval keeps the whole cache and selects, at every step, the tokens that the current query needs. Quest (Tang et al., 2024), InfLLM (Xiao et al., 2024a), and ShadowKV (Sun et al., 2024) score blocks of consecutive tokens, InfiniGen (Lee et al., 2024) predicts the tokens from the previous layer, MagicPIG (Chen et al., 2025b) samples them with locality-sensitive hashing, and PQCache (Zhang et al., 2025) searches them individually with a vector index. Blocks mix important and unimportant tokens in one unit, and token-level search needs one index entry per token. Cluster-based indices group tokens by content instead.

Cluster-based indices are closest to our work. ClusterKV (Liu et al., 2025) and RetroInfer (Chen et al., 2025a) partition the keys by cosine distance and rank the clusters by their centroid logits, and RetroInfer also estimates the clusters that it does not load. ParisKV (Qi et al., 2026) instead assigns the keys to fixed directions on the unit sphere. All three group keys by direction and ignore the queries. SIEVE derives the partition from the queries, and existing retrieval and estimation procedures read its index as is.

Score-aware quantization for inner product search measures errors in inner products rather than in vectors. ScaNN (Guo et al., 2020) penalizes the error along each datapoint more than the error across it, norm-explicit quantization (Dai et al., 2020) encodes the norms separately, and query-aware quantization (Zhang et al., 2023a) replaces uniform queries by samples of the actual ones. These losses only need to rank the top results, since the other points are discarded. In a cluster-based index, the clusters that are not retrieved still enter the estimate, so our objective penalizes every key, whitens the metric by the queries, and gates its anisotropic term by key's importance.

3 Preliminaries

3.1 Full Attention and KV Cache

We consider one attention head at a decoding step. Let $k_1, \dots, k_n \in \mathbb{R}^d$ and $v_1, \dots, v_n \in \mathbb{R}^{d_v}$ denote the keys and values of the n previous tokens, stored in the key-value (KV) cache, and let $q \in \mathbb{R}^d$ denote the query of the current token, rescaled by $\frac{1}{\sqrt{d}}$ so that $\langle q, k_i \rangle$ is the attention logit of

token i. The attention output is

$$o(q) = \frac{N(q)}{Z(q)}, \qquad N(q) \triangleq \sum_{i=1}^{n} e^{\langle q, k_i \rangle} v_i, \qquad Z(q) \triangleq \sum_{i=1}^{n} e^{\langle q, k_i \rangle}. \tag{1}$$

Every decoding step reads all n key-value pairs and performs little arithmetic on each, so its latency is bound by memory bandwidth and grows linearly with n (Tang et al., 2024).

3.2 Sparse Attention with an Offloaded Cache

When the cache exceeds GPU memory, it is offloaded to CPU memory, whose link to the GPU has about 60 times less bandwidth than GPU memory (Chen et al., 2025a), so the whole cache cannot be transferred at every step. Sparse attention transfers only a small subset: for a given query, few tokens dominate the attention output, and these tokens change from one step to the next (Zhang et al., 2023b; Tang et al., 2024), as we also measure in Appendix I.

The steady tokens $\mathcal{D} \subseteq \{1, \ldots, n\}$ reside on the GPU and are attended at every step (Chen et al., 2025a). They comprise n_{sink} initial tokens known as attention sinks (Xiao et al., 2024b), a window of n_{win} recent tokens, and the pending tokens $\mathcal{P}$, generated during decoding and not yet indexed. The other tokens form the indexed region $\mathcal{X} = \{1, \ldots, n\} \setminus \mathcal{D}$ in CPU memory. At each step, a set $\mathcal{S}(q) \subseteq \mathcal{X}$ of at most B tokens is loaded to the GPU, where B is the retrieval budget, and attention is computed exactly over $\mathcal{D} \cup \mathcal{S}(q)$. Scoring every token of $\mathcal{X}$ would cost as much as full attention, so $\mathcal{S}(q)$ is selected from a compact summary of $\mathcal{X}$ kept on the GPU, which we call the *index* (Liu et al., 2025). The index is built after prefill and extended during decoding, so that it also serves long generations: the generated tokens that leave the recent window join $\mathcal{P}$, and every n_{flush} generated tokens, $\mathcal{P}$ is indexed and appended to $\mathcal{X}$. Figure 1a shows this layout.

3.3 Cluster-Based Indices

A cluster-based index partitions $\mathcal{X}$ into clusters $C_1, \ldots, C_m$ by k-means under the cosine distance (Liu et al., 2025; Chen et al., 2025a), which ClusterKV prefers because outlier channels of large magnitude make Euclidean distances unstable (Liu et al., 2025). RetroInfer runs spherical k-means on centered keys within segments of the context (Chen et al., 2025a). Each cluster C_j of size n_j is represented by the mean of its keys, $c_j = \frac{1}{n_j} \sum_{i \in C_j} k_i$, and its tokens are stored contiguously in CPU memory. At each step, the clusters are ranked by their centroid logits $\langle q, c_j \rangle$ and loaded in this order until B tokens are reached, which gives $\mathcal{S}(q)$. At every flush, the pending tokens are clustered as a new segment.

RetroInfer also estimates the clusters ranked next, which form the estimation zone $\mathcal{E}(q)$, without loading them (Chen et al., 2025a). Its index keeps on the GPU the centroid, the size, and the value sum $n_j \bar{v}_j$ of every cluster, where $\bar{v}_j = \frac{1}{n_j} \sum_{i \in C_j} v_i$, and it estimates the contribution of C_j to the denominator and the numerator by $\hat{Z}_j = n_j e^{\langle q, c_j \rangle}$ and $\hat{N}_j = \hat{Z}_j \bar{v}_j$. The clusters ranked below $\mathcal{E}(q)$ are omitted. Since c_j is the mean of the keys, Jensen's inequality gives $\hat{Z}_j \leq \sum_{i \in C_j} e^{\langle q, k_i \rangle}$, so the estimate never overestimates the mass of a cluster. The exact and estimated terms are combined in one softmax as shown in Figure 1b and the equation

$$\hat{o}(q) = \frac{\sum_{i \in \mathcal{D} \cup \mathcal{S}(q)} e^{\langle q, k_i \rangle} v_i + \sum_{j \in \mathcal{E}(q)} \hat{N}_j}{\sum_{i \in \mathcal{D} \cup \mathcal{S}(q)} e^{\langle q, k_i \rangle} + \sum_{j \in \mathcal{E}(q)} \hat{Z}_j}. \tag{2}$$

4 Methodology

Section 4.1 identifies three limitations of the angular clustering in the indices of Section 3.3. Section 4.2 replaces the angular distance by a robust metric in logit space. Section 4.3 reweights this metric toward the keys that the index must retrieve and derives the matching centroid update.

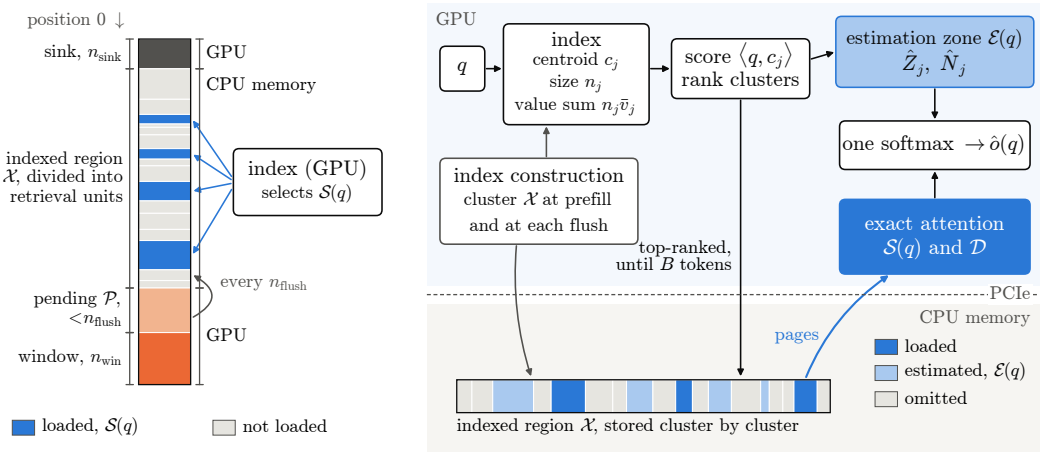

Figure 1: Sparse attention with an offloaded cache. (a) Layout of the KV cache at one decoding step. The sink, the pending tokens $\mathcal{P}$ and the recent window form the steady tokens $\mathcal{D}$, which reside on the GPU and are attended exactly, while the indexed region $\mathcal{X}$ resides in CPU memory and is divided into retrieval units. Every n_{flush} generated tokens, the pending tokens are indexed and appended to $\mathcal{X}$. (b) One decoding step with a cluster-based index. The query scores the clusters by their centroid logits, the top-ranked clusters are loaded until B tokens are reached and attended exactly with $\mathcal{D}$, and the clusters ranked next form the estimation zone $\mathcal{E}(q)$, which contributes through the centroids, sizes, and value sums. The two parts are combined in one softmax, as in equation 2.

4.1 Limitations of Angular Clustering

The index replaces the logit $\langle q, k_i\rangle$ of every key $k_i \in C_j$ by the centroid logit $\langle q, c_j\rangle$, both to rank the clusters and to estimate them. We call the difference $x_i(q) \triangleq \langle q, k_i - c_j\rangle$ the *residual logit* of k_i. Since c_j is the mean of the keys, the centroid logit is the average logit of the cluster, and $x_i(q)$ is the excess of the logit of k_i over this average. The replacement therefore divides the weight $e^{\langle q,k_i\rangle}$ of k_i by $e^{x_i(q)}$, an effect we call *dilution*, which is largest for the strongest key of a cluster. Angular clustering does not keep the residual logits small: it controls the angle between the keys, whereas $x_i(q)$ depends on the difference $k_i - c_j$ along q.

Dilution in the ranking. The index ranks clusters rather than keys, so C_j falls below another cluster $C_{j'}$ whenever $\langle q, c_{j'}\rangle > \langle q, k_i\rangle - x_i(q)$, even if k_i outscores every key of $C_{j'}$. The cluster is then excluded together with k_i, as the next example illustrates.

Example 4.1 (Dilution in the ranking)**.** Take the keys $k_1 = 2\,(1, 0)$, $k_2 = (\cos 5°, \sin 5°)$ and $k_3 = (\cos 5°, -\sin 5°)$. Their pairwise cosine similarities are at least 0.985, so angular clustering places them in one cluster C_1, although k_1 has twice the norm of the others. Their mean is $c_1 = (1.331, 0)$. At the query $q = (5, 0)$, the keys have logits 10, 5.0 and 5.0, whereas the score of C_1 is $\langle q, c_1\rangle = 6.65$, so k_1 is diluted by the factor $e^{x_1} \approx 28$. A second cluster C_2 of three keys with logit 7 scores $7 > 6.65$, so under a budget of one cluster the index loads C_2 and excludes k_1. Yet the weight of k_1 is twenty times that of each key of C_2 and 6.7 times the total mass of C_2. The residual $k_1 - c_1$ points along q, where it changes the logit most, and the angular distance ignores it because it is a difference in norm. Centering the keys, as RetroInfer does, shifts every logit by the same constant and leaves the example unchanged.

Dilution in the estimate. For a cluster C_j in the estimation zone, the index replaces the true contributions

$$Z_j = e^{\langle q,c_j\rangle} \sum_{i\in C_j} e^{x_i(q)}, \qquad N_j = e^{\langle q,c_j\rangle} \sum_{i\in C_j} e^{x_i(q)} v_i$$

by $\hat{Z}_j = n_j e^{\langle q,c_j\rangle}$ and $\hat{N}_j = e^{\langle q,c_j\rangle} \sum_{i\in C_j} v_i$, as if all residual logits were zero. A key with a large residual logit is then misrepresented twice. The mass of its cluster is underestimated, since

$\frac{Z_j}{\hat{Z}_j} \geq \frac{1}{n_j} e^{x_i(q)}$, and its value is averaged with the others, so the estimated numerator can even point opposite to the true one, as the next example illustrates.

Example 4.2 (Dilution in the estimate)**.** Take the cluster C_1 of Example 4.1 and the query $q = 5\,(\cos 45°, \sin 45°)$. The keys have logits 7.07, 3.83 and 3.21, so k_1 holds 94% of the mass of the cluster, whereas the centroid logit is 4.71. Let $v_1 = u$ and $v_2 = v_3 = -u$ for a unit vector u. The true numerator is $N_1 = (e^{7.07} - e^{3.83} - e^{3.21})\,u \approx 1106\,u$, whereas the estimate gives each value the weight $e^{4.71} \approx 110$ and returns $\hat{N}_1 \approx -110\,u$, which points the opposite way. The estimated mass $\hat{Z}_1 \approx 332$ is also 3.8 times smaller than $Z_1 \approx 1248$.

Mismatch with the query geometry. The residual logit depends only on the component of $k_i - c_j$ along the query. Let Q denote the distribution of the queries of a head at decoding time, with second moment $\Sigma \triangleq \mathbb{E}_{q\sim Q}[qq^\top] = \sum_{l=1}^{d} \lambda_l u_l u_l^\top$. The average squared residual logit of a residual r is $\mathbb{E}_{q\sim Q}[\langle q, r\rangle^2] = \sum_{l=1}^{d} \lambda_l \langle u_l, r\rangle^2$, which weights each direction u_l by the energy λ_l of the queries along it, whereas the cosine distance weights all directions equally. The two weightings differ sharply. On continued natural text, the leading direction of Σ is the mean query, with $|\cos(u_1, \mathbb{E}q)| \geq 0.999$, and it carries 64% of the trace for both models, so a residual along it costs about 80 times more than along an average direction. As a result, the angle between a key and its centroid predicts residual logit poorly. For the partition of RetroInfer, the rank correlation between the two is only 0.37 to 0.47, and keys at the same angular distance have residual logits that differ by about an order of magnitude. Appendix D reports these measurements, which hold across regimes.

