CALIBKV: CALIBRATED BUDGETS FOR LONG-CONTEXT ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse attention must decide which cached keys to use and how many to retain. Fixed budgets cannot adapt to each request, while sound score bounds in our measurements admit much of the cache. We propose CALIBKV, which estimates a per-head retention budget from the request's keys and selects keys by exact score at decode. A learned query subspace and a separate calibration bank set the budget during prefill; the current query determines the selected keys. We establish pairwise residual coverage under exchangeability and evaluate quality and cost at the resulting budgets. On book-passage language modeling tasks using three *War and Peace* segments, mean $\Delta\mathrm{NLL}$ relative to Dense ranges from -0.018 to $+0.010$ over 8K to 128K contexts, while the mean budget falls from 31.2% to 2.1%. LongBench macro scores are within 0.12 points of Dense on Llama and Qwen. On MATH-500, CALIBKV chooses a 99.87% budget and scores 46.6%, compared with 44.6% for Dense. At long contexts, its single-layer attention calculation is faster than Dense.

1 INTRODUCTION

Generating a token from a long context requires attention scores for every key in the KV cache. Page scores (Tang et al., 2024), hashing (Chen et al., 2025), product quantization (Zhang et al., 2025a), and approximate nearest-neighbor (ANN) search (Liu et al., 2025) reduce this cost by selecting part of the cache. Missing a high-scoring key can remove a substantial share of attention mass. Louver uses a sound gate to retain every key above a supplied score threshold (Dehghankar & Asudeh, 2026). The number of keys admitted by such a gate still depends on the threshold and the tightness of its bound.

We first examine how many keys sound screening admits in practice. On the Llama-3.1-8B request in Figure 1, Cauchy-Schwarz projection bounds (Teflioudi et al., 2015; Li et al., 2017) admit mean fractions of 25.49% at 16K and 11.99% at 64K. Quest's page bounds (Tang et al., 2024) and Louver's paper ball-bound procedure admit more than 95% at both lengths (Section 3). At the same sampled positions, CALIBKV selects 19.50% and 3.66%, including always-attend keys. These measurements motivate estimating a retention budget from the request's score distribution.

This gap points to two distinct decisions in sparse attention. To choose keys, the decode query can rank the current cache by exact score. To choose a count, the method must account for how the request's keys distribute scores across heads. The relevant score range changes across heads and prompts, so one count need not serve each head equally well. A fixed top-k rule makes that decision before observing the request, while a screening bound can admit many more keys than attention needs. Such a bound describes candidates at a threshold, but it does not prescribe how many keys attention should retain. We set the count during prefill and let each decode query choose positions within it.

CALIBKV estimates a budget from the observed score distribution for each query head h. It fits a rank-m query subspace offline and holds out a bank of decode queries for calibration. At prefill, it scores this bank against the request keys and converts the average admitted fraction into a count B_h. At decode, exact scores select the top B_h keys. The approximation therefore determines how many keys to retain, and the current query determines which keys are retained. Four calibration choices shape the resulting budgets: keeping the subspace-fit and calibration sets disjoint (L1), matching calibration queries to served traffic (L2) and its position range (L3), and handling sinks and norm outliers separately (L4). Section 4.3 measures the effect of each choice.

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

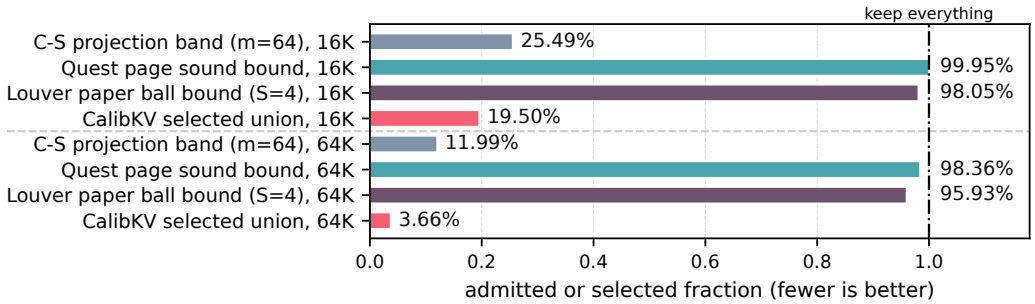

Figure 1: **Sound-bound candidates and calibrated selection.** Mean fractions on Llama-3.1-8B. The first three methods count screening candidates; exact-score CALIBKV counts its selected union, including always-attend keys.

We compare CALIBKV with Dense, exact top-k Oracle, and recent sparse attention methods (Tang et al., 2024; Singhania et al., 2024; Dehghankar & Asudeh, 2026). Among the evaluated sparse paths, CALIBKV gives the lowest mean *War and Peace* ΔNLL in seven of eight model/context blocks. Its mean Llama budget falls from 31.2% to 2.1% as context grows from 8K to 128K. At long contexts, its single-layer attention calculation is faster than Dense.

Contributions. First, we measure three families of sound attention-score bounds and show that their admitted sets are often large at decode-query thresholds. Second, we introduce a budget estimator that combines an offline query model with the current request's keys, followed by exact selection at decode. Third, we evaluate quality, budgets, and computational cost through book-passage language modeling, downstream tasks, full-model GPU timing, and parameter sensitivity and calibration controls. CALIBKV maintains near-dense quality while adapting attention budgets to each request. On book-passage language modeling and LongBench, it achieves this quality using substantially fewer keys than Dense.

2 RELATED WORK

KV-cache methods either evict entries across decode steps or select entries for each query. Eviction policies use sinks, recency, or heavy-hitter statistics (Xiao et al., 2024b; Zhang et al., 2023; 2025b). Per-query methods retrieve entries using page scores, hashing, quantization, ANN indexes, or low-rank landmarks (Tang et al., 2024; Chen et al., 2025; Zhang et al., 2025a; Liu et al., 2025; Sun et al., 2025). Loki (Singhania et al., 2024) and SparQ (Ribar et al., 2024) rank entries with reduced-dimensional scores under a fixed budget. CALIBKV uses a query subspace and residual calibration to estimate the budget, then selects entries using exact scores.

Louver's ball-halfspace gate (Dehghankar & Asudeh, 2026), Cauchy-Schwarz projection bands (Teflioudi et al., 2015; Li et al., 2017), and Quest page upper bounds (Tang et al., 2024) return all keys above a supplied score threshold. Section 3 measures their admitted fractions. The pairwise residual result uses the rank argument from split conformal prediction (Vovk et al., 2005; Lei et al., 2018). Appendix A compares the methods in more detail.

3 MEASURED SOUND BOUNDS ADMIT LARGE FRACTIONS

Setting. Fix a layer and head with cache length n, cached keys $\kappa_1, \ldots, \kappa_n \in \mathbb{R}^d$, decode query q, and exact scores $s_i = q^\top \kappa_i$. Here k is the reference top-k count, and τ_k^* is the k-th largest score. A selector's *admitted set* contains the keys it returns at a supplied threshold. For an approximation $\hat{s}_i$, the residual $r_i = s_i - \hat{s}_i$ is the score component not captured by the approximation. A sound upper bound permits rejecting a key only when that bound is below the threshold.