4.2 Score-Space Metric

Robust objective. We choose the partition so that the residual logits are small on average over the decoding queries. For a key $k_i \in C_j$, $\mathbb{E}_{q\sim Q}[x_i(q)^2] = \|k_i - c_j\|_\Sigma^2$, which depends on Q only through its second moment Σ. We estimate Σ from the prefill queries as described in Appendix B, and denote the estimate by Σ as well. The estimate is built from a few hundred queries in d dimensions, so it is noisy and nearly singular along the directions of low energy, where a residual costs almost nothing, as measured in Appendix B. We therefore minimize the worst case over all second moments within spectral distance $\rho\bar{\lambda}$ of the estimate, where the mean eigenvalue $\bar{\lambda} = \frac{1}{d}\operatorname{tr}\Sigma$ makes ρ invariant to the scale of the queries:

$$\min_{\mathcal{C}} \max_{\Sigma' \in \mathcal{U}} \sum_{j=1}^{m} \sum_{i \in C_j} \|k_i - c_j\|_{\Sigma'}^2, \qquad \mathcal{U} = \big\{\Sigma' \succeq 0 : \|\Sigma' - \Sigma\|_2 \leq \rho\bar{\lambda}\big\}, \tag{3}$$

with $c_j = \frac{1}{n_j}\sum_{i\in C_j} k_i$. The inner maximum has a closed form.

Lemma 4.3 (Worst-case metric)**.** *For every positive semidefinite matrix $M \in \mathbb{R}^{d\times d}$,*

$$\max_{\Sigma' \in \mathcal{U}} \operatorname{tr}(\Sigma' M) = \operatorname{tr}(\Sigma_\rho M), \qquad \Sigma_\rho \triangleq \Sigma + \rho\bar{\lambda} I,$$

and the maximum is attained at $\Sigma' = \Sigma_\rho$ for every M.

The proof is given in Appendix A.1. Since the maximizer Σ_ρ does not depend on the partition, problem equation 3 reduces to k-means on the whitened keys $\Sigma_\rho^{1/2} k_i$, which Lloyd's algorithm solves. This metric weights each direction u_l of a residual by $\lambda_l + \rho\bar{\lambda}$, the energy of the queries along it plus a floor that keeps every direction penalized. It also retains the norm of the keys, so it penalizes the residual $k_1 - c_1$ behind both examples of Section 4.1.

Effect on the estimate. The same objective controls both errors of the estimate, uniformly over the uncertainty set.

Proposition 4.4 (Estimation error)**.** *Let C_j be a cluster whose centroid c_j is the arithmetic mean of its n_j keys. For a query q, let $\delta(q) = \max_{i\in C_j} |x_i(q)|$ and $D_j(q) = \frac{1}{n_j}\sum_{i\in C_j} x_i(q)^2$, and let $U_j = \frac{1}{n_j}\sum_{i\in C_j} \|v_i - \bar{v}_j\|_2^2$. Then*

$$\log\big(1 + \tfrac{1}{2} e^{-\delta(q)} D_j(q)\big) \leq \log\frac{Z_j}{\hat{Z}_j} \leq \tfrac{1}{2} e^{\delta(q)} D_j(q), \tag{4}$$

$$N_j - \hat{Z}_j \bar{v}_j = (Z_j - \hat{Z}_j)\,\bar{v}_j + R_j, \qquad \|R_j\|_2 \leq \hat{Z}_j\, e^{\delta(q)} \sqrt{D_j(q)\, U_j}. \tag{5}$$

Moreover, for every query distribution Q' whose second moment lies in $\mathcal{U}$,

$$\mathbb{E}_{q\sim Q'}[D_j(q)] \le \frac{1}{n_j}\sum_{i\in C_j}\|k_i - c_j\|^2_{\Sigma_\rho}. \tag{6}$$

The proof is given in Appendix A.2, which also extends the bounds to centroids other than the mean. Up to the factor $e^{\pm\delta(q)}$, the mass error equation 4 is half the dispersion $D_j(q)$ of the residual logits. The error of the numerator equation 5 adds a remainder R_j of at most $e^{\delta(q)}\sqrt{D_j(q)U_j}$ relative to $\hat{Z}_j$, which captures the averaging of the values and the reversal of Example 4.2. By equation 6 and Jensen's inequality, the averages of $D_j(q)$ and $\sqrt{D_j(q)}$ under every query distribution of the uncertainty set are bounded through the within-cluster term of equation 3, so minimizing equation 3 reduces both errors.

The factor $e^{\delta(q)}$, however, makes these bounds loose when one key has a large residual logit, which is the dilution regime of Section 4.1. The robust metric does not target this regime: Σ averages over all queries, so a direction used only by the few queries that attend to a key carries little weight in it. Section 4.3 reweights the objective toward these queries.

4.3 Weighting Toward Retrieval Targets

Two roles of a key. A key reaches the output either exactly, when its cluster is retrieved, or through its centroid, when its cluster is estimated. When $\langle q, k_i\rangle$ is high, k_i should be retrieved, and dilution can exclude its cluster, so its residual logit deserves a larger weight. When $\langle q, k_i\rangle$ is low, k_i enters the output through the estimate, and the robust objective already keeps its residual logit small on average. We therefore add a term that targets the keys of the first kind, which requires the importance of each key as a retrieval target and the direction of the queries that attend to it.

Importance and axis. We compute both quantities under the Gaussian model $q \sim \mathcal{N}(\mu, \Sigma_{0,\rho})$, where μ is the mean of the queries and $\Sigma_{0,\rho} \triangleq \Sigma_\rho - \mu\mu^\top$, so that the model has the second moment Σ_ρ of Section 4.2. Softmax attention does not change when a common vector is added to every key, so we measure the importance of k_i by its expected attention weight relative to the mean key $\bar{k}$ of its segment. We use this weight only relative to its median τ over the segment:

$$\log\omega_i \triangleq \log\mathbb{E}\big[e^{\langle q, k_i-\bar{k}\rangle}\big] = \langle\mu, k_i-\bar{k}\rangle + \tfrac{1}{2}\|k_i-\bar{k}\|^2_{\Sigma_{0,\rho}}, \qquad s_i = \max\big\{\tanh\big(\tfrac{1}{2}(\log\omega_i - \tau)\big),\, 0\big\}.$$

The importance s_i vanishes for the keys at or below the median and saturates at 1. An error in the magnitude of ω_i, which depends on the tail of the logit, therefore cannot raise the weight of a key above one. Reweighting the model by $e^{\langle q, k_i-\bar{k}\rangle}$ gives the distribution of the queries that attend to k_i, whose mean is $m_i = \mu + \Sigma_{0,\rho}(k_i - \bar{k})$. The inner product $\langle m_i, k_i - c_j\rangle$ is then the residual logit of k_i at this mean query, which is the quantity that dilution measures. Appendix C discusses these design choices and checks them against the actual decoding queries.

Assignment cost. In the whitened coordinates $\tilde{k}_i = \Sigma_\rho^{1/2}(k_i - \bar{k})$ and $\tilde{c} = \Sigma_\rho^{1/2}(c - \bar{k})$, with the unit axis $\hat{m}_i = \frac{\Sigma_\rho^{-1/2} m_i}{\|\Sigma_\rho^{-1/2} m_i\|_2}$, the cost of assigning k_i to the centroid c is

$$\mathrm{cost}(k_i, c) = \|\tilde{k}_i - \tilde{c}\|^2 + \gamma\, s_i \langle\hat{m}_i, \tilde{k}_i - \tilde{c}\rangle^2, \tag{7}$$

where $\gamma > 0$. The first term is the robust metric $\|k_i - c\|^2_{\Sigma_\rho}$ and protects every key. The second penalizes the residual logit of important keys along the direction of the queries that attend to them. By the Cauchy–Schwarz inequality, the second term never exceeds γs_i times the first, so the penalty along the axis is at most $1+\gamma$ times the penalty across it. The partition minimizes the total cost $\sum_j\sum_{i\in C_j}\mathrm{cost}(k_i, c_j)$ over the assignment and the centroids, which reduces to the robust objective equation 3 for $\gamma = 0$.

Optimization. We minimize the total cost by Lloyd's algorithm, with the cost equation 7 in both steps. Given the assignment, the cost of a cluster is a positive definite quadratic function of its centroid, whose minimizer solves

$$G_j\tilde{c}_j = \sum_{i\in C_j}\big(\tilde{k}_i + \gamma s_i\langle\hat{m}_i, \tilde{k}_i\rangle\hat{m}_i\big), \qquad G_j = n_j I + \gamma\sum_{i\in C_j} s_i\hat{m}_i\hat{m}_i^\top. \tag{8}$$

Both steps minimize the same objective, which therefore does not increase along the iterations. The centroid equation 8 is not the mean of the keys: it moves toward the important keys along their axes. Since $s_i \leq 1$, the condition number of G_j is at most $1 + \gamma$, so six iterations of conjugate gradient reduce the error of equation 8 by a factor of more than 10^3 for $\gamma = 2$, independently of the size of the cluster. The index stores n_j, $c_j = \Sigma_\rho^{-1/2}\tilde{c}_j + \bar{k}$ and $n_j\bar{v}_j$ as in Section 3.3, so existing decoding kernels read it without modification. Algorithm 1 in Appendix C gives the complete procedure.

5 Experimental Results

Section 5.1 evaluates SIEVE on long-context benchmarks, and Section 5.2 evaluates it on reasoning benchmarks with long generation. Section 5.3 isolates the contribution of each component of SIEVE. Due to space constraints, Appendix J reports the latency and memory of every method.

Setup. We implement our method and all baselines in a single sparse-attention harness built on the ShadowKV codebase (Sun et al., 2024)[1]. Every method shares the same model forward pass, the same KV storage, and the same decoding loop, so the methods differ only in how they index the cache and select tokens at each decoding step. For ParisKV and RetroInfer, we call the official implementations released by their authors inside this harness. No method evicts tokens, and all methods share the layout of Figure 1a, with $n_{\text{sink}} = 32$ and $n_{\text{flush}} = 256$. The budget B counts only the retrieved tokens. Appendix E gives n_{win} and the decoding settings of every benchmark.

Baselines. We compare SIEVE against four sparse-attention methods, Quest (Tang et al., 2024), ShadowKV (Sun et al., 2024), RetroInfer (Chen et al., 2025a), and ParisKV (Qi et al., 2026), and against full attention over the entire cache. Every baseline runs with the same sink, recent window, indexing interval, and budget B as SIEVE, and keeps the remaining hyperparameters released by its authors. ShadowKV builds its low-rank key cache from the prefill and is not designed for long generation, a limitation also reported by Qi et al. (2026). We therefore evaluate ShadowKV on the long-context benchmarks only.

Models. On the long-context benchmarks, we use Llama-3.1-8B-Instruct[2] and Qwen3-4B-Instruct-2507[3]. The reasoning benchmarks require long chains of thought, so we replace Llama-3.1-8B-Instruct with DeepSeek-R1-Distill-Llama-8B[4], which shares the architecture of Llama-3.1-8B and is distilled for long-form reasoning. All scores are scaled by 100.

5.1 Long-Context Benchmarks

Benchmarks. RULER (Hsieh et al., 2024) is a synthetic suite of 13 subtasks that retrieve, trace, or aggregate information spread over a long context. We sweep the context length over $\{16, 32, 64, 128\}$K with 100 samples per subtask, and scale the budget with the context: B is $\frac{1}{32}$ of the context on Llama-3.1-8B-Instruct and $\frac{1}{128}$ on Qwen3-4B-Instruct-2507. Qwen3-4B-Instruct-2507 receives the smaller fraction because every method matches full attention on it at $\frac{1}{32}$. LongBench (Bai et al., 2024) contains 16 tasks over real documents, each scored with its official metric. Following the KV Press protocol (Devoto et al., 2024), we truncate each context to its first 65,536 tokens and append the question. We report the macro-average over 14 tasks and exclude `lcc` and `multi_news`, whose prompts are short enough for the exact regions and the budget to cover most of them.

Results. Table 1 shows that SIEVE is the best sparse method in every column. On RULER, it stays within 0.3 points of full attention for every length and model, and exceeds it at 128K on Qwen3-4B-Instruct-2507, with 83.0 against 82.2. The strongest baseline, in contrast, trails full attention by 0.5 to 2.1 points. Since SIEVE and RetroInfer use the same retrieval and the same estimate, the gap between them isolates the effect of the partition: SIEVE improves on RetroInfer in every RULER setting, by 1.0 to 1.9 points on Llama-3.1-8B-Instruct and by 0.8 to 1.6 points on Qwen3-4B-Instruct-2507. On LongBench, every method stays within 0.5 points of full attention at $B = 256$,

[1] https://github.com/bytedance/ShadowKV

[2] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

[3] https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507

[4] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B

Table 1: Macro-average scores (↑) on RULER and LongBench, at the budgets of Section 5.1. Bold indicates the best sparse method in each column. Per-task scores are given in Appendices F and G.

	Llama-3.1-8B-Instruct					Qwen3-4B-Instruct-2507				
	RULER				LongBench	RULER				LongBench
Method	16K	32K	64K	128K		16K	32K	64K	128K	
Full attention	93.0	88.7	86.2	77.7	45.7	94.9	94.3	90.2	82.2	44.4
SIEVE	**93.0**	**88.9**	**85.9**	**77.5**	**45.8**	**94.7**	**94.2**	**90.1**	**83.0**	**44.4**
Quest	89.1	86.8	83.9	74.5	45.4	86.2	87.2	81.7	74.0	44.0
ShadowKV	90.4	87.3	83.6	74.1	45.5	91.0	91.5	86.1	77.2	44.2
RetroInfer	91.7	87.9	84.8	75.6	45.3	93.9	93.2	88.5	81.7	44.1
ParisKV	91.4	87.9	84.7	74.8	45.2	93.7	93.8	89.5	79.7	44.3

Table 2: Accuracy (↑) on three groups of RULER subtasks: `niah_multikey_3` (MK3), variable tracking (VT), and the average of common-word and frequent-word extraction (AGG). Bold indicates the best sparse method in each column, and is omitted when all sparse methods tie.

	MK3				VT				AGG			
Method	16K	32K	64K	128K	16K	32K	64K	128K	16K	32K	64K	128K
Llama-3.1-8B-Instruct												
Full attention	100.0	100.0	98.0	63.0	100.0	99.0	97.4	69.4	79.9	55.7	43.4	35.5
SIEVE	**100.0**	**100.0**	**96.0**	**57.0**	**99.6**	**98.6**	**96.2**	**73.0**	**80.1**	**57.5**	**43.9**	**35.5**
Quest	99.0	97.0	84.0	34.0	98.0	95.4	87.6	70.2	56.5	46.5	43.4	31.0
ShadowKV	**100.0**	**100.0**	**96.0**	47.0	96.8	90.6	82.4	62.2	65.4	52.4	36.4	30.5
RetroInfer	98.0	**100.0**	93.0	46.0	**99.6**	98.2	**96.2**	67.2	73.6	50.8	40.6	35.4
ParisKV	**100.0**	97.0	94.0	40.0	98.0	96.4	89.6	62.6	71.5	54.1	42.5	33.0
Qwen3-4B-Instruct-2507												
Full attention	98.0	100.0	100.0	88.0	100.0	100.0	99.0	91.8	95.2	90.4	72.1	49.3
SIEVE	98.0	**100.0**	**100.0**	**90.0**	**100.0**	**100.0**	98.8	92.2	**94.3**	**89.2**	69.7	**50.4**
Quest	38.0	58.0	36.0	12.0	99.2	**100.0**	**99.0**	87.6	75.2	70.2	50.3	43.4
ShadowKV	98.0	99.0	79.0	32.0	**100.0**	99.6	96.8	86.0	73.8	75.1	57.7	44.9
RetroInfer	98.0	**100.0**	**100.0**	87.0	**100.0**	**100.0**	98.8	**93.6**	90.0	83.6	60.7	45.2
ParisKV	**99.0**	**100.0**	98.0	59.0	**100.0**	**100.0**	98.8	88.4	88.4	87.5	**70.5**	49.3

and SIEVE matches it on both models, so we read this benchmark as showing parity rather than as ranking the methods.

Retrieval, tracing, and aggregation. The macro-average of RULER is compressed by the subtasks on which every method is nearly exact. Table 2 therefore isolates one group of subtasks from each category of RULER, except question answering, whose scores are limited by the model rather than by the index, as discussed in Appendix F.

On `niah_multikey_3`, which hides the target among keys that resemble it, the methods separate from 64K on. At 128K, SIEVE is the best sparse method on both models, and it improves on RetroInfer by 11 points on Llama-3.1-8B-Instruct and by 3 points on Qwen3-4B-Instruct-2507, where it also exceeds full attention. This subtask is where dilution is most likely, since angular clustering can group the target with the keys that resemble it and score the cluster by their mean. The target term of Section 4.3 counters this effect, which is consistent with the ablation of Section 5.3, where it raises the accuracy on this subtask from 97 to 100. Variable tracking depends on a few specific tokens and behaves like retrieval. At 128K, SIEVE is the only method at or above full attention on both models, and it improves on RetroInfer by 5.8 points on Llama-3.1-8B-Instruct.

Table 3: Results on AIME-25 (Mean@4) and GPQA Diamond (accuracy). Full attention uses the entire cache. Bold indicates the highest reported sparse score for each benchmark, model, and budget, including ties.

	Qwen3-4B-Instruct-2507				DeepSeek-R1-Distill-Llama-8B			
	AIME-25		GPQA Diamond		AIME-25		GPQA Diamond	
Method	$B = 128$	$B = 256$	$B = 128$	$B = 256$	$B = 128$	$B = 256$	$B = 128$	$B = 256$
Full attention	47.5		55.6		32.5		49.5	
SIEVE	**50.0**	**49.2**	**61.6**	**60.6**	**33.4**	**33.4**	**49.0**	**50.5**
Quest	35.8	43.3	53.0	**60.6**	25.4	26.7	47.0	48.5
RetroInfer	49.2	47.5	60.6	59.6	29.2	31.7	45.4	46.5
ParisKV	45.8	46.7	55.1	55.1	26.7	30.8	47.0	48.0

On aggregation, SIEVE improves on RetroInfer at every length on both models, by up to 9.0 points, and it matches or exceeds full attention at every length on Llama-3.1-8B-Instruct. Aggregation spreads the attention mass over more tokens than retrieval, as measured on Qwen3-4B-Instruct-2507 in Appendix I, so we expect a larger share of the output to pass through the estimate. Since SIEVE and RetroInfer share the estimator, the gain comes from the partition. The ablation of Section 5.3 confirms that the partition of SIEVE lowers the estimation error by a third, which is consistent with Proposition 4.4. The only case where a baseline exceeds SIEVE on aggregation is ParisKV on Qwen3-4B-Instruct-2507 at 64K, by 0.8 points.

5.2 Reasoning Benchmarks

Benchmarks. AIME-25 contains the 30 problems of the American Invitational Mathematics Examination 2025. We draw 4 samples per problem with up to 32,768 generated tokens and report Mean@4. GPQA (Rein et al., 2023) contains the 198 multiple-choice questions of its Diamond split, which require graduate-level scientific reasoning. We draw one sample per question with the same generation limit and report accuracy. Both benchmarks run at budgets $B = 128$ and $B = 256$.

Results. Table 3 reports results on both benchmarks. SIEVE achieves the highest sparse score on both models at both budgets, tying with Quest on GPQA Diamond for Qwen3-4B-Instruct-2507 at $B = 256$. Compared with RetroInfer, SIEVE improves AIME-25 Mean@4 by up to 4.2 points and GPQA Diamond accuracy by up to 4.0 points, both on DeepSeek-R1-Distill-Llama-8B. SIEVE exceeds full attention in seven of the eight settings; the remaining setting, GPQA Diamond on DeepSeek at $B = 128$, is only 0.5 points lower. Its scores remain similar across budgets, supporting the preservation of reasoning performance under limited retrieval budgets.

5.3 Ablation Study

We ablate the components of SIEVE on Qwen3-4B-Instruct-2507 at 32K with $B = 128$. The query-induced metric improves index quality but requires ridge regularization to preserve retrieval accuracy. The target term yields the largest reduction in estimation error and the best macro-average accuracy. Appendix H presents the setup and full results.

6 Conclusion

We introduced SIEVE, a query-aware clustering framework that improves sparse attention by grouping keys in the logit space induced by queries. Our analysis characterized how centroid approximation underestimates the contributions of important keys and established bounds on errors in estimating attention mass and the attention numerator. These insights motivated a robust query-aware clustering objective that places additional emphasis on important retrieval targets. Experiments on long-context benchmarks demonstrate that SIEVE outperforms sparse attention baselines while maintaining accuracy close to full attention, highlighting the value of query-aware clustering for both retrieval and attention estimation.

AI USE STATEMENT

We used generative AI tools as assistants for writing and editing, for software development and code review, for the analysis of experimental results. The research idea is entirely the authors' own. The authors reviewed and verified all AI-assisted work and take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

Appendix A gives the complete proofs of Lemma 4.3 and Proposition 4.4, including the extension of the bounds to centroids other than the mean. Appendix B specifies how the query moments are estimated from the prefill queries, and Appendix C gives the complete clustering procedure in Algorithm 1. Appendix E lists the evaluation protocol of every benchmark, namely the prompts, context lengths, truncation, budgets, and decoding settings, together with the hyperparameters of SIEVE, which are shared by all models and benchmarks. All methods run in one harness with the same forward pass and decoding loop, and the baselines use the official implementations of their authors where available. Appendices F, G, and H report the per-task results and the ablation, and Appendix J describes the efficiency measurements.

ETHICS STATEMENT

This work studies the index of sparse attention for long-context inference. It introduces no new data collection, involves no human subjects, and is evaluated entirely on public benchmarks and pretrained models under their respective licenses. SIEVE requires no training and leaves the pretrained weights unchanged. It changes only how the keys of the cache are grouped, which determines the tokens that each query reads exactly. Like every sparse attention method, it approximates full attention and can therefore change the outputs of the model, in particular when a token that a query needs is not retrieved. We therefore compare every method with full attention on each task in Appendices F and G, including the settings in which SIEVE falls below it.

REFERENCES

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 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)*, pages 3119–3137, 2024.

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.

Yaoqi Chen, Jinkai Zhang, Baotong Lu, Qianxi Zhang, Chengruidong Zhang, Jing Liu, Jingjia Luo, Di Liu, Huiqiang Jiang, Qi Chen, et al. Retroinfer: A vector storage engine for scalable long-context llm inference. *arXiv preprint arXiv:2505.02922*, 2025a.

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. In *International Conference on Learning Representations*, volume 2025, pages 44169–44190, 2025b.

Xinyan Dai, Xiao Yan, Kelvin KW Ng, Jiu Liu, and James Cheng. Norm-explicit quantization: Improving vector quantization for maximum inner product search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 51–58, 2020.

Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective l_2 norm-based strategy for kv cache compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18476–18499, 2024.

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.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *Advances in Neural Information Processing Systems*, 38:113152–113188, 2026.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7376–7399, 2025.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, pages 3887–3896. PMLR, 2020.

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.

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. {InfiniGen}: Efficient generative inference of large language models with dynamic {KV} cache management. In *18th USENIX symposium on operating systems design and implementation (OSDI 24)*, pages 155–172, 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.

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)*, pages 1–7. IEEE, 2025.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024a.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024b.

J Nocedal and S Wright. Numerical optimization, 2006.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are multi-state rnns. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18724–18741, 2024.

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.

Yanlin Qi, Xinhang Chen, Huiqiang Jiang, Qitong Wang, Botao Peng, and Themis Palpanas. Pariskv: Fast and drift-robust kv-cache retrieval for long-context llms. *arXiv preprint arXiv:2602.07721*, 2026.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. Shadowkv: Kv cache in shadows for high-throughput long-context llm inference. *arXiv preprint arXiv:2410.21465*, 2024.

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.

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.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. *Advances in neural information processing systems*, 37:119638–119661, 2024a.

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, pages 21875–21895, 2024b.

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, 2025.

Jin Zhang, Defu Lian, Haodi Zhang, Baoyun Wang, and Enhong Chen. Query-aware quantization for maximum inner product search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4875–4883, 2023a.

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, 2023b.

A PROOFS

A.1 PROOF OF LEMMA 4.3

Proof of Lemma 4.3. Let $\Sigma' \in \mathcal{U}$ and $E = \Sigma' - \Sigma$, so that $\|E\|_2 \le \rho\bar{\lambda}$. Since M is positive semidefinite, we have $\operatorname{tr}(EM) = \operatorname{tr}(M^{1/2}EM^{1/2})$, and $M^{1/2}EM^{1/2} \preceq \|E\|_2 M$ because $E \preceq \|E\|_2 I$. Taking the trace gives $\operatorname{tr}(EM) \le \|E\|_2 \operatorname{tr}(M) \le \rho\bar{\lambda}\operatorname{tr}(M)$, and therefore $\operatorname{tr}(\Sigma' M) \le \operatorname{tr}(\Sigma M) + \rho\bar{\lambda}\operatorname{tr}(M) = \operatorname{tr}(\Sigma_\rho M)$. The matrix Σ_ρ is positive semidefinite and satisfies $\|\Sigma_\rho - \Sigma\|_2 = \rho\bar{\lambda}$, so it lies in $\mathcal{U}$ and attains the bound. This completes the proof. □

A.2 PROOF OF PROPOSITION 4.4

Proof of Proposition 4.4. Write $x_i = x_i(q)$, $\delta = \delta(q)$ and $D_j = D_j(q)$. The expressions of Z_j and $\hat{Z}_j$ in Section 4.1 give

$$Z_j = e^{\langle q, c_j\rangle} \sum_{i \in C_j} e^{x_i}, \qquad \hat{Z}_j = n_j e^{\langle q, c_j\rangle}, \qquad \frac{Z_j}{\hat{Z}_j} = \frac{1}{n_j} \sum_{i \in C_j} e^{x_i}.$$

For equation 4, the centroid is the mean of the keys, so $\sum_{i \in C_j} x_i = 0$ and

$$\frac{Z_j}{\hat{Z}_j} = 1 + \frac{1}{n_j} \sum_{i \in C_j} (e^{x_i} - 1 - x_i).$$

By Taylor's theorem with the Lagrange remainder, $\frac{x^2}{2} e^{-|x|} \le e^x - 1 - x \le \frac{x^2}{2} e^{|x|}$ for every x. Summing over $i \in C_j$ and using $|x_i| \le \delta$ gives $1 + \frac{1}{2}e^{-\delta} D_j \le \frac{Z_j}{\hat{Z}_j} \le 1 + \frac{1}{2}e^{\delta} D_j$. Taking logarithms yields the lower bound, and the upper bound follows from $\log(1+u) \le u$ for $u \ge 0$.