Three families of sound bounds. **Cauchy-Schwarz (C-S) projection bounds** (Teflioudi et al., 2015; Li et al., 2017) admit keys satisfying $\hat{s}(q, \kappa) + \delta(q)R(\kappa) \geq \tau$, where $\delta(q)$ and $R(\kappa)$ are the

norms of the query and key components orthogonal to the fitted subspace (Theorem 1). **Quest page bounds** (Tang et al., 2024) use per-channel minima and maxima to bound scores within each 16-key page. Threshold screening admits a page if its bound reaches τ. Each diagnostic obtains $\tau \leq \tau_k^*$ from its own exact-scored seed set. **Louver's ball bounds** (Dehghankar & Asudeh, 2026) bound each subspace cluster by its query-center score plus its radius times the query norm. We implement the paper's threshold scan over these bounds and use its released SampleMax reservoir estimator for τ. Each filter retains all keys scoring at least its own threshold.

Figure 1 measures the candidate sets returned by these sound-screening rules. The CALIBKV bars show exact-score selection rather than sound-bound screening. Each method uses its own threshold or budget.

What the bounds admit. The three bounds use the same dense decode captures from a *Moby-Dick* request at each length. Quest and Louver admit most of the cache at both lengths. C-S admits less, while exact-score CALIBKV selects a smaller union that includes always-attend keys. All bars use the same layer, head, and decode-position grid. Appendix D.2 gives the protocol and the separate historical fixed-threshold sweep. At each context length, the bound results average 8,192 observations over sampled layers, heads, and decode positions. All three sound filters retain every key above their own estimated threshold in those captures. The plotted quantity is the size of the admitted set before any exact-score post-filtering; it is not a fixed-budget top-k result.

Slack and score margin. A sound filter accounts for the largest score consistent with its stored information. For a projection bound at threshold τ, a key is excluded only when the margin $\tau - \hat{s}(q, \kappa)$ exceeds $\delta(q)R(\kappa)$. Large admitted fractions mean that few keys satisfy this exclusion test. A key with a low exact score can still pass when its bound is loose, so a threshold near the top-k score need not produce a small candidate set. CALIBKV replaces the worst-case residual allowance with a measured one to estimate a retention count. Appendix B analyzes the residual band.

4 CALIBKV: PER-REQUEST KV-CACHE BUDGETING

CALIBKV separates budget estimation from key selection. Offline, it learns a per-head query subspace and stores a separate bank of calibration queries. For each new request, prefill compares exact and low-dimensional scores between that bank and the request keys. The measured errors supply a residual allowance, which turns the approximate scores into candidate counts. Averaging these counts gives a budget B_h for each head. At each decode step, the current query scores all cached keys exactly, selects the top B_h, and computes attention over those entries together with an always-attend set. Calibration is done once per request, while the retained keys change with each decode query. This is the *exact-score* path used by unqualified references to CALIBKV throughout the paper. The *rank-m* alternative screens keys before exact scoring during decode (Section 4.2). Figure 2 shows both paths.

Notation. Consider one layer and query head h. The cache contains n keys $K = \{\kappa_1, \ldots, \kappa_n\}$ and corresponding values, with $\kappa_i \in \mathbb{R}^d$. For a decode query q, the exact score is $s_i = q^\top \kappa_i$. We omit the common attention scale from selection formulas because it does not change the ranking. The symbol k is a reference top-k count, whereas B_h is the request's estimated retention count. The offline subspace has m orthonormal columns, $U_h \in \mathbb{R}^{d \times m}$, and Q_{cal} is the separate calibration-query bank. Defaults are $m = 64$, $d = 128$, and $k = 128$.

4.1 OFFLINE, PREFILL, AND DECODE

(1) Offline: fit the query subspace and retain a calibration bank. This stage takes captured decode queries from example requests and returns reusable per-head state. We split the queries into a PCA-fit set and a disjoint bank Q_{cal}. The leading m PCA directions form U_h, while the bank is held out for the request-level calculation below. Each query head has its own subspace, even when several query heads share a KV head: pooling the four query heads in a GQA group increases residuals and admitted fractions.

The measured rank-64 projections contain 70-85% of squared query norm. Separate rank-8/16 controls at 16K compare subspaces fitted on another request with those fitted on the evaluated request:

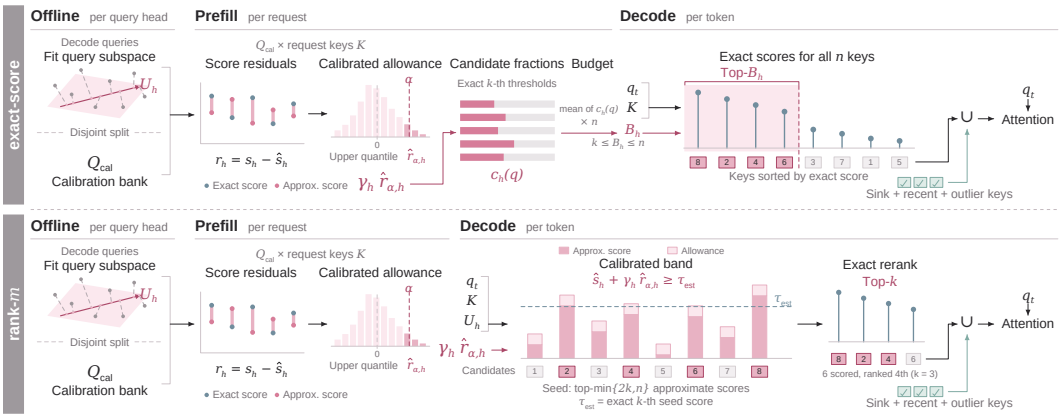

Figure 2: **CALIBKV workflow.** Top: the exact-score path uses an offline query subspace and calibration bank to set a per-request budget B_h during prefill; each decode query scores all keys exactly and attends to the top B_h plus the always-attend set. Bottom: the rank-m path uses approximate scores to screen keys, then exactly reranks the candidates and attends to the top k plus the always-attend set.

the ratio of their mean projection-residual norms is 0.90 on Qwen and 0.96-0.98 on Llama. These measurements show how a subspace fitted on one request transfers to another within the same model. Appendix D.3 reports the corresponding rank and calibration controls.

The offline stage also estimates a per-head scale factor γ_h for content shift. For each request-key fold, it compares the upper residual quantile from other-request queries with that from the fold's own queries, then takes the largest cross/self ratio, clipped below at one. Prefill multiplies the request's residual quantile by this stored factor; Table 4 reports median-ratio variants.

(2) Prefill: turn calibration scores into a request budget. This stage combines the offline state with the new request's keys and returns one count B_h per head. Let $\bar{\kappa}_h$ be the mean key. The projected score and its residual are

$$\hat{s}_h(q,\kappa) = q^\top \bar{\kappa}_h + (U_h^\top q)^\top U_h^\top (\kappa - \bar{\kappa}_h), \qquad r_h(q,\kappa) = q^\top \kappa - \hat{s}_h(q,\kappa).$$

The first term retains the mean-key contribution exactly; the second uses only m projected coordinates. The residual measures the score component missed by this projection.