For equation 5, since $N_j = e^{\langle q, c_j\rangle} \sum_{i \in C_j} e^{x_i} v_i$ and $\hat{Z}_j \bar{v}_j = e^{\langle q, c_j\rangle} \sum_{i \in C_j} v_i$, we have

$$N_j - \hat{Z}_j \bar{v}_j = e^{\langle q, c_j\rangle} \sum_{i \in C_j} (e^{x_i} - 1)(v_i - \bar{v}_j) + e^{\langle q, c_j\rangle} \sum_{i \in C_j} (e^{x_i} - 1)\, \bar{v}_j.$$

The second term equals $(Z_j - \hat{Z}_j)\bar{v}_j$, and we let R_j denote the first. Since $|e^x - 1| \le |x| e^{|x|}$ for every x, the Cauchy–Schwarz inequality gives

$$\begin{aligned} \|R_j\|_2 &\le e^{\langle q, c_j\rangle} e^{\delta} \sum_{i \in C_j} |x_i|\, \|v_i - \bar{v}_j\|_2 \\ &\le e^{\langle q, c_j\rangle} e^{\delta} \Big(\sum_{i \in C_j} x_i^2 \Big)^{1/2} \Big(\sum_{i \in C_j} \|v_i - \bar{v}_j\|_2^2 \Big)^{1/2} = \hat{Z}_j e^{\delta} \sqrt{D_j U_j}. \end{aligned}$$

Finally, for a query distribution Q' with second moment Σ', linearity of the expectation gives $\mathbb{E}_{q \sim Q'}[D_j(q)] = \operatorname{tr}(\Sigma' M_j)$ with $M_j = \frac{1}{n_j} \sum_{i \in C_j} (k_i - c_j)(k_i - c_j)^\top$. Lemma 4.3 bounds this quantity by $\operatorname{tr}(\Sigma_\rho M_j)$ whenever $\Sigma' \in \mathcal{U}$, which is equation 6. This completes the proof. □

Centroids other than the mean. The proofs of equation 5 and equation 6 do not use that c_j is the mean of the keys, so both bounds hold for every centroid. For equation 4, let c_j be any centroid and let $\bar{x}_j = \frac{1}{n_j} \sum_{i \in C_j} x_i$ be the average residual logit, which vanishes only for the mean. The shifted residual logits $y_i = x_i - \bar{x}_j$ sum to zero, satisfy $|y_i| \le 2\delta$, and have the average square $\tilde{D}_j = \frac{1}{n_j} \sum_{i \in C_j} y_i^2 = D_j - \bar{x}_j^2$, which is the variance of the residual logits. Since $\frac{Z_j}{\hat{Z}_j} = e^{\bar{x}_j} \frac{1}{n_j} \sum_{i \in C_j} e^{y_i}$, the argument for equation 4 applied to the y_i gives

$$\log\big(1 + \tfrac{1}{2} e^{-2\delta} \tilde{D}_j\big) \le \log \frac{Z_j}{\hat{Z}_j} - \bar{x}_j \le \tfrac{1}{2} e^{2\delta} \tilde{D}_j.$$

For a centroid other than the mean, the log-ratio thus gains the first-order term $\bar{x}_j(q) = \big\langle q, \frac{1}{n_j} \sum_{i \in C_j} k_i - c_j \big\rangle$, which measures how far the centroid lies from the mean of its keys along q.

B Estimation of the Query Moments

Keys and queries are taken after the rotary position embedding, and the index is read by queries issued after the prompt. Let P be the length of the prompt and $R(p)$ the rotary matrix at position p. We keep the last T_0 prefill queries of each query head, pool the query heads that share a key-value head, and undo the rotation of each query at its own position, $\tilde{q} = R(p)^\top q_p$. Let $\tilde{\mu}$ and $\tilde{S}$ denote the mean and the second moment of these rotation-free queries. We place them at the next W decoding positions by the mixture

$$\mu = \frac{1}{W}\sum_{\delta=1}^{W} R(P+\delta)\,\tilde{\mu}, \qquad \Sigma = \frac{1}{W}\sum_{\delta=1}^{W} R(P+\delta)\,\tilde{S}\,R(P+\delta)^\top,$$

so that $r^\top \Sigma r$ is the average squared logit of a residual r over the queries and the positions of the window. Both moments come from the same sample, so $\tilde{S} \succeq \tilde{\mu}\tilde{\mu}^\top$. Averaging this inequality over the positions and applying Jensen's inequality to the convex map $x \mapsto xx^\top$ gives $\Sigma \succeq \mu\mu^\top$, so the matrix $\Sigma_{0,\rho} = \Sigma - \mu\mu^\top + \rho\bar{\lambda} I$ of Section 4.3 is positive definite. We use $T_0 = 100$ and $W = 128$.

Quality of the estimate. We measure how well Σ_ρ predicts the residual logits of the queries that read the index. For a residual r and a set of queries, we compare the realized and the predicted squared residual logit through the ratio

$$R(r) = \frac{\frac{1}{T}\sum_{t=1}^{T}\langle q_t, r\rangle^2}{r^\top \Sigma_\rho r},$$

where $q_1, \ldots, q_T$ are the queries of the set. We take the residuals $r = k_i - c_j$ of the partition of RetroInfer, so that the measurement does not depend on our objective. We use two sets of queries on 16 prompts of CWE and of `niah_multikey_3` at 32K: the decoding queries, and as a reference, the 100 prefill queries that precede the estimation window, which the estimate does not use. Table 4 reports the share of residuals with $R(r) > 1$.

Without ridge, about half of the residuals have $R(r) > 1$, and the median of $\log_{10} R(r)$ lies between -0.03 and -0.02 in every setting, so the estimate is unbiased in the median. Decoding queries are no harder to predict than held-out prefill queries, with a share of 44% to 46% against 46% to 54%. They also stay within the directions covered by the estimate. The mean of the decoding queries has a cosine of 0.92 to 0.94 with the estimated mean μ, and only 2.3% to 4.2% of their energy lies in the subspace that holds the smallest 1% of the trace of Σ, against 5.8% to 7.2% for the held-out prefill queries. The errors that matter come from a heavy tail: 0.27% to 0.43% of the pairs of query and residual have a squared residual logit ten times larger than predicted, against 0.16% for Gaussian queries, on both sets. A small ridge removes this tail. The share of residuals with $R(r) > 1$ falls to 3% to 6% at $\rho = 1$, below 1% at $\rho = 3$, and to zero at $\rho = 10$. Beyond this bulk, the decoding queries leave 1.2 to 2.6 times as many residuals with $R(r) > 1$ as the held-out prefill queries at $\rho = 1$, so their far tail is slightly heavier.

Table 4: Quality of the estimate of the query moments on RULER at 32K. The cosine compares the estimated mean μ with the mean μ_{dec} of the decoding queries. The uncovered energy is the share of the energy of the queries, in percent, that lies in the subspace holding the smallest 1% of the trace of the estimate Σ. The last five columns give the share of residuals, in percent, with $R(r) > 1$ for each value of the ridge ρ. MK3 is `niah_multikey_3`, and the held-out prefill queries are not used by the estimate.

					Share with $R(r) > 1$				
Model	Task	$\cos(\mu, \mu_{\mathrm{dec}})$	Queries	Uncovered energy	$\rho = 0$	$\rho = 1$	$\rho = 3$	$\rho = 10$	$\rho = 30$
Qwen3-4B-Instruct-2507	CWE	0.921	decoding	3.7	43.8	3.9	0.4	0.0	0.0
			held-out prefill	5.8	48.3	1.5	0.1	0.0	0.0
	MK3	0.929	decoding	2.3	45.8	4.2	0.4	0.0	0.0
			held-out prefill	6.9	45.5	2.3	0.1	0.0	0.0
Llama-3.1-8B-Instruct	CWE	0.935	decoding	4.2	43.7	3.3	0.1	0.0	0.0
			held-out prefill	7.2	53.6	2.8	0.1	0.0	0.0
	MK3	0.922	decoding	2.4	46.1	6.0	0.6	0.0	0.0
			held-out prefill	6.9	46.8	2.8	0.1	0.0	0.0

Choice of the ridge. The measurement above shows that the estimate is sound, but it does not single out a value of ρ, since the failures that matter for the task are rare events that an average over residuals does not see. We therefore choose ρ on the task. Table 5 varies ρ with every other component of SIEVE fixed, in the stress setting of Section 5.3. The ridge trades aggregation for retrieval. Without it, `niah_multikey_3` drops to 65.0, although CWE reaches 78.5. Small values recover most of this loss, with 97.0 at $\rho = 1$ and 99.0 at $\rho = 3$. From $\rho = 5$ on, `niah_multikey_3` is solved on every sample, and CWE stays between 74.4 and 73.2 up to $\rho = 10$ before it falls to 69.9 at $\rho = 30$. On a failing sample at small ρ, the needle falls into a cluster of about 20 unrelated keys in the head that copies it, and the score of this cluster drops below the budget. We set $\rho = 10$ inside the region where `niah_multikey_3` is solved, rather than at its boundary, and keep it fixed across models, context lengths, and benchmarks.

Table 5: Accuracy (↑) of SIEVE for each value of the ridge ρ, on Qwen3-4B-Instruct-2507 at 32K with $B = 128$. The shaded column is the value used in all experiments.

Task	$\rho = 0$	$\rho = 1$	$\rho = 3$	$\rho = 5$	$\rho = 7$	$\rho = 10$	$\rho = 30$
CWE	78.5	**80.6**	76.6	74.4	74.0	73.2	69.9
`niah_multikey_3`	65.0	97.0	99.0	**100.0**	**100.0**	**100.0**	**100.0**

C Details of SIEVE

Design choices of the target term. The Gaussian model of Section 4.3 serves only to give the importance and the axis in closed form. Among the distributions with mean μ and second moment Σ_ρ, it has the largest entropy. The importance ω_i is an expectation of an exponential, so its magnitude is governed by the tail of the logit, which the Gaussian model does not capture. The queries are indeed heavier-tailed than the model: Appendix B finds that 0.27% to 0.43% of the pairs of query and residual have a squared residual logit ten times larger than predicted, against 0.16% for Gaussian queries. We therefore use ω_i only through its position relative to the median of the segment, and the saturation of s_i at 1 bounds the influence of any error in its magnitude. The axis m_i is the gradient of $\log \omega_i$ with respect to k_i, and it equals q itself for the single query of Example 4.1, where $\langle q, k_1 - c_1 \rangle = x_1$. We normalize the axis in the whitened space rather than weight it by $\|\Sigma_\rho^{-1/2} m_i\|_2^2$, since this length grows with $\|k_i - \bar{k}\|$ and would make the ratio between the penalties along and across the axis unbounded.

Accuracy of the Gaussian model. We compare the importance and the axis of the Gaussian model with their empirical counterparts over the decoding queries $q_1, \ldots, q_T$,

$$w_i = \frac{1}{T} \sum_{t=1}^{T} e^{\langle q_t, k_i - \bar{k} \rangle}, \qquad \bar{m}_i = \frac{\sum_{t=1}^{T} e^{\langle q_t, k_i - \bar{k} \rangle} q_t}{\sum_{t=1}^{T} e^{\langle q_t, k_i - \bar{k} \rangle}},$$

on the prompts of Appendix B. Table 6 reports four quantities for each prompt, layer, and key-value head. The first is the rank correlation between $\log \omega_i$ and $\log w_i$, and the second is the share of keys that fall on the same side of the segment median under both. The third is the absolute difference $|\log \omega_i - \log w_i|$, and the fourth is the cosine between the whitened axes $\Sigma_\rho^{-1/2} m_i$ and $\Sigma_\rho^{-1/2} \bar{m}_i$, averaged over the keys with $s_i > 0$ with the weights s_i.

Table 6: Accuracy of the Gaussian model against the decoding queries on RULER at 32K. The first two columns give the median and the interquartile range, the third gives the median and the 99th percentile, and the last gives the interquartile range. MK3 is `niah_multikey_3`.

Model	Task	Rank correlation	Same side of the median	$\lvert \log \omega_i - \log w_i \rvert$	Axis cosine
Qwen3-4B-Instruct-2507	CWE	0.855 [0.746, 0.922]	0.836 [0.775, 0.884]	0.79 / 5.4	[0.833, 0.913]
	MK3	0.826 [0.610, 0.919]	0.817 [0.701, 0.882]	0.87 / 6.7	[0.852, 0.912]
Llama-3.1-8B-Instruct	CWE	0.887 [0.800, 0.930]	0.857 [0.805, 0.890]	0.66 / 4.0	[0.745, 0.866]
	MK3	0.820 [0.609, 0.901]	0.818 [0.721, 0.869]	0.86 / 6.4	[0.720, 0.838]

The Gaussian model ranks the keys well. The rank correlation has a median of 0.82 to 0.89, and 82% to 86% of the keys fall on the same side of the median, against 50% for a random ranking. Its magnitudes, in contrast, are unreliable in the tail. The difference of the log-importances has a median of 0.66 to 0.87, about a factor of two, but a 99th percentile of 4.0 to 6.7, a factor of 55 to 800. The design of s_i accounts for this behavior, since it uses ω_i only relative to the median and saturates above it. The axis of the model is also well aligned with the empirical axis. In three quarters of the heads, their cosine exceeds 0.83 on Qwen3-4B-Instruct-2507 and 0.72 on Llama-3.1-8B-Instruct, against about $\frac{1}{\sqrt{d}} \approx 0.09$ for two random directions.

Conjugate gradient. Forming and factoring the $d \times d$ matrix G_j would cost $O(d^3)$ per cluster. Conjugate gradient only needs products with G_j, each of cost $O(n_j d)$. Since $s_i \le 1$, the eigenvalues of G_j lie in $[n_j, (1+\gamma)n_j]$, so its condition number satisfies $\kappa_j \le 1+\gamma$. After t iterations, the error in the norm induced by G_j is reduced by at least the factor $2\big(\frac{\sqrt{\kappa_j}-1}{\sqrt{\kappa_j}+1}\big)^t$ (Nocedal and Wright, 2006, Section 5.1), which is below 10^{-3} for $\gamma = 2$ and $t = 6$.

Estimate with the centroid of SIEVE. Since the centroid equation 8 is not the mean, the residual logits of a cluster no longer sum to zero. The bound equation 4 then holds for $\log \frac{Z_j}{\hat{Z}_j} - \bar{x}_j(q)$, where $\bar{x}_j(q)$ is the average residual logit of C_j, with $\delta(q)$ doubled and $D_j(q)$ replaced by the variance of the residual logits, as shown in Appendix A.2. The bounds equation 5 and equation 6 hold unchanged.

Procedure. Algorithm 1 summarizes the construction of SIEVE for one segment of one attention head. The indexed region is split into 16 segments as in Section 3.3, each segment is clustered independently with one cluster per 16 keys on average, and the centroids are initialized at evenly spaced keys. The per-key quantities s_i and $\hat{m}_i$ are computed once before the iterations, so an assignment costs one whitened distance and one inner product per centroid. Keys indexed during decoding are clustered in the same whitened space with the first term of equation 7 alone. The free parameters are ρ and γ, and their values, shared by all models and benchmarks, are given in Appendix E.

Algorithm 1 Index construction of SIEVE for one segment of one attention head

Input: keys $\{k_i\}$ and values $\{v_i\}$ of the segment, query moments μ and Σ, number of clusters m, parameters ρ and γ, number of iterations L
$\Sigma_\rho \leftarrow \Sigma + \frac{\rho}{d}(\operatorname{tr}\Sigma)I, \quad \Sigma_{0,\rho} \leftarrow \Sigma_\rho - \mu\mu^\top, \quad \bar{k} \leftarrow$ mean of $\{k_i\}$
for each key k_i **do**
  $\tilde{k}_i \leftarrow \Sigma_\rho^{1/2}(k_i - \bar{k}), \quad \log\omega_i \leftarrow \langle \mu, k_i - \bar{k}\rangle + \frac{1}{2}\|k_i - \bar{k}\|^2_{\Sigma_{0,\rho}}$
  $\tilde{m}_i \leftarrow \Sigma_\rho^{-1/2}\big(\mu + \Sigma_{0,\rho}(k_i - \bar{k})\big), \quad \hat{m}_i \leftarrow \frac{\tilde{m}_i}{\|\tilde{m}_i\|_2}$
end for
$\tau \leftarrow$ median of $\{\log\omega_i\}, \quad s_i \leftarrow \max\{\tanh(\frac{1}{2}(\log\omega_i - \tau)), 0\}$ for all i
Initialize the centroids $\{\tilde{c}_j\}_{j=1}^m$ at evenly spaced keys
for $\ell = 1$ to L **do**
  Assign each key k_i to $\arg\min_j \operatorname{cost}(k_i, c_j)$, with the cost equation 7
  Update each $\tilde{c}_j$ by solving equation 8 with conjugate gradient
end for
Output: n_j, $c_j = \Sigma_\rho^{-1/2}\tilde{c}_j + \bar{k}$ and $n_j\bar{v}_j$ for $j = 1, \dots, m$

D Query Geometry Across Regimes

Setup. We measure the query geometry of Section 4.1 in three regimes that differ in how the decoding queries arise. In the first, the model continues natural text from the ProLong pretraining mixture (Gao et al., 2025), with no question anywhere. In the second, the model answers the questions of LongBench (Bai et al., 2024) over real documents, with the prompt templates and the middle truncation of its protocol. In the third, the model answers one prompt from each of eight RULER tasks (Hsieh et al., 2024). Each regime has eight prompts of 32K tokens. The decoding queries are generated greedily for up to 256 tokens, and the question regimes stop at the end-of-sequence token after at least 16 tokens. We take the queries after the rotary embedding, scale them by $\frac{1}{\sqrt{d}}$, and pool the four query heads of each key-value head. We compute one second moment $\Sigma = \mathbb{E}[qq^\top]$

per prompt, layer and key-value head, for layers 4, 12, 20 and 28 of Qwen3-4B-Instruct-2507 and Llama-3.1-8B-Instruct. For the partition, we run the spherical k-means of RetroInfer on the centered keys of each prompt, with 16 segments, 10 iterations and one cluster per 16 keys.

Quantities. For the spectrum, we report the share $\frac{\lambda_1}{\operatorname{tr}\Sigma}$ of the leading direction and the number k_{90} of directions that hold 90% of the trace. For the partition, we report three quantities per head. The first is the rank correlation between the angular distance $1-\cos(k_i, c_j)$ and the expected squared residual logit $\|k_i - c_j\|_\Sigma^2$. The second is the spread of the residual logit within one decile of angular distance, measured as $\log_{10}$ of the ratio between its 95th and 5th percentiles and averaged over the deciles by the median. The third is the share of the residual energy $\sum_i \|k_i - c_j\|^2$ that lies in the subspace holding 90% of the trace of Σ, together with the value $\frac{k_{90}}{d}$ that an isotropic residual would give.

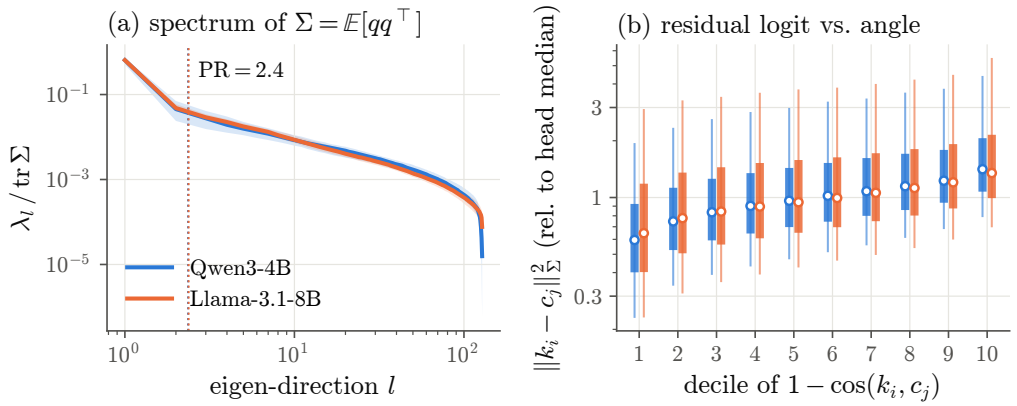

Figure 2: Geometry of the decoding queries on continued natural text, for layers 4, 12, 20 and 28 of Qwen3-4B-Instruct-2507 and Llama-3.1-8B-Instruct. (a) Normalized spectrum of the query second moment Σ, with the median over heads and the interquartile range. (b) Expected squared residual logit $\|k_i - c_j\|_\Sigma^2$, relative to the median of its head, against the decile of the angular distance $1-\cos(k_i, c_j)$, for the partition built by RetroInfer. Keys are sorted by their angular distance to their centroid and split into ten groups of equal size. The energy of the queries concentrates on the direction of the mean query, and keys at the same angular distance from their centroid have residual logits that differ by about an order of magnitude.

Table 7: Query geometry across models and regimes. Each entry is the median over prompts, layers and key-value heads, with the interquartile range in brackets. The spread is in orders of magnitude, and the isotropic share is $\frac{k_{90}}{d}$.

Model	Regime	$\frac{\lambda_1}{\operatorname{tr}\Sigma}$	k_{90}	Rank corr.	Spread	Residual share	Isotropic share
Qwen3-4B-Instruct-2507	ProLong	0.64 [0.51, 0.73]	30 [22, 39]	0.47 [0.44, 0.54]	0.74 [0.62, 0.86]	0.32 [0.22, 0.41]	0.23 [0.17, 0.30]
	LongBench	0.62 [0.53, 0.73]	32.5 [19, 39]	0.47 [0.42, 0.52]	0.82 [0.68, 0.91]	0.30 [0.19, 0.39]	0.25 [0.15, 0.30]
	RULER	0.63 [0.54, 0.75]	15 [10, 19]	0.39 [0.35, 0.46]	0.90 [0.78, 1.00]	0.14 [0.09, 0.19]	0.12 [0.08, 0.15]
Llama-3.1-8B-Instruct	ProLong	0.64 [0.59, 0.69]	25 [20, 31]	0.37 [0.33, 0.42]	0.89 [0.75, 1.03]	0.28 [0.20, 0.38]	0.20 [0.16, 0.24]
	LongBench	0.65 [0.60, 0.71]	22 [15, 28]	0.37 [0.34, 0.41]	0.92 [0.79, 1.07]	0.24 [0.16, 0.33]	0.17 [0.12, 0.22]
	RULER	0.67 [0.60, 0.73]	12 [9, 16]	0.34 [0.29, 0.40]	0.95 [0.83, 1.10]	0.12 [0.08, 0.18]	0.09 [0.07, 0.12]

Results. Figure 2 shows the spectrum and the residual logits on continued natural text, and Table 7 shows that the concentration of the spectrum is the same in all three regimes and for both models. The leading direction carries 62% to 67% of the trace, and its cosine with the mean query is at least 0.998 at the median in every row. The number k_{90} is smaller for RULER, whose answers are short and repetitive and therefore exercise fewer directions. The partition of RetroInfer shows the same pattern in every regime: the rank correlation stays between 0.34 and 0.47, the residual logits in one

angular decile spread over about an order of magnitude, and the residual share exceeds the isotropic share. Figures 3 and 4 repeat Figure 2 for the two question regimes. The concentration also varies across layers. For example, layer 20 of Qwen3-4B-Instruct-2507 has a leading share of 0.47 and $k_{90} = 41$ on ProLong, against 0.64 and 30 over all layers.

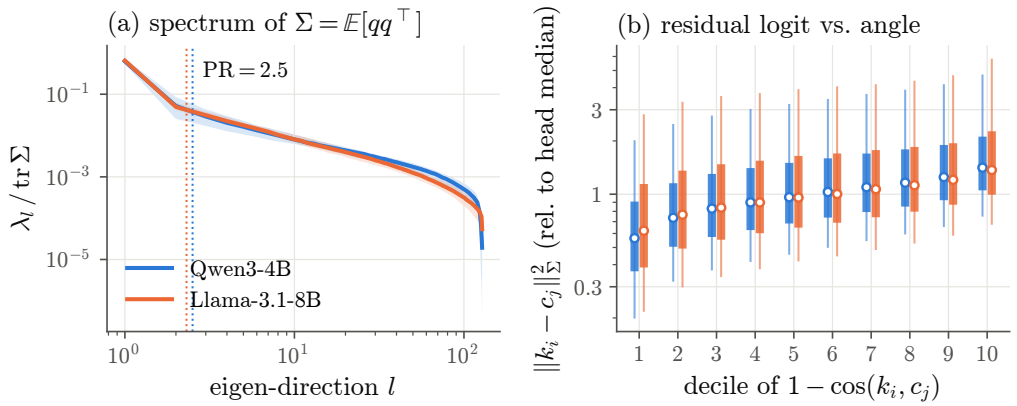

Figure 3: Query geometry on LongBench, in the layout of Figure 2. (a) Normalized spectrum of Σ, with the median over heads and the interquartile range. (b) Expected squared residual logit, relative to the median of its head, against the decile of angular distance, for the partition of RetroInfer. We observe the same concentration and the same weak dependence on the angle as for continued natural text.

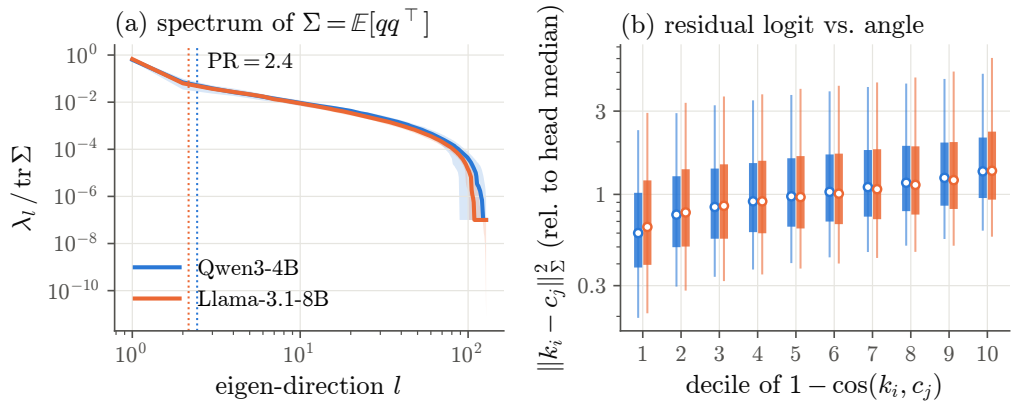

Figure 4: Query geometry on RULER, in the layout of Figure 2. (a) Normalized spectrum of Σ, with the median over heads and the interquartile range. (b) Expected squared residual logit, relative to the median of its head, against the decile of angular distance, for the partition of RetroInfer. The leading direction keeps the same share as for continued natural text, while the short answers of RULER exercise fewer directions in the tail.

E Experimental Configuration

This appendix lists the configuration of every model and benchmark in Section 5. All methods run in the same harness with the same forward pass, so the settings below apply to every method unless stated otherwise.

Exact regions and budget. Every method uses the layout of Section 3.2, with a sink of $n_{\text{sink}} = 32$ tokens and a flush every $n_{\text{flush}} = 256$ generated tokens, on every benchmark. The budget B counts only the retrieved tokens, and the steady tokens are attended exactly in addition to B. The recent window holds $n_{\text{win}} = 256$ tokens on the long-context benchmarks. The prompts of the reasoning benchmarks are only 58 to 215 tokens long, so such a window would cover the entire prompt and leave nothing to retrieve. We therefore set $n_{\text{win}} = 32$ on these benchmarks. Table 8 summarizes the protocol.

Table 8: Protocol per benchmark and model. The budget B counts retrieved tokens only, and the steady tokens are attended exactly in addition to B.

Benchmark	Model	Context	Budget B	n_{sink}	n_{win}	n_{flush}	Decoding	Max new tokens
RULER	Llama-3.1-8B-Instruct	16K, 32K, 64K, 128K	512, 1024, 2048, 4096 ($\frac{1}{32}$)	32	256	256	greedy	task default
	Qwen3-4B-Instruct-2507	16K, 32K, 64K, 128K	128, 256, 512, 1024 ($\frac{1}{128}$)	32	256	256	greedy	task default
LongBench	Llama-3.1-8B-Instruct	first 65,536 tokens	128, 256	32	256	256	greedy	task default
	Qwen3-4B-Instruct-2507	first 65,536 tokens	128, 256	32	256	256	greedy	task default
AIME-25	DeepSeek-R1-Distill-Llama-8B	prompt + generation	128, 256	32	32	256	model card	32,768
	Qwen3-4B-Instruct-2507	prompt + generation	128, 256	32	32	256	model card	32,768
GPQA-Diamond	DeepSeek-R1-Distill-Llama-8B	prompt + generation	128, 256	32	32	256	model card	32,768
	Qwen3-4B-Instruct-2507	prompt + generation	128, 256	32	32	256	model card	32,768

Hyperparameters of SIEVE. Table 9 lists the hyperparameters of SIEVE, which are shared by all models and benchmarks. The page size and the size of the estimation zone are those of RetroInfer, so that the two methods retrieve and estimate the same number of tokens and clusters.