Prefill computes exact and approximate scores between Q_{cal} and the request keys, forming two $|Q_{\text{cal}}| \times n$ matrices. Their difference supplies the upper empirical residual quantile $\hat{r}_{\alpha,h}$. Adding the scaled allowance gives the band

$$C_h(q,\tau) = \{\kappa \in K : \hat{s}_h(q,\kappa) + \gamma_h \hat{r}_{\alpha,h} \ge \tau\}.$$

For each bank query, the exact scores provide the k-th largest score $\tau_k^*(q;K)$. We count the keys admitted at that threshold and average the fractions:

$$c_h(q) = \frac{|C_h(q, \tau_k^*(q;K))|}{n}, \qquad \tilde{b}_h = \frac{1}{|Q_{\text{cal}}|} \sum_{q \in Q_{\text{cal}}} c_h(q).$$

The request budget is

$$B_h = \min\{n, \max\{k, \lfloor n\tilde{b}_h \rfloor + 1\}\}, \qquad b_h = B_h/n.$$

Thus, k sets the reference threshold for counting candidates, whereas B_h sets the count used at decode. The subspace is reused, but the residual quantile and budget are recomputed for each request.

The request's keys make the residual estimate reflect the cache being served. Table 5 illustrates the importance of calibration source: combining self- and cross-request inputs lowers the measured violation rate from 1.5×10^{-2} for cross-request inputs alone to 3.0×10^{-4} at 64K on Llama-3.1-8B ($m = 16$, $\alpha = 0.9999$). Section 4.3 separates the effects of query source and position range.

Norm-outlier keys are excluded from residual calibration and always attended. This always-attend set also includes the first four sink tokens (Xiao et al., 2024b) and the 64 most recent tokens. If the mean calibrated-band fraction $\tilde{b}_h$ exceeds 0.35, the head uses Dense attention and its reported budget is n.

(3) Decode: select by exact scores at the estimated budget. Given the current query, cache, and prefill budget, this stage computes $q^\top \kappa_i$ for all n keys and selects the exact top B_h. It combines those positions with the always-attend set. After the usual attention scaling, softmax and value aggregation operate on this union. Thus, selection reads every key, but attention aggregates values only at retained positions. New keys enter the recent window immediately. Calibration determines how many keys to keep; the current query determines which keys. Since $B_h \geq k$, the selected set contains the exact top-k keys. Section 5.1 compares these budgets with fixed top-k selection.

Worked 64K exact-score request. Consider the 64K Llama-3.1-8B run with $k = 128$, $m = 64$, and $|Q_{\text{cal}}| = 256$. Offline captures produce one subspace U_h and a separate 256-query calibration bank for each head. The prompt then produces 65,536 keys per head. At prefill, every bank query is scored against those keys both exactly and in the rank-64 subspace. The score differences set the residual quantile, while the exact scores set that query's 128th-score threshold. We count how many approximate scores, after adding the residual allowance, cross this threshold and average the 256 counts to obtain B_h. After dense-head routing, the mean B_h is 2,053 keys, or 3.1% of the cache. At each decode step, exact scores select the top B_h keys, which are combined with the sink, recent, and norm-outlier keys before attention.

4.2 Rank-m Screening Alternative

This alternative screens keys using approximate scores. It projects the current query and ranks the cache in the learned subspace. To estimate the unknown exact top-k threshold, it takes the $\min\{2k, n\}$ largest approximate scores, computes the seed scores exactly, and uses their k-th largest value as τ_{est}. For $n \geq k$, Lemma 1 gives $\tau_{\text{est}} \leq \tau_k^*$, so the band at the estimated threshold contains the band at the true threshold. This two-stage rule reduces the admitted fraction from 65% for the earlier calibrated-threshold variant to 14% in the reported control.

The path then forms $C_h(q, \tau_{\text{est}})$, computes full-dimensional scores for those candidates, and retains their top k together with the always-attend set. New keys receive their projections when they leave the recent window. Exact reranking resolves the order within the candidate set but cannot recover excluded keys. Full-cache scores therefore measure the *top-k miss rate* during evaluation, defined as the fraction of true top-k keys excluded by screening. These diagnostic scores are separate from those required to execute the screened path. Thus, the two paths use the calibrated band differently: rank-m screens keys before exact reranking, whereas exact-score CalibKV uses it only to estimate a count.

4.3 Calibration Requirements

Four calibration choices determine how the offline query model transfers to a request. Their measurements appear in Appendix C.

L1: keep calibration separate from subspace fitting. We use disjoint queries for PCA fitting and residual calibration. Reusing the fit queries understates the residual allowance; a 64K rank-m control compares this reuse with an even/odd split using quality and full-score diagnostics. In that control, reuse admits about 0.1% of keys but misses 92% of the true top-k set; the disjoint split admits 13-14% and lowers the miss rate to 12%.

L2: match calibration queries to decode traffic. The bank should cover the type of decode queries expected at inference. A traffic-matched bank gives a 16.4% candidate fraction at 16K; prefill-query and narrow-content banks alter both the fraction and the miss rate. With the same decode-fitted subspace, a prefill-query bank instead admits 95-97% of the cache. The bank therefore needs to represent the queries used for selection, not merely the text in the prompt.

L3: cover the served position range. Offline captures should span the positions used by the request. Position-matched captures improve the quality-budget tradeoff at 64K; Table 4 gives the strengthened configuration. In a separate 32K rank-m control, replacing the position-matched subspace with

a 16K-fit subspace raises the candidate fraction from 3.2% to 12.7% and ΔNLL from $+0.139$ to $+0.261$.

L4: exclude always-attended outliers from calibration. We attend sink and norm-outlier keys separately and calibrate the remaining keys. This keeps their large residuals out of the quantile used to count candidates. The paired control in Appendix C separates this effect from traffic matching.

4.4 Coverage Analysis

For exchangeable query-key pairs $Z_1, \ldots, Z_N, Z_*$ independent of the fitted quantities in $\hat{s}$, let $\hat{r}_\alpha$ be the $\lceil (N+1)\alpha \rceil$-th calibration-residual order statistic (or $+\infty$ when the index is $N+1$). For the band $C(q,\tau) = \{\kappa : \hat{s}(q,\kappa) + \hat{r}_\alpha \geq \tau\}$, Theorem 2 shows that for any threshold τ,

$$\Pr\big[\kappa_* \notin C(q_*,\tau) \ \wedge \ q_*^\top \kappa_* \geq \tau\big] \leq 1-\alpha.$$

For a random query and length-n cache, suppose a uniformly sampled key forms a query-key pair satisfying the theorem's exchangeability assumptions with the calibration pairs. If exactly k cache positions are relevant, expected recall is at least $\max\{0, 1 - n(1-\alpha)/k\}$. Appendix B gives the proof and the separate rank-m threshold lemma.

5 Experiments

Models and hardware. We evaluate Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct (Yang et al., 2024), Qwen3-8B (Yang et al., 2025), DeepSeek-R1-Distill-Llama-8B (DeepSeek-AI, 2025), and Llama-3.1-70B-Instruct (Appendix D.5). Quality runs use bf16, batch size 1, and NVIDIA A100-80GB GPUs.