Table 9: Hyperparameters of SIEVE, shared by all models and benchmarks.

Component	Hyperparameter	Value
Query moments	prefill queries per query head T_0	100
	decoding positions of the placement window W	128
Objective	ridge ρ	10
	weight of the target term γ	2
Clustering	segments per indexed region	16
	mean number of keys per cluster	16
	Lloyd iterations	10
	conjugate-gradient iterations per centroid update	6
Retrieval and estimate	page size	8 tokens
	share of clusters in the estimation zone	0.232

Benchmarks and metrics. Table 10 lists the number of samples and the reported metric of each benchmark. All scores are scaled by 100. On RULER, the budget is a fixed fraction of the context, so it grows with the context length. Qwen3-4B-Instruct-2507 receives $\frac{1}{128}$ instead of $\frac{1}{32}$ because every method matches full attention on it at $\frac{1}{32}$, which leaves no room to separate the methods. On LongBench and on the reasoning benchmarks, the budget is an absolute number of tokens.

Table 10: Samples and reported metric of each benchmark.

Benchmark	Tasks	Samples	Reported metric
RULER	13 subtasks	100 per subtask	accuracy, macro over 13 subtasks
LongBench	16 tasks	100 per task	official metric, macro over 14 tasks
AIME-25	30 problems	4 per problem	Mean@4 and Pass@4
GPQA-Diamond	198 questions	1 per question	accuracy

Long-context inputs. RULER prompts are generated with the official RULER scripts for each model tokenizer and context length. LongBench follows the KV Press protocol (Devoto et al., 2024): the context is truncated to its first $65{,}536$ tokens, the question is appended, and the model chat template is applied. The two tasks `lcc` and `multi_news` have median prompts of $2{,}542$ and $1{,}846$ tokens, so a budget of 128 or 256 tokens together with the exact regions covers a large part of each prompt. We therefore exclude them from the main average and report the average over all 16 tasks in Table 13.

Sampling on the reasoning benchmarks. The reasoning benchmarks follow the sampling parameters recommended by each model card, which we verified against the `generation_config.json` shipped with the weights. Qwen3-4B-Instruct-2507 uses temperature 0.7, top-p 0.8, and top-k 20. DeepSeek-R1-Distill-Llama-8B uses temperature 0.6 and top-p 0.95 without top-k, no system prompt, and a response that opens with the `<think>` tag, as prescribed by its chat template. Because the two models sample differently, their scores are not on a common scale, and we report each model in its own table.

F RULER Results per Subtask

Tables 11 and 12 report the accuracy of every method on each subtask of RULER. S1 to S3 denote the three single-needle subtasks, MK1 to MK3 the three multi-key subtasks, MV and MQ the multi-value and multi-query subtasks, VT variable tracking, CWE and FWE common-word and frequent-word extraction, and QA1 and QA2 the two question-answering subtasks.

Table 11: Accuracy (↑), scaled by 100, per RULER subtask on Llama-3.1-8B-Instruct, with budget $B = \frac{1}{32}$ of the context. Avg is the macro-average over the 13 subtasks. Bold indicates the best sparse method in each column of a context length, and is omitted when all sparse methods tie.

Context	Method	S1	S2	S3	MK1	MK2	MK3	MV	MQ	VT	CWE	FWE	QA1	QA2	Avg
16K	Full attention	100.0	100.0	100.0	100.0	100.0	100.0	99.8	100.0	100.0	70.5	89.3	85.0	65.0	93.0
	SIEVE	100.0	100.0	100.0	**100.0**	**100.0**	**100.0**	99.0	99.8	**99.6**	70.8	**89.3**	**85.0**	**65.0**	**93.0**
	Quest	100.0	100.0	100.0	**100.0**	**100.0**	99.0	99.8	**100.0**	98.0	33.9	79.0	84.0	64.0	89.1
	ShadowKV	100.0	100.0	100.0	**100.0**	**100.0**	**100.0**	99.3	99.8	96.8	51.7	79.0	**85.0**	64.0	90.4
	RetroInfer	100.0	100.0	100.0	99.0	99.0	98.0	99.0	**100.0**	**99.6**	60.8	86.3	**85.0**	**65.0**	91.7
	ParisKV	100.0	100.0	100.0	**100.0**	**100.0**	**100.0**	**100.0**	99.8	98.0	**71.0**	72.0	84.0	64.0	91.4
32K	Full attention	100.0	100.0	100.0	100.0	100.0	100.0	98.0	100.0	99.0	17.4	94.0	86.0	59.0	88.7
	SIEVE	100.0	100.0	100.0	100.0	100.0	**100.0**	97.0	**100.0**	**98.6**	23.2	**91.7**	86.0	59.0	**88.9**
	Quest	100.0	100.0	100.0	100.0	100.0	97.0	**99.0**	99.5	95.4	12.7	80.3	86.0	59.0	86.8
	ShadowKV	100.0	100.0	100.0	100.0	100.0	**100.0**	97.0	99.8	90.6	19.0	85.7	84.0	59.0	87.3
	RetroInfer	100.0	100.0	100.0	100.0	100.0	**100.0**	97.0	99.8	98.2	11.3	90.3	**87.0**	59.0	87.9
	ParisKV	100.0	100.0	100.0	100.0	100.0	97.0	97.0	99.8	96.4	**23.8**	84.3	85.0	59.0	87.9
64K	Full attention	100.0	100.0	100.0	100.0	99.0	98.0	97.5	99.0	97.4	1.1	85.7	83.0	60.0	86.2
	SIEVE	100.0	100.0	100.0	100.0	**99.0**	**96.0**	**97.0**	**99.3**	**96.2**	**4.1**	83.7	83.0	**59.0**	**85.9**
	Quest	100.0	100.0	100.0	100.0	98.0	84.0	94.5	99.0	87.6	1.5	**85.3**	83.0	58.0	83.9
	ShadowKV	100.0	100.0	100.0	100.0	98.0	**96.0**	**97.0**	99.0	82.4	1.8	71.0	83.0	58.0	83.6
	RetroInfer	100.0	100.0	100.0	100.0	98.0	93.0	94.5	99.0	**96.2**	2.2	79.0	83.0	58.0	84.8
	ParisKV	100.0	100.0	100.0	100.0	98.0	94.0	94.5	98.8	89.6	2.9	82.0	83.0	58.0	84.7
128K	Full attention	100.0	100.0	99.0	96.0	88.0	63.0	96.0	99.3	69.4	0.0	71.0	81.0	48.0	77.7
	SIEVE	100.0	**100.0**	99.0	96.0	87.0	**57.0**	**96.0**	**100.0**	**73.0**	0.0	**71.0**	**81.0**	**48.0**	**77.5**
	Quest	100.0	**100.0**	**100.0**	96.0	83.0	34.0	95.5	99.3	70.2	0.0	62.0	80.0	**48.0**	74.5
	ShadowKV	100.0	99.0	99.0	96.0	83.0	47.0	93.2	98.8	62.2	0.0	61.0	78.0	46.0	74.1
	RetroInfer	100.0	**100.0**	99.0	96.0	84.0	46.0	94.3	**100.0**	67.2	0.0	70.7	78.0	47.0	75.6
	ParisKV	100.0	**100.0**	98.0	96.0	**88.0**	40.0	95.5	99.3	62.6	0.0	66.0	79.0	**48.0**	74.8

Table 12: Accuracy (↑), scaled by 100, per RULER subtask on Qwen3-4B-Instruct-2507, with budget $B = \frac{1}{128}$ of the context. Avg is the macro-average over the 13 subtasks. Bold indicates the best sparse method in each column of a context length, and is omitted when all sparse methods tie.

Context	Method	S1	S2	S3	MK1	MK2	MK3	MV	MQ	VT	CWE	FWE	QA1	QA2	Avg
16K	Full attention	100.0	100.0	100.0	100.0	100.0	98.0	99.8	100.0	100.0	91.6	98.7	85.0	60.0	94.9
	SIEVE	100.0	100.0	**100.0**	100.0	**100.0**	98.0	**99.5**	100.0	**100.0**	**89.5**	**99.0**	**85.0**	**60.0**	**94.7**
	Quest	100.0	100.0	99.0	100.0	98.0	38.0	94.8	100.0	99.2	53.3	97.0	83.0	58.0	86.2
	ShadowKV	100.0	100.0	**100.0**	100.0	**100.0**	98.0	98.3	100.0	**100.0**	49.5	98.0	82.0	57.0	91.0
	RetroInfer	100.0	100.0	99.0	100.0	**100.0**	98.0	**99.5**	100.0	**100.0**	82.0	98.0	84.0	**60.0**	93.9
	ParisKV	100.0	100.0	**100.0**	100.0	**100.0**	**99.0**	**99.5**	100.0	**100.0**	81.8	95.0	84.0	59.0	93.7
32K	Full attention	100.0	100.0	100.0	100.0	100.0	100.0	99.8	100.0	100.0	84.8	96.0	83.0	62.0	94.3
	SIEVE	100.0	100.0	100.0	100.0	**100.0**	**100.0**	99.3	**100.0**	**100.0**	**83.6**	94.7	84.0	**63.0**	**94.2**
	Quest	100.0	100.0	100.0	100.0	99.0	58.0	95.8	**100.0**	**100.0**	49.3	91.0	79.0	61.0	87.2
	ShadowKV	100.0	100.0	100.0	100.0	**100.0**	99.0	99.5	99.8	99.6	55.4	94.7	80.0	62.0	91.5
	RetroInfer	100.0	100.0	100.0	100.0	**100.0**	**100.0**	**99.8**	**100.0**	**100.0**	71.9	**95.3**	**85.0**	60.0	93.2
	ParisKV	100.0	100.0	100.0	100.0	**100.0**	**100.0**	**99.8**	**100.0**	**100.0**	79.7	**95.3**	84.0	61.0	93.8
64K	Full attention	100.0	100.0	100.0	99.0	100.0	100.0	100.0	100.0	99.0	51.5	92.7	72.0	59.0	90.2
	SIEVE	100.0	100.0	100.0	**99.0**	**100.0**	**100.0**	**100.0**	99.5	98.8	47.0	**92.3**	**76.0**	59.0	**90.1**
	Quest	100.0	100.0	100.0	**99.0**	98.0	36.0	96.5	99.5	**99.0**	23.3	77.3	73.0	**60.0**	81.7
	ShadowKV	100.0	100.0	100.0	98.0	**100.0**	79.0	99.8	99.8	96.8	38.4	77.0	74.0	57.0	86.1
	RetroInfer	100.0	100.0	100.0	**99.0**	**100.0**	**100.0**	99.8	99.5	98.8	30.6	90.7	73.0	59.0	88.5
	ParisKV	100.0	100.0	100.0	**99.0**	**100.0**	98.0	99.8	**100.0**	98.8	**57.9**	83.0	70.0	57.0	89.5
128K	Full attention	100.0	100.0	100.0	99.0	91.0	88.0	99.8	97.5	91.8	14.8	83.7	58.0	45.0	82.2
	SIEVE	100.0	100.0	100.0	**99.0**	92.0	**90.0**	**99.8**	97.0	92.2	14.1	**86.7**	**60.0**	**48.0**	**83.0**
	Quest	100.0	100.0	100.0	98.0	83.0	12.0	97.8	97.5	87.6	10.5	76.3	56.0	43.0	74.0
	ShadowKV	100.0	100.0	100.0	**99.0**	**93.0**	32.0	99.5	**97.8**	86.0	15.8	74.0	58.0	**48.0**	77.2
	RetroInfer	100.0	100.0	100.0	98.0	91.0	87.0	**99.8**	96.5	**93.6**	7.3	83.0	59.0	47.0	81.7
	ParisKV	100.0	100.0	100.0	97.0	91.0	59.0	**99.8**	**97.8**	88.4	**21.6**	77.0	57.0	47.0	79.7

Saturated subtasks. On eight subtasks, namely S1 to S3, MK1, MV, MQ, QA1, and QA2, the sparse methods differ by at most 6 points in every setting. The six retrieval subtasks among them are nearly saturated, with every method at 93 or above. The two question-answering subtasks are limited by the model rather than by the index, since full attention itself scores between 45 and 86 on them and the sparse methods follow it closely.

Subtasks that separate the methods. The remaining five subtasks carry the differences in the macro-average. On MK3, Quest fails on Qwen3-4B-Instruct-2507 at every context length, with scores between 12 and 58, and ShadowKV and ParisKV degrade sharply at 128K on both models. SIEVE stays within 6 points of full attention on MK3 in every setting and exceeds it at 128K on Qwen3-4B-Instruct-2507. MK2 separates the methods only at 128K on Qwen3-4B-Instruct-2507, by up to 10 points. On CWE, full attention itself fails beyond 32K on Llama-3.1-8B-Instruct, with 1.1 at 64K and 0.0 at 128K, so this subtask measures the index only at the shorter contexts.

Comparison with RetroInfer and full attention. Across the 104 combinations of subtask and setting, SIEVE scores higher than RetroInfer in 41, lower in 6, and ties in the remaining 57. The largest advantage of RetroInfer is 1.4 points, on VT at 128K on Qwen3-4B-Instruct-2507. Against full attention, the largest loss of SIEVE is 6.0 points, on MK3 at 128K on Llama-3.1-8B-Instruct, followed by 4.5 points on CWE at 64K on Qwen3-4B-Instruct-2507, and every other loss is at most 2.3 points. SIEVE also exceeds full attention in 16 combinations. With 100 samples per subtask, a difference of a few points on a single subtask is within sampling noise, so we attribute these gains to noise rather than to a systematic effect.

G LongBench Results per Task

Table 13 reports the score of every method on each task of LongBench at budget $B = 256$, together with the macro-averages over 14 and 16 tasks. The 14-task average excludes `lcc` and `multi_news`, for the reason given in Appendix E.

Table 13: Score (↑), scaled by 100, per LongBench task at budget $B = 256$. Avg14 excludes `lcc` and `multi_news`, and Avg16 averages all tasks. Bold indicates the best sparse method in each column of a model, and is omitted when all sparse methods tie.

Model	Method	NQA	Qasper	MFQA	HQA	2WQA	MSQ	GovR	QMSum	MNews	TREC	TQA	SAMSum	PCount	PRet	LCC	RB-P	Avg14	Avg16
Llama-3.1-8B-Instruct	Full attention	31.4	46.6	55.2	57.9	52.1	34.0	35.3	25.4	27.3	20.9	89.7	40.2	9.1	100.0	53.4	42.1	45.7	45.0
	SIEVE	**31.6**	**47.2**	**56.6**	57.9	51.0	34.6	**34.9**	**25.6**	**27.2**	**21.2**	89.4	38.7	**9.1**	100.0	**54.3**	43.3	**45.8**	**45.2**
	Quest	31.1	45.5	55.6	**58.3**	51.1	33.3	33.4	25.2	26.9	20.3	89.4	37.9	9.0	100.0	51.1	**44.8**	45.4	44.6
	ShadowKV	31.5	44.5	55.6	56.4	**52.0**	**34.9**	34.3	24.9	**27.2**	20.6	**89.7**	39.0	**9.1**	100.0	53.4	44.3	45.5	44.8
	RetroInfer	31.5	45.4	56.1	57.9	51.1	34.0	34.7	25.2	**27.2**	20.3	88.9	37.9	**9.1**	100.0	53.4	41.9	45.3	44.7
	ParisKV	30.4	45.5	56.0	56.4	51.0	34.0	34.7	25.2	**27.2**	19.6	89.4	**40.2**	**9.1**	100.0	53.4	41.9	45.2	44.6
Qwen3-4B-Instruct-2507	Full attention	30.9	41.9	46.6	51.4	35.7	15.3	30.9	22.5	24.1	72.0	83.4	42.5	2.9	92.1	63.2	53.4	44.4	44.3
	SIEVE	**30.7**	**42.4**	45.7	**53.0**	36.5	**16.4**	30.3	**22.8**	23.7	71.0	83.9	41.5	2.2	92.4	**63.6**	52.5	**44.4**	**44.3**
	Quest	29.7	41.7	45.6	52.8	**36.6**	14.2	29.1	22.5	23.7	**72.0**	**84.2**	40.1	3.0	91.0	61.9	53.5	44.0	43.9
	ShadowKV	30.0	41.5	46.5	52.8	36.5	15.0	**31.8**	**22.8**	**24.3**	**72.0**	84.1	**42.1**	**3.5**	**94.0**	62.3	46.1	44.2	44.1
	RetroInfer	30.0	41.7	45.6	51.4	36.1	**16.4**	30.3	22.6	24.1	**72.0**	83.5	41.6	2.2	92.7	62.3	51.2	44.1	44.0
	ParisKV	30.0	41.5	**46.7**	52.1	36.5	15.4	31.0	22.5	24.2	71.0	83.5	41.8	2.4	91.5	61.5	**53.7**	44.3	44.1

Per-task differences. The sparse methods differ by at most 3.2 points on every task, except `repobench-p` on Qwen3-4B-Instruct-2507, where ShadowKV drops to 46.1 against 53.4 for full attention. SIEVE differs from full attention by at most 1.6 points on every task, in either direction. With 100 samples per task, differences of this size on a single task are within sampling noise, which is consistent with the parity observed on the averages. The 16-task average leads to the same conclusion as the 14-task average: SIEVE matches or exceeds full attention and is the best sparse method on both models.

H Ablation Study

H.1 Ablation setting

Variants. The five variants of Table 14 differ only in how the partition is built. The first is the spherical k-means of RetroInfer on the centered keys. The second replaces the angular distance by the Euclidean distance between the centered keys, with the mean as centroid. The third replaces the Euclidean distance by $\|k_i - c_j\|_\Sigma$, that is, the metric of Section 4.2 with $\rho = 0$, and the fourth uses Σ_ρ with the value of ρ in Table 9. The first four variants use the mean of the keys as the centroid. The fifth adds the target term of Section 4.3 with the centroid update equation 8, which gives SIEVE. All variants use 16 segments, 16 keys per cluster on average, and the retrieval and estimate of Section 3.3.

Index quality. The *mass recall* is the share of the attention mass of the indexed region that falls on the retrieved tokens,

$$\frac{\sum_{i\in\mathcal{S}(q)} e^{\langle q,k_i\rangle}}{\sum_{i\in\mathcal{X}} e^{\langle q,k_i\rangle}},$$

and the *estimation error* is the relative error of the contribution of the estimation zone to the numerator of equation 2,

$$\frac{\left\|\sum_{j\in\mathcal{E}(q)}(\hat{N}_j - N_j)\right\|_2}{\left\|\sum_{j\in\mathcal{E}(q)} N_j\right\|_2}.$$

We average both quantities over the decoding queries of every prompt, layer, and key-value head, and then over the 13 subtasks for Table 14. Tables 15, 16 and 17 report every subtask.

H.2 Ablation results

We run the study on Qwen3-4B-Instruct-2507 at 32K with $B = 128$, that is, $\frac{1}{256}$ of the context, where the budget is small enough for each component to matter. Table 14 reports the accuracy together with the mass recall and the estimation error defined above.

Results. Table 14 shows that every component contributes. The Euclidean distance raises the accuracy on aggregation from 78.8 to 82.4, which isolates the effect of the norm of the keys, but lowers it on `niah_multikey_3` from 100 to 96. The metric Σ raises the mass recall from 46.4 to 51.1, lowers the estimation error from 63.2 to 54.3, and raises the accuracy on aggregation to 86.3. Without ridge, however, it loses accuracy on retrieval, down to 52 on `niah_multikey_3` and 95.3 on

Table 14: Ablation of the components of SIEVE, in the setting of Section 5.3. MV is `niah_multivalue`, and the other accuracy columns are as in Table 2. The two index-quality columns average the 13 subtasks. Bold indicates the best variant in each column.

	Accuracy (↑)				Index quality	
Variant	Macro	MK3	MV	AGG	Mass recall (↑)	Estimation error (↓)
Angular distance	92.2	**100.0**	97.8	78.8	45.5	64.5
+ Euclidean distance	92.4	96.0	98.3	82.4	46.4	63.2
+ query-induced metric Σ	88.9	52.0	95.3	**86.3**	**51.1**	54.3
+ robust metric Σ_ρ	92.8	97.0	98.3	83.8	49.6	59.8
+ target term, **SIEVE**	**93.5**	**100.0**	**99.8**	85.3	50.7	**43.1**

`niah_multivalue`, and brings the macro-average below that of angular clustering. The robust metric Σ_ρ recovers most of this loss, with 97 on `niah_multikey_3`, at the cost of part of the gains in index quality. This is the trade-off of the ridge: it keeps penalized the directions where the estimate has little energy, and spends some resolution on them. The target term then brings both retrieval subtasks to their best values, 100 and 99.8, lowers the estimation error from 59.8 to 43.1, the largest reduction in the table, and gives the best macro-average, 93.5. It thus reduces the dilution of Section 4.1 both in the ranking and in the estimate, since it moves each centroid toward the keys that carry most of the mass of its cluster.

Table 15: Accuracy (↑), scaled by 100, per RULER subtask for the variants of Table 14, on Qwen3-4B-Instruct-2507 at 32K with $B = 128$. Bold indicates the best variant in each column, and is omitted when all variants tie.

Variant	S1	S2	S3	MK1	MK2	MK3	MV	MQ	VT	CWE	FWE	QA1	QA2	Macro
Angular distance	100.0	**100.0**	**100.0**	100.0	**100.0**	**100.0**	97.8	**100.0**	**100.0**	63.0	94.7	**82.0**	61.0	92.2
+ Euclidean distance	100.0	**100.0**	**100.0**	100.0	**100.0**	96.0	98.3	**100.0**	**100.0**	69.1	95.7	**82.0**	60.0	92.4
+ query-induced metric Σ	100.0	99.0	99.0	100.0	99.0	52.0	95.3	99.3	**100.0**	**77.2**	95.3	79.0	60.0	88.9
+ robust metric Σ_ρ	100.0	**100.0**	**100.0**	100.0	**100.0**	97.0	98.3	**100.0**	99.6	72.2	95.3	**82.0**	62.0	92.8
+ target term, **SIEVE**	100.0	**100.0**	**100.0**	100.0	**100.0**	**100.0**	**99.8**	**100.0**	**100.0**	73.2	**97.3**	**82.0**	**63.0**	**93.5**

Table 16: Mass recall (↑), scaled by 100, per RULER subtask for the variants of Table 14. Bold indicates the best variant in each column.

Variant	S1	S2	S3	MK1	MK2	MK3	MV	MQ	VT	CWE	FWE	QA1	QA2	Macro
Angular distance	51.0	49.9	49.9	49.0	40.7	39.9	48.5	48.4	50.0	38.7	37.2	42.6	45.4	45.5
+ Euclidean distance	53.1	49.4	49.2	49.1	41.5	41.4	48.5	48.6	54.0	40.7	38.9	42.6	45.7	46.4
+ query-induced metric Σ	**57.9**	53.5	55.3	**55.2**	**44.1**	**47.0**	53.0	**54.4**	**57.7**	**46.0**	**41.7**	**48.0**	**50.1**	**51.1**
+ robust metric Σ_ρ	55.7	52.6	53.7	53.2	43.2	44.8	52.3	53.0	57.1	44.6	40.7	45.7	48.5	49.6
+ target term, **SIEVE**	56.7	**54.3**	**55.4**	54.8	44.0	45.9	**54.2**	53.7	57.5	45.0	40.8	47.4	49.8	50.7

Table 17: Estimation error (↓), scaled by 100, per RULER subtask for the variants of Table 14. Bold indicates the best variant in each column.

Variant	S1	S2	S3	MK1	MK2	MK3	MV	MQ	VT	CWE	FWE	QA1	QA2	Macro
Angular distance	49.2	69.5	69.1	70.4	60.7	61.3	69.7	70.0	62.6	62.0	56.2	69.4	68.9	64.5
+ Euclidean distance	47.3	68.7	68.6	69.4	59.5	59.7	68.9	68.9	59.6	60.3	54.6	68.6	68.0	63.2
+ query-induced metric Σ	38.8	59.5	61.4	60.0	50.2	51.3	61.1	58.0	52.4	50.1	44.8	59.9	58.5	54.3
+ robust metric Σ_ρ	43.3	65.5	65.2	65.9	56.4	56.6	65.6	64.8	55.7	56.4	50.8	65.7	65.1	59.8
+ target term, **SIEVE**	**29.2**	**46.9**	**47.8**	**47.6**	**41.1**	**40.6**	**46.6**	**47.4**	**38.6**	**43.3**	**38.1**	**46.8**	**46.0**	**43.1**

Results per subtask. Without ridge, the metric Σ loses accuracy on most retrieval subtasks, not only on `niah_multikey_3`, whereas its gains in index quality are spread over all subtasks. The

reduction of the estimation error by the target term is also spread over all subtasks, from 59.8 to 43.1 on average.

I On the Sparsity of Attention

Section 3.2 relies on three properties of attention at decoding time: a small subset of tokens holds most of the attention mass, the attention sink and the recent window hold a large and stable share of it, and the dominant tokens change from one decoding step to the next. This appendix measures the three properties on the models of Section 5.

Setup. We use Qwen3-4B-Instruct-2507 and Llama-3.1-8B-Instruct on four regimes with contexts of 32K tokens: continued natural text from the ProLong pretraining mixture (Gao et al., 2025), the questions of LongBench (Bai et al., 2024), and the retrieval and aggregation subtasks of RULER (Hsieh et al., 2024). The decoding queries are generated greedily for up to 256 tokens. For every decoding query and every query head, we compute the exact attention weights over the whole cache. We call the first $n_{\text{sink}} = 32$ tokens the sink, the last $n_{\text{win}} = 256$ tokens the recent window, and the remaining tokens the indexed region $\mathcal{X}$, as in Section 5. No generated token leaves the recent window within 256 steps, so $\mathcal{P}$ stays empty.

Quantities. We measure three quantities per decoding query and head. The first is the share of the attention mass on the sink, the recent window, and the indexed region. The second is the concentration of the mass within the indexed region: the smallest fraction ν_{90} of the tokens of $\mathcal{X}$ that holds 90% of the mass of $\mathcal{X}$, and the share of the mass of $\mathcal{X}$ held by its top tokens, for the budget fractions $\frac{1}{32}$, $\frac{1}{128}$ and $\frac{1}{256}$ of the context. The third is the stability of the dominant tokens: the fraction of the top-B tokens of $\mathcal{X}$ at one decoding step that remain in the top-B tokens Δ steps later.

Table 18: Sparsity of attention at decoding time, for contexts of 32K tokens, in percent. The three masses are shares of the total attention mass, averaged over decoding queries, layers, and query heads, so that they sum to 100. The last two columns give the median over decoding queries, layers, and query heads, with the interquartile range in brackets, of ν_{90}, the smallest fraction of the indexed region that holds 90% of its mass, and of the share of the mass of the indexed region held by its top $\frac{1}{128}$ of the context.