Baselines. All methods use the same evaluation harness. **Dense** attends to the full cache. The **Oracle** selects the exact top 128 keys. **Quest** ranks 16-key pages to fill a 128-key budget (Tang et al., 2024). **Loki** ranks keys in per-head PCA coordinates at rank 32 and retains 25%, with no sink or recent window (Singhania et al., 2024). **Louver** uses its released gate and threshold estimator (Dehghankar & Asudeh, 2026). We report both exact-score and rank-m **CalibKV**. Exact-score CalibKV estimates B_h per request and attends the exact top B_h. Rank-m determines its candidate count from the calibrated band, then attends the exact top k within those candidates. Louver likewise obtains its candidate count from its request-level gate. Appendix E records the reproduction details.

Tasks, metrics, and setup. We measure mean teacher-forced token negative log-likelihood (NLL) on public-domain book passages. We also evaluate needle retrieval, LongBench (Bai et al., 2024), RULER (Hsieh et al., 2024), and MATH-500 (Lightman et al., 2024). Offline full-score evaluation supplies rank-m diagnostic miss rates. Moby-Dick uses one segment; War and Peace reports mean±standard deviation over three disjoint segments. In Table 1, Needle is the retrieved fraction; pink rows identify CalibKV, and bold marks the lowest ΔNLL in each model/context block. Defaults are $k = 128$, $m = 64$, $|Q_{\text{cal}}| = 256$, sink/recent windows 4/64, and $\alpha = 0.999$. For exact-score CalibKV, the budget column averages $b_h = B_h/n$, charging Dense-routed heads as 1. Before accounting for overlap with the selected keys, the separate always-attend union increases this cache fraction by at most 0.31 percentage points at 32K, 0.20 at 64K, and 0.15 at 128K.

5.1 Quality at Estimated Budgets

Table 1 compares quality and selection fractions at the settings above. Louver and rank-m CalibKV report candidate fractions before exact top-k reranking; the other methods report keys retained for attention.

NLL at the estimated budgets. Among the compared sparse paths, CalibKV has the lowest mean War and Peace ΔNLL in seven of eight model/context blocks, and the lowest Moby-Dick ΔNLL in six. The result is strongest at long Llama contexts: at 128K it uses a 2.1% budget while achieving mean ΔNLL -0.018, compared with Loki's $+0.128$ at 25%. CalibKV also matches Dense on every listed needle setting.

Table 1: **Main quality results.** ΔNLL is relative to Dense (lower is better). MD is *Moby-Dick*; WP is *War and Peace*, reported as mean±SD over three segments. Qwen2.5 stops at its 32K trained context length.

Model	Method	Ctx	Budget	ΔNLL (MD)	ΔNLL ±SD (WP)	Needle (Dense)
Llama-3.1-8B-Instruct	Oracle	8K	1.5%	+0.039	+0.027 ± 0.030	1.00 (1.00)
	Quest	8K	1.5%	+0.112	−0.001 ± 0.014	1.00 (1.00)
	Loki	8K	25.0%	+0.150	**−0.011 ± 0.027**	1.00 (1.00)
	Louver	8K	63.6%	+0.020	+0.019 ± 0.030	1.00 (1.00)
	CALIBKV	8K	31.2%	**+0.002**	+0.004 ± 0.016	1.00 (1.00)
	CALIBKV (rank-m)	8K	34.4%	+0.119	+0.033 ± 0.020	1.00 (1.00)
	Oracle	16K	0.8%	+0.043	+0.015 ± 0.039	1.00 (1.00)
	Quest	16K	0.8%	+0.138	+0.048 ± 0.072	1.00 (1.00)
	Loki	16K	25.0%	+0.057	+0.014 ± 0.030	1.00 (1.00)
	Louver	16K	36.7%	+0.030	+0.023 ± 0.045	1.00 (1.00)
	CALIBKV	16K	18.0%	**−0.003**	**−0.001 ± 0.008**	1.00 (1.00)
	CALIBKV (rank-m)	16K	16.4%	+0.091	+0.040 ± 0.063	1.00 (1.00)
	Oracle	32K	0.4%	+0.087	+0.035 ± 0.040	1.00 (1.00)
	Quest	32K	0.4%	+0.051	+0.089 ± 0.066	1.00 (1.00)
	Loki	32K	25.0%	**+0.000**	+0.019 ± 0.025	1.00 (1.00)
	Louver	32K	31.8%	+0.033	+0.044 ± 0.027	1.00 (1.00)
	CALIBKV	32K	5.7%	+0.026	**+0.004 ± 0.009**	1.00 (1.00)
	CALIBKV (rank-m)	32K	3.2%	+0.139	+0.081 ± 0.067	1.00 (1.00)
	Oracle	64K	0.2%	+0.083	+0.033 ± 0.056	0.50 (0.50)
	Quest	64K	0.2%	+0.140	+0.112 ± 0.049	0.50 (0.50)
	Loki	64K	25.0%	+0.139	+0.017 ± 0.011	0.50 (0.50)
	Louver	64K	52.5%	+0.130	+0.031 ± 0.047	0.50 (0.50)
	CALIBKV	64K	3.1%	**+0.010**	**−0.003 ± 0.016**	0.50 (0.50)
	CALIBKV (rank-m)	64K	2.1%	+0.111	+0.038 ± 0.059	0.50 (0.50)
	Oracle	128K	0.1%	+0.202	+0.082 ± 0.040	0.50 (0.50)
	Quest	128K	0.1%	+0.834	+0.571 ± 0.127	0.50 (0.50)
	Loki	128K	25.0%	+0.169	+0.128 ± 0.053	0.50 (0.50)
	Louver	128K	57.8%	+0.245	+0.079 ± 0.038	0.50 (0.50)
	CALIBKV	128K	2.1%	**+0.098**	**−0.018 ± 0.010**	0.50 (0.50)
	CALIBKV (rank-m)	128K	1.7%	+0.120	+0.079 ± 0.028	0.50 (0.50)
Qwen2.5-7B-Instruct	Oracle	8K	1.6%	+0.075	+0.045 ± 0.034	1.00 (1.00)
	Quest	8K	1.5%	+0.179	+0.091 ± 0.033	1.00 (1.00)
	Loki	8K	25.0%	+0.226	+0.199 ± 0.188	1.00 (1.00)
	Louver	8K	60.2%	+0.080	+0.045 ± 0.022	0.92 (1.00)
	CALIBKV	8K	13.5%	**+0.008**	**+0.010 ± 0.010**	1.00 (1.00)
	CALIBKV (rank-m)	8K	10.6%	+0.093	+0.085 ± 0.029	1.00 (1.00)
	Oracle	16K	0.8%	+0.046	+0.004 ± 0.021	1.00 (1.00)
	Quest	16K	0.8%	+0.041	+0.094 ± 0.026	1.00 (1.00)
	Loki	16K	25.0%	+0.599	+0.224 ± 0.087	1.00 (1.00)
	Louver	16K	33.7%	+0.045	+0.020 ± 0.015	1.00 (1.00)
	CALIBKV	16K	7.4%	+0.024	**−0.009 ± 0.005**	1.00 (1.00)
	CALIBKV (rank-m)	16K	5.7%	**+0.013**	+0.020 ± 0.021	1.00 (1.00)
	Oracle	32K	0.4%	+0.092	+0.039 ± 0.012	1.00 (1.00)
	Quest	32K	0.4%	+0.051	+0.107 ± 0.042	1.00 (1.00)
	Loki	32K	25.0%	+0.538	+0.276 ± 0.135	1.00 (1.00)
	Louver	32K	33.7%	+0.133	+0.052 ± 0.020	1.00 (1.00)
	CALIBKV	32K	5.4%	**+0.023**	**+0.006 ± 0.010**	1.00 (1.00)
	CALIBKV (rank-m)	32K	4.7%	+0.118	+0.047 ± 0.014	1.00 (1.00)

The Qwen rows show the same quality-budget pattern at its evaluated lengths. At 32K, exact-score CALIBKV uses 5.4% of the cache and has mean *War and Peace* ΔNLL +0.006, while Louver admits 33.7% at +0.052 and Loki retains 25% at +0.276. The comparison uses each method's reported operating point rather than assigning them a common key count.

Exact selection at estimated budgets. The exact-score path improves mean *War and Peace* ΔNLL over rank-m screening in all eight model/context blocks. On Llama, rank-m gives +0.040, +0.038, and +0.079 at 16K, 64K, and 128K, while exact-score selection gives −0.001, −0.003, and −0.018 at the budgets in Table 1. Exact scoring therefore improves quality in the measured settings.

Budget versus context length. The retained count grows much less than cache length, so the retained fraction decreases at longer contexts. On Llama, the mean budget falls from 31.2% at 8K to 2.1% at 128K. On Qwen, it falls from 13.5% at 8K to 5.4% at 32K. In absolute terms, the Llama means range from 1.9K to 2.9K selected keys. At 128K, the three-segment ΔNLL means are $+0.082$ for the exact top-128 Oracle, $+0.571$ for Quest, and -0.018 for CALIBKV at its estimated 2.1% budget.

5.2 DOWNSTREAM TASKS

Table 2 reports retrieval, LongBench, RULER, and MATH-500 results. Among sparse methods, CALIBKV has the best average rank and attains or ties the best score in seven of eight rows. It has the highest Qwen2.5-7B LongBench macro score and ties Loki at 91 EM on 16K Llama RULER variable tracking. Its LongBench macro is 0.12 points below Dense on Llama and 0.02 points below on Qwen. Loki is 0.06 and 2.16 points below Dense, respectively; Quest is 1.36 and 3.28 points below. Louver is 0.06 points above Dense on Llama and 0.03 points below on Qwen. CALIBKV uses 9 to 57% of the cache on these LongBench tasks and 16% on the reported RULER cell. The per-task LongBench results show where the Qwen macro comes from. On narrativeqa, CALIBKV scores 27.14 versus Dense's 26.58 while using 9% of the cache; on hotpotqa it scores 53.33 versus 53.38 at 11%. The code-completion task lcc uses 29% and scores 61.18 versus Dense's 61.78. Appendix D.4 reports all ten tasks per model.

On MATH-500, CALIBKV adaptively expands its budget to 99.87% of the cache, rather than enforcing a small fixed count. At this near-Dense operating point, CALIBKV scores 46.6%, compared with 44.6% for Dense. Across the generation limits in Appendix Table 6, Oracle and Quest score 6 to 20 percentage points below Dense. On the full 500-example run with a 2048-token limit, Loki scores 39.2% at a 25% budget, and Louver scores 41.0% at a 92% budget.

Table 2: **Downstream tasks.** Scores are followed by mean selection fractions (Dense $= 1$; Louver counts gate candidates). Bold marks the best sparse-method score. Average rank covers the eight rows and sparse methods only; ties use mean ranks. Appendix D.4 gives per-task results.

Task (model)	Dense	Oracle	Quest	Loki	Louver	CALIBKV
needle (Qwen2.5-7B, 16K)	100	**100** (.008)	**100** (.008)	**100** (.25)	**100** (.32)	**100** (.07)
needle (Qwen3-8B, 16K)	100	**100** (.008)	75 (.008)	**100** (.25)	75 (.32)	**100** (.03)
multikey NIAH (Llama, 16-32K)	100	**100** (.004-.008)	**100** (.004-.008)	**100** (.25)	**100** (.30-.33)	**100** (.12-.16)
multi-needle 8-vault (Llama, 32K)	100	**100** (.004)	**100** (.004)	**100** (.25)	**100** (.31)	**100** (.05)
LongBench macro (Llama, 10 tasks)	42.20	42.01 (.01-.06)	40.84 (.01-.06)	42.14 (.25)	**42.26** (.32-.78)	42.08 (.17-.57)
LongBench macro (Qwen2.5-7B, 10 tasks)	41.89	41.64 (.01-.06)	38.61 (.01-.05)	39.73 (.25)	41.86 (.33-.73)	**41.87** (.09-.29)
RULER vt EM (Llama, 16K; 100 examples)	94	83 (.008)	26 (.008)	**91** (.25)	70 (.35)	**91** (.16)
MATH-500 (R1-Distill-8B; 2048-token cap)	44.6	38.6 (.09)	37.2 (.09)	39.2 (.25)	41.0 (.92)	**46.6** (.99)
Average rank ↓		3.1	4.2	2.7	2.8	**2.2**

5.3 PARAMETER SENSITIVITY

Section 5.1 compares exact-score and rank-m selection as a selection-path ablation. Here we vary subspace rank, calibration-bank size, and residual-quantile level to measure their effects on candidate fraction and screening misses. Appendix C gives the query-source, position-range, and outlier controls.

Subspace rank has a much larger effect on candidate fraction than calibration-bank size in the measured grid. Figure 3 examines subspace rank, calibration-bank size, and residual-quantile level on Llama-3.1-8B. In the 16K in-sample rank sweep, increasing m from 8 to 64 reduces the candidate fraction from 77% to 4.1%; $m = d = 128$ reproduces exact scores to numerical precision. Expanding the bank from 64 to 1024 queries changes the mean fraction by at most two percentage points. Increasing m sharpens the approximate ranking, so fewer keys enter the calibrated band.

Increasing α raises the residual allowance: the band admits more candidates and misses fewer true top-k keys. The figure therefore separates representation quality from the conservativeness of the allowance. The disjoint-split controls in Section 4.3 separate the effect of rank from calibration leakage. The in-sample sweep diagnoses the screen; Table 1 reports the evaluated disjoint-split configurations.