Model	Regime	Sink mass	Recent mass	Indexed mass	ν_{90}	Top-$\frac{1}{128}$ share
Qwen3-4B-Instruct-2507	ProLong	22.2	34.9	42.9	2.69 [0.30, 8.01]	79.2 [56.8, 93.7]
	LongBench	29.9	42.2	28.0	2.27 [0.09, 8.54]	82.4 [56.1, 96.0]
	RULER, retrieval	29.5	38.7	31.8	2.74 [0.05, 10.70]	81.9 [50.9, 96.2]
	RULER, aggregation	29.7	42.3	28.0	4.35 [0.26, 12.50]	74.0 [47.7, 93.9]
Llama-3.1-8B-Instruct	ProLong	53.5	24.9	21.6	8.92 [4.78, 14.30]	52.0 [41.2, 65.6]
	LongBench	53.0	33.7	13.3	9.84 [5.33, 15.90]	52.9 [40.8, 67.2]
	RULER, retrieval	53.8	28.0	18.2	11.20 [5.47, 18.50]	53.8 [38.7, 70.8]
	RULER, aggregation	49.2	29.1	21.7	10.10 [2.48, 20.10]	61.2 [38.4, 84.8]

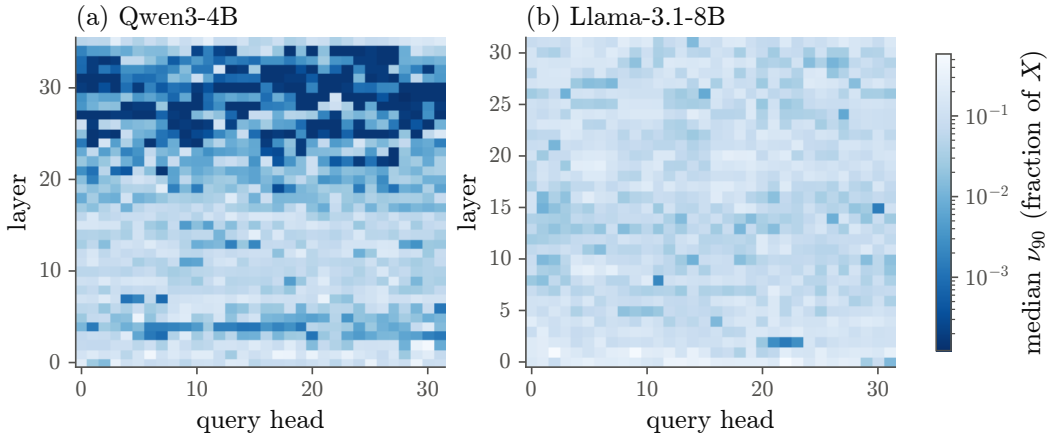

Figure 5: Concentration ν_{90} of the attention mass within the indexed region, for every layer and query head, on continued natural text at 32K tokens. (a) Qwen3-4B-Instruct-2507. (b) Llama-3.1-8B-Instruct. Each cell is the median over decoding queries, and darker cells indicate sparser heads.

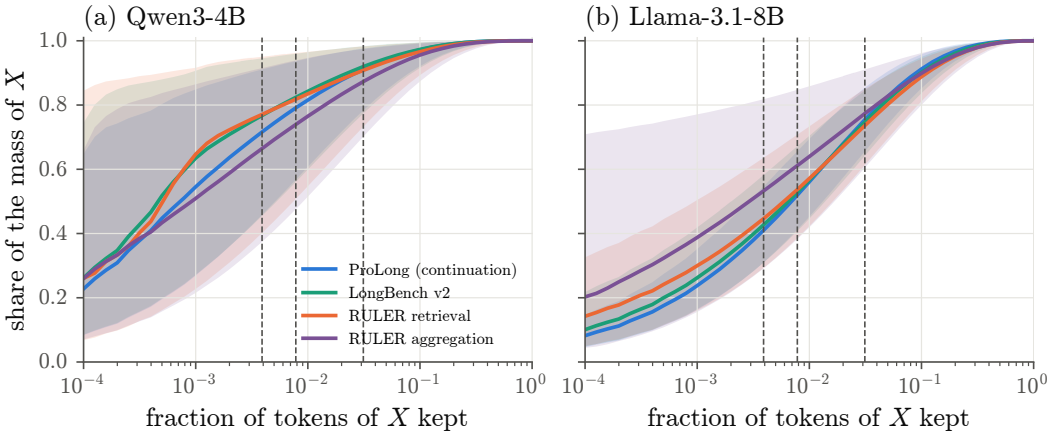

Figure 6: Share of the attention mass of the indexed region held by its top tokens, against the fraction of tokens kept, for the four regimes. Each curve is the median over decoding queries, layers, and query heads, with the interquartile range shaded. The vertical lines mark the budget fractions $\frac{1}{32}$, $\frac{1}{128}$ and $\frac{1}{256}$ of the context.

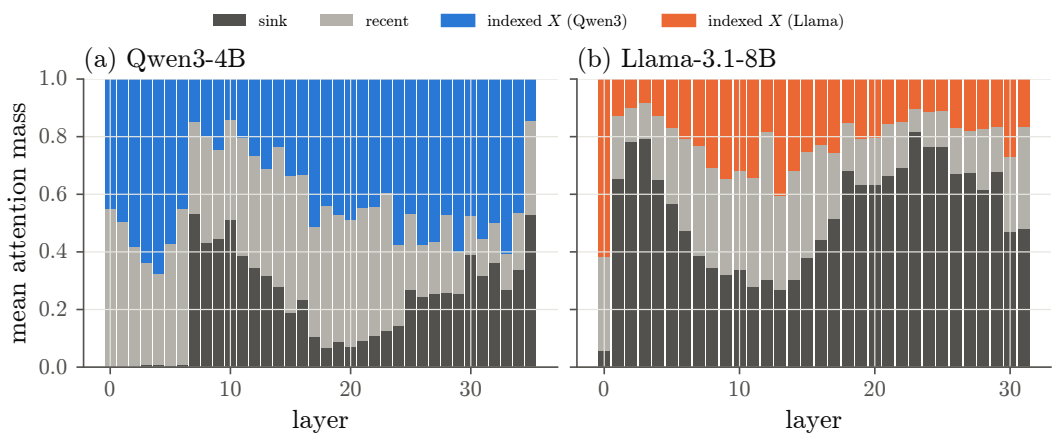

Figure 7: Share of the attention mass on the sink, the recent window, and the indexed region, for every layer, on continued natural text at 32K tokens. Each bar is the average over decoding queries and query heads, so that the three shares sum to one.

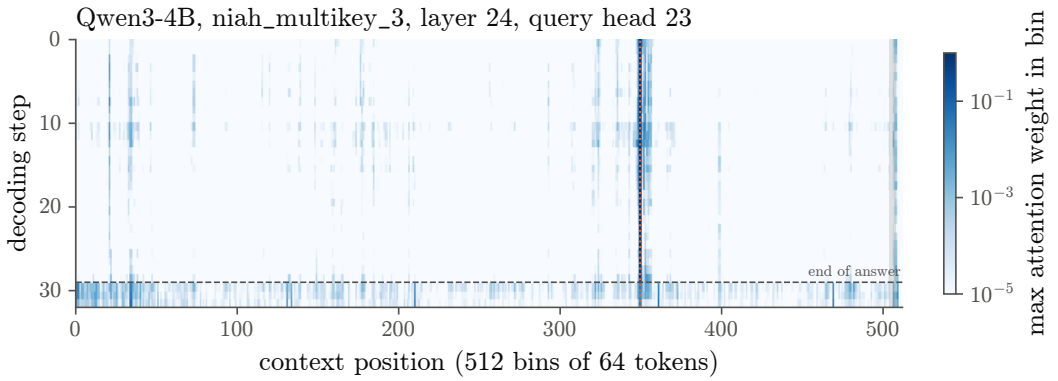

Figure 8: Attention weights of one head over the positions of the context, for the first 32 decoding steps of a `niah_multikey_3` prompt of 32K tokens. The head is query head 23 of layer 24 of Qwen3-4B-Instruct-2507, which places the largest mass on the needle among all heads. The shaded bands mark the sink and the recent window.

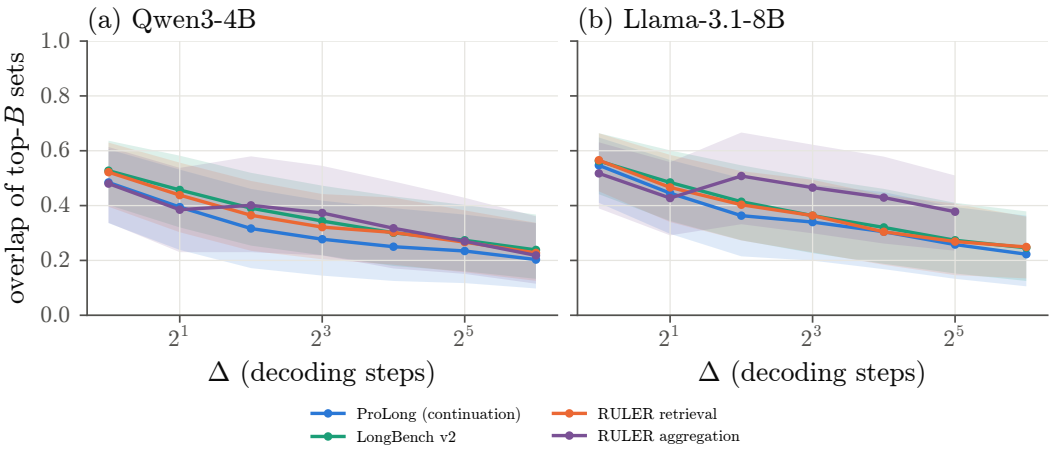

Figure 9: Fraction of the top-B tokens of the indexed region at one decoding step that remain in the top-B tokens Δ steps later, for the budget $B = \frac{1}{128}$ of the context. Each curve is the median over decoding queries, layers, and query heads for one regime. The curves of RULER stop at $\Delta = 16$, since its answers are too short to measure longer gaps.

Results. The steady tokens hold most of the attention mass. On average over decoding queries, layers, and heads, the sink and the recent window together hold 57% to 72% of the mass on Qwen3-4B-Instruct-2507 and 78% to 87% on Llama-3.1-8B-Instruct, across the four regimes of Table 18. The sink alone dominates many heads of Llama-3.1-8B-Instruct: its median mass exceeds one half in 49% to 59% of the heads, against 12% to 24% for Qwen3-4B-Instruct-2507. Figure 7 shows how these shares vary across layers.

Within the indexed region, the mass concentrates on few tokens. The median of ν_{90} is 2.3% to 4.4% on Qwen3-4B-Instruct-2507 and 8.9% to 11.2% on Llama-3.1-8B-Instruct, and the top $\frac{1}{128}$ of the context holds 74% to 82% and 52% to 61% of the mass of the region, respectively. Qwen3-4B-Instruct-2507 is therefore markedly sparser, which is consistent with the smaller budget that it receives in Section 5.1. The concentration also varies across layers and heads, as shown in Figure 5. In Qwen3-4B-Instruct-2507, it increases sharply with depth: the median of ν_{90} falls from about 10% in the first layers to below 0.3% between layers 25 and 33. In Llama-3.1-8B-Instruct, it stays between 5% and 12% after the first two layers. On Qwen3-4B-Instruct-2507, aggregation spreads the mass over more tokens than retrieval, with a median of ν_{90} of 4.4% against 2.7%, whereas the two regimes are close on Llama-3.1-8B-Instruct.

The dominant tokens change during decoding. In the head of Figure 8, the mass on the needle stays between 74% and 100% during the first 29 decoding steps and falls below 3% at the next three steps. Across heads, the top-B tokens change quickly, as shown in Figure 9. After one decoding step, the median overlap is 0.48 to 0.57 in every regime, so about half of the top-B tokens are replaced at each step. After 16 steps, it falls to 0.25 to 0.32, with the exception of aggregation on Llama-3.1-8B-Instruct at 0.43. After 64 steps, it reaches 0.20 to 0.25 on ProLong and LongBench, whose generations are long enough to measure this gap. The pattern is the same on both models, so the retrieved tokens must be selected anew at every step, as assumed in Section 3.2.

J Efficiency

Setup. We measure the prefill time, the median latency of one decoding step, and the peak GPU memory at 128K tokens, with the budgets of Section 5.1, that is, $B = 1024$ on Qwen3-4B-Instruct-2507 and $B = 4096$ on Llama-3.1-8B-Instruct. Table 19 reports each quantity relative to full attention, averaged over the two models.

Results. SIEVE decodes 1.44 times faster than full attention on average and lowers the peak memory by 33.3%, at the cost of a 6.6% longer prefill for the construction of the index. These numbers are

Table 19: Efficiency relative to full attention at 128K tokens, averaged over the two models. Bold indicates the best sparse method in each column.

Method	Prefill time (↓)	Decoding speedup (↑)	Peak memory (↓)
SIEVE	$+6.6\%$	$1.44\times$	-33.3%
Quest	$+\mathbf{2.2}\%$	$\mathbf{1.53}\times$	-31.4%
ShadowKV	$+26.8\%$	$1.39\times$	$-\mathbf{36.7}\%$
RetroInfer	$+5.7\%$	$1.45\times$	-33.7%
ParisKV	$+3.8\%$	$1.19\times$	-25.6%

close to those of RetroInfer, which shares the retrieval and the estimate of SIEVE. The two methods differ by $0.01\times$ in speedup and by 0.4 points in peak memory, and the partition of SIEVE adds about one percent of the prefill time. The speedup of SIEVE is larger on Qwen3-4B-Instruct-2507 than on Llama-3.1-8B-Instruct, $1.67\times$ against $1.21\times$, since the budget is four times larger relative to the context on Llama-3.1-8B-Instruct. Among the baselines, Quest decodes fastest, since it needs no estimation zone, but it trails SIEVE by up to 9.0 points on RULER. ShadowKV has the lowest peak memory thanks to its low-rank key cache, but it spends 26.8% more time than full attention to build this cache at prefill.