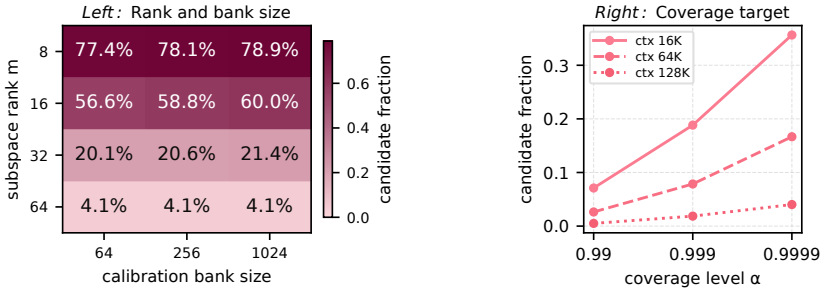

Figure 3: **Calibration parameter sensitivity.** Left: candidate fraction by subspace rank and calibration-bank size. Right: increasing the empirical quantile level α raises the candidate fraction and reduces measured rank-m misses; at fixed α, the fraction decreases with context length.

5.4 Computational Cost

Table 3 reports full-model GPU request time and attention calculation time. The full-model runs use Llama-3.1-8B with native grouped-query attention (GQA), batch size 1, and 256 generated tokens. At each length, three fixed prompts each receive one full warmup request followed by three timed repetitions. Prefill includes work specific to the current request; reusable fitting is measured once in the offline column and excluded from end-to-end time. For CalibKV, offline fitting and calibration-bank preparation are reusable. Prefill scores the bank against request keys to set B_h; decode scores and selects keys before attention. The attention column is part of decode time, not another end-to-end stage.

Method	Offline (s)	Prefill (s)	Decode (ms/token)	Attention (ms/step)	End-to-end (s)
Dense	–	0.73/4.08/36.18	17.53/23.87/48.05	0.10/0.34/1.31	5.20/10.16/48.43
Quest	–	0.73/4.09/36.25	25.59/26.74/33.21	0.41/0.41/1.06	7.26/10.91/44.72
Loki	3.42	0.74/4.08/36.15	41.83/65.31/134.61	0.31/0.82/2.53	11.40/20.73/70.48
Louver	–	1.05/5.04/40.15	24.25/24.73/43.92	0.14/0.19/0.87	7.24/11.35/51.35
CalibKV	84.95	1.35/4.90/37.99	29.49/31.58/53.05	0.29/0.33/0.98	8.87/12.95/51.52

Table 3: **GPU timing.** Triples give mean times at 8K/32K/128K, from nine requests per length for full-model timing and repeated single-layer measurements for attention.

At long contexts, CalibKV reduces single-layer attention time to 0.33 versus 0.34 ms per step at 32K and 0.98 versus 1.31 ms at 128K. End-to-end time is 51.52 versus 48.43 s, a 6.4% increase, due to request-specific prefill calibration and decode-time exact selection. Over 256 generated tokens, the prefill and decode costs outweigh the attention-component saving.

6 Conclusion and Limitations

In conclusion, we studied how long-context sparse attention can set a retention count for each request and head. Sound score bounds often admit large candidate sets at the measured thresholds, while CalibKV uses the learned subspace to estimate a count and exact decode scores to select keys. On Llama *War and Peace* segments, the mean budget falls from 31.2% to 2.1% as context grows from 8K to 128K, with mean ΔNLL between -0.018 and $+0.010$. LongBench scores remain close to Dense on Llama and Qwen. The separate attention measurement shows a benefit from selecting fewer keys, while full-model timing identifies prefill and selection as the remaining systems costs.

The analysis and timing results have distinct limits. The pairwise coverage result in Theorem 2 assumes exchangeable query-key pairs and does not certify a request-level budget or attention-output error for the crossed calibration matrix used here. The timing results cover batch-1 requests on one A100. Exact-score selection and request-specific prefill currently outweigh the single-layer attention savings in full-model time. Fusing selection with attention and reducing prefill calibration cost are direct engineering directions.

AI USE STATEMENT

We used ChatGPT and Claude Code to polish the manuscript and to help develop and examine parts of the theoretical analysis. They also assisted with implementation and code debugging. The authors reviewed the AI-generated content and take responsibility for the paper.

ETHICS STATEMENT

The experiments use publicly released models, public benchmarks, and public-domain texts, with no human subjects, personal data, or private datasets. Because sparse attention can change model outputs, quality results and full-score diagnostics are reported alongside budget fractions. Deployment carries the same misuse and output-quality risks as the underlying models.

REPRODUCIBILITY STATEMENT

Section 5 describes the evaluated models, tasks, and metrics. Appendix E gives the experimental settings, baseline reproduction details, and measurement protocol. The assumptions and derivations for the theoretical results appear in Appendix B. We will release the code upon acceptance.

REFERENCES

Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

Yushi Bai, Xin Lv, Jiajie Zhang, et al. LongBench: A bilingual, multitask benchmark for long context understanding. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, et al. MagicPIG: LSH sampling for efficient LLM generation. In *International Conference on Learning Representations (ICLR)*, 2025.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Mohsen Dehghankar and Abolfazl Asudeh. Sparse attention as a range searching problem: Towards an inference-efficient index for KV cache. *arXiv preprint arXiv:2605.06763*, 2026.

Aditya Desai, Kumar Agrawal, Shuo Yang, Alejandro Cuadron Lafuente, Luis Gaspar Schroeder, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. vAttention: Verified sparse attention via sampling. In *International Conference on Learning Representations (ICLR)*, 2026.

Paolo Ferragina and Giorgio Vinciguerra. The PGM-index: a fully-dynamic compressed learned index with provable worst-case bounds. *Proceedings of the VLDB Endowment*, 13(8):1162-1175, 2020.

Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Junhui He, Junna Xing, Nan Wang, Rui Xu, Shangyu Wu, Peng Zhou, Qiang Liu, Chun Jason Xue, and Qingan Li. A^2ATS: Retrieval-based KV cache reduction via windowed rotary position embedding and query-aware vector quantization. *arXiv preprint arXiv:2502.12665*, 2025.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, et al. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling (COLM)*, 2024.

Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *ACM SIGMOD International Conference on Management of Data*, 2018.

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523): 1094-1111, 2018.

Hui Li, Tsz Nam Chan, Man Lung Yiu, and Nikos Mamoulis. FEXIPRO: Fast and exact inner product retrieval in recommender systems. In *ACM SIGMOD International Conference on Management of Data*, 2017.

Wenhao Li, Yuxin Zhang, Gen Luo, Haiyuan Wan, Ziyang Gong, Fei Chao, and Rongrong Ji. Spotlight attention: Towards efficient LLM generation via non-linear hashing-based KV cache retrieval. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, et al. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

Di Liu, Meng Chen, Baotong Lu, et al. RetrievalAttention: Accelerating long-context LLM inference via vector retrieval. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Pierre-Emmanuel Mazaré, Gergely Szilvasy, Maria Lomeli, Francisco Massa, Naila Murray, Hervé Jégou, and Matthijs Douze. Inference-time sparse attention with asymmetric indexing. *arXiv preprint arXiv:2502.08246*, 2025.

Vikram Nathan, Jialin Ding, Mohammad Alizadeh, and Tim Kraska. Learning multi-dimensional indexes. In *ACM SIGMOD International Conference on Management of Data*, 2020.

Xuanfan Ni, Liyan Xu, Chenyang Lyu, Longyue Wang, Mo Yu, Lemao Liu, Fandong Meng, Jie Zhou, and Piji Li. ReFreeKV: Towards threshold-free KV cache compression. In *Findings of the Association for Computational Linguistics (ACL Findings)*, 2026.

Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive confidence machines for regression. In *European Conference on Machine Learning (ECML)*, 2002.

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. In *International Conference on Machine Learning (ICML)*, 2026.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. SparQ attention: Bandwidth-efficient LLM inference. In *International Conference on Machine Learning (ICML)*, 2024.

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank keys for efficient sparse attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, et al. ShadowKV: KV cache in shadows for high-throughput long-context LLM inference. In *International Conference on Machine Learning (ICML)*, 2025.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context LLM inference. In *International Conference on Machine Learning (ICML)*, 2024.

Christina Teflioudi, Rainer Gemulla, and Olga Mykytiuk. LEMP: Fast retrieval of large entries in a matrix product. In *ACM SIGMOD International Conference on Management of Data*, 2015.

Mariano Tepper, Ishwar Singh Bhati, Cecilia Aguerrebere, Mark Hildebrand, and Theodore L. Willke. LeanVec: Searching vectors faster by making them fit. *Transactions on Machine Learning Research*, 2024.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.

Chaojun Xiao, Pengle Zhang, Xu Han, et al. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 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 (ICLR)*, 2024b.

An Yang et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Hailin Zhang, Xiaodong Ji, et al. PQCache: Product quantization-based KVCache for long context LLM inference. In *Proceedings of the ACM on Management of Data (SIGMOD)*, 2025a.

Yanqi Zhang, Yuwei Hu, Runyuan Zhao, John C.S. Lui, and Haibo Chen. DiffKV: Differentiated memory management for large language models with parallel KV compaction. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2025b.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, et al. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Appendix Contents

A. **Extended Related Work**

B. **Theory and Proofs**

B.1. Theorem 1: Deterministic Sound Band

B.2. Theorem 2: Pairwise Residual Coverage

B.3. Theorem 3: GQA Score-Representation Bytes

C. **Calibration Diagnostics**

D. **Extended Experiment Results**

D.1. Calibration Variants and Main-Table Provenance

D.2. Sound-Bound Admitted Fractions

D.3. Operating-Point Grids (m, α, k)

D.4. LongBench and RULER Per-Task Tables

D.5. Llama-3.1-70B Results

E. **Experimental Protocol and Baseline Reproduction**

A Extended Related Work

The comparison in Section 2 distinguishes eviction, retrieval, and budget estimation. This section adds details about how long a selection persists, how scores are approximated, and what is fitted offline.

Eviction and per-query selection. Eviction policies use sinks and recency (Xiao et al., 2024b), heavy-hitter statistics (Zhang et al., 2023), or per-token compaction (Zhang et al., 2025b); an eviction persists across later decode queries. Per-query retrieval instead rebuilds the selected set through page scores (Tang et al., 2024), LSH sampling (Chen et al., 2025), product quantization (Zhang et al., 2025a), ANN search (Liu et al., 2025; Xiao et al., 2024a), or low-rank landmarks (Sun et al., 2025). ReFreeKV stops pruning through a distribution-level criterion (Ni et al., 2026). Threshold-recall methods include Louver's ball-halfspace gate (Dehghankar & Asudeh, 2026), Cauchy-Schwarz MIPS bands (Teflioudi et al., 2015; Li et al., 2017), and Quest page bounds; vAttention supplies a sampling-based recall guarantee (Desai et al., 2026).

Approximate-score selectors. Loki fits a PCA basis offline from key states. At decode, the leading PCA coordinates rank the cache under a fixed top-k budget, after which full-dimensional scores compute attention over the selected entries (Singhania et al., 2024). SparQ selects the largest-magnitude coordinates of each query, ranks keys through the matching coordinates, includes a recent window, and corrects for unselected values (Ribar et al., 2024). Both approximations choose entries for a supplied count.

CalibKV fits its basis to queries and uses low-dimensional scores before decode to estimate B_h. Decode then selects the exact top B_h entries. This separates count estimation from the approximate ranking performed by Loki and SparQ.

Other data-dependent structures. ShadowKV models low-rank structure in the keys (Sun et al., 2025). A2ATS, SAAP, ParisKV, and Spotlight Attention construct other shared offline representations (He et al., 2025; Mazaré et al., 2025; Qi et al., 2026; Li et al., 2025). Query-head-specific representations incur the GQA score traffic quantified by Theorem 3.

Learned indexes and calibration. Classical learned indexes model the data distribution to predict record locations, lookup ranges, or multidimensional layouts (Kraska et al., 2018; Ferragina & Vinciguerra, 2020; Nathan et al., 2020). CalibKV fits a projection for attention-score approximation. This use of the data distribution is closer to LeanVec's dimensionality reduction for vector search (Tepper et al., 2024). The residual quantile uses the finite-sample rank argument from split conformal prediction (Vovk et al., 2005; Papadopoulos et al., 2002; Lei et al., 2018; Angelopoulos & Bates, 2021).

B THEORY AND PROOFS

This section establishes the deterministic Cauchy-Schwarz band, finite-sample pairwise residual coverage, the threshold ordering used by the rank-m diagnostic, and the score-representation byte model for GQA.

B.1 THEOREM 1: DETERMINISTIC SOUND BAND

Theorem 1 (Sound band). *Fix $U \in \mathbb{R}^{d\times m}$ with orthonormal columns and a mean key $\bar{\kappa}$. Define $\hat{s}(q,\kappa) = q^\top\bar{\kappa} + (U^\top q)^\top U^\top(\kappa-\bar{\kappa})$, $\delta(q) = \|(I-UU^\top)q\|$, and $R(\kappa) = \|(I-UU^\top)(\kappa-\bar{\kappa})\|$. Then, for every q and κ,*

$$\left|q^\top\kappa - \hat{s}(q,\kappa)\right| \;\le\; \delta(q)R(\kappa),$$

and $\{\kappa : \hat{s}(q,\kappa) + \delta(q)R(\kappa) \ge \tau\}$ contains every key satisfying $q^\top\kappa \ge \tau$. Thus, its recall relative to the supplied threshold is 1.

Proof. Let $P = I - UU^\top$. Since P is a symmetric orthogonal projector, the residual is $q^\top P(\kappa-\bar{\kappa}) = (Pq)^\top P(\kappa-\bar{\kappa})$. Cauchy-Schwarz gives the stated bound. In particular, $q^\top\kappa \le \hat{s}(q,\kappa) + \delta(q)R(\kappa)$, so every key with $q^\top\kappa \ge \tau$ is admitted. □

The guarantee concerns recall at a supplied threshold. Figure 1 and Table 7 measure the separate question of how many keys the band admits.

B.2 THEOREM 2: PAIRWISE RESIDUAL COVERAGE

Theorem 2 (Pairwise residual coverage). *Let $Z_1,\dots,Z_N, Z_*$ be exchangeable query-key pairs, with $Z_i = (q_i,\kappa_i)$, and let $r(Z) = q^\top\kappa - \hat{s}(q,\kappa)$. Assume that every fitted quantity in $\hat{s}$, including its subspace and centering, is independent of these pairs. Set $j = \lceil (N+1)\alpha\rceil$ for $\alpha \in (0,1)$ and let $\hat{r}_\alpha$ be the j-th smallest value among $r(Z_1),\dots,r(Z_N)$, with $\hat{r}_\alpha = +\infty$ when $j = N+1$. Then*

$$\Pr[r(Z_*) \le \hat{r}_\alpha] \ge \alpha.$$

For any threshold τ, including one determined by the test query and cache, define $C(q,\tau) = \{\kappa : \hat{s}(q,\kappa) + \hat{r}_\alpha \ge \tau\}$. The joint probability of relevance and exclusion satisfies

$$\Pr\left[\kappa_* \notin C(q_*,\tau) \;\wedge\; q_*^\top\kappa_* \ge \tau\right] \le 1-\alpha.$$

Proof. Condition on the fitted quantities in $\hat{s}$. With independent random tie-breaking, exchangeability makes the test residual's rank uniform on $\{1,\dots,N+1\}$. For $j \le N$, the strict event $r(Z_*) > \hat{r}_\alpha$ implies that this rank exceeds j, so its probability is at most $(N+1-j)/(N+1) \le 1-\alpha$. For $j = N+1$, the event is empty. If a relevant key is excluded, then $\hat{s}(q_*,\kappa_*) + \hat{r}_\alpha < \tau \le q_*^\top\kappa_*$, which implies $r(Z_*) > \hat{r}_\alpha$. The joint miss event is therefore contained in the residual-tail event. □

Cache-level consequence. Let (q,K) be a random query and length-n cache. Sample J uniformly from $\{1,\dots,n\}$, independently of the calibration data and the query and cache. Suppose $Z_* = (q,\kappa_J)$ satisfies the assumptions of Theorem 2 with the calibration pairs. If exactly $k \ge 1$ cache positions satisfy the relevance threshold and M counts relevant positions excluded by the band, then

$$\mathbb{E}[M] \le n(1-\alpha), \qquad \mathbb{E}[\text{recall}] \ge \max\left\{0, 1 - \frac{n(1-\alpha)}{k}\right\}.$$

Indeed, $\mathbb{E}[M]/n$ is the joint miss probability of the uniformly sampled position. Since its relevance probability is k/n, the conditional miss probability given relevance is at most $\min\{1, n(1-\alpha)/k\}$.

Connection to the exact-score estimator. The theorem's random unit is one query-key pair. The $256 \times n$ crossed matrix used for exact-score budget estimation reuses every offline query across request keys and every request key across offline queries, producing the dependence described in Section 6. Section 5 evaluates quality at the estimated budgets; Appendix C reports the separate rank-m screening diagnostics.

Lemma 1 (Two-stage threshold ordering). *For any $I \subseteq [n]$ with $|I| \geq k$, the k-th largest exact score within I is no greater than the global k-th score τ_k^*. In particular, let I contain the $\min\{2k, n\}$ largest approximate scores and assume $n \geq k$. The resulting threshold satisfies $\tau_{\text{est}} \leq \tau_k^*$, so the band at τ_{est} contains the band at τ_k^*.*

Proof. The first statement follows from $I \subseteq [n]$. Since $C(q, \tau_1) \supseteq C(q, \tau_2)$ whenever $\tau_1 \leq \tau_2$, substituting $\tau_{\text{est}} \leq \tau_k^*$ gives the result. □

The lemma compares the two thresholds used by the rank-m diagnostic; Theorem 2 gives the residual-quantile statement.

B.3 Theorem 3: GQA Score-Representation Bytes

Rank-m scoring projects the query, scores n projected keys, selects a top-$2k$ seed, scores that seed exactly, and scores the admitted candidates exactly. The exact-score path scans the native keys, selects top-B_h, and attends over those entries. The following model isolates the score-representation traffic that distinguishes the two layouts.

Theorem 3 (GQA score-representation bytes). *Assume fp16 storage, H_q query heads, H_{kv} KV heads, head dimension d, and cache length n. Suppose an exact-score kernel reads each native KV key once and reuses it across the query heads in its GQA group, while the rank-m layout stores and reads a separate projected key for each query head. Exact scoring then reads $2nH_{\text{kv}}d$ bytes per decode step, and rank-m scoring reads $2nH_q m$ bytes. The projected representation has lower read traffic exactly when*

$$m < d\,\frac{H_{\text{kv}}}{H_q}.$$

Proof. Each of the H_{kv} native key heads stores d fp16 values per cache position, giving $2nH_{\text{kv}}d$ bytes. A query-head-specific projection stores m fp16 values for each of the H_q query heads, giving $2nH_q m$ bytes. Comparing the two expressions gives the condition. □

This count includes only the key representations read to compute scores. Total decode cost also includes query projection, selection, value access, output writes, and any further cache reuse. For Llama-3.1-8B, $H_q = 32$, $H_{\text{kv}} = 8$, and $d = 128$, so the projected representation reads fewer bytes when $m < 32$; the evaluated rank is $m = 64$. Section 5.4 measures native-GQA full-model requests and a separate 32-head attention calculation.

C Calibration Diagnostics

Section 4.3 states the four calibration requirements and their main effects. The controls below provide the additional comparisons.

L1: separation control. At 64K and $m = 64$, reusing the PCA fit queries as Q_{cal} makes $\hat{r}_{\alpha,h} \approx 0$: the candidate fraction drops to ∼0.1%, but the rank-m true-top-k miss rate rises to 0.92. An even/odd split gives a 0.12 miss rate at a 13-14% candidate fraction. Both configurations score 1.0 on needle retrieval, so full-score evaluation distinguishes their screening behavior. At 64K and $m = 64$, reuse and even/odd splitting produce ΔNLL pairs of $+0.38/+0.47$ and $+0.21/+0.18$ on War and Peace/Moby-Dick, respectively. In the 16K control, applying the in-sample setting after the split raises the fraction to 33.7% at $+0.053$ ΔNLL; retuning α gives the 16.4%, $+0.087$ point in Table 4.

L2: source attribution. Prefill-query calibration paired with a decode-fitted U_h admits 95-97% of the cache. A bank covering a narrow set of content admits 1.0% while missing 80% of the true top-k keys in the screened path. A bank matched to decode traffic gives the 16.4% candidate fraction at 16K reported in Section 4.3. Replacing GQA-pooled subspaces with per-query-head fits moves the 8K fraction from 98.6% to 97.1%, and an independent recapture reproduces the high-fraction result. Matching each bank to its own $(U_h, \hat{r}_{\alpha,h})$ accounts for the large reduction reported in Section 4.3. Table 5 separates cross-request, self-request, and combined sources at 64K.

L3: position control. At distant positions, a subspace fitted on mixed 16K and 64K queries captures 0.67-0.70 of the squared query norm, versus more than 0.85 at 16K. Its ΔNLL is $+0.18 \pm 0.09$ at a 13% candidate fraction. With four position-matched request captures instead of two, the two reported segment differences fall to $+0.098$ and $+0.123$ at 2.0-2.2% (Table 4).

L4: outlier accounting. Sink and norm-outlier pairs inflate the residual quantile, producing a 97.1% candidate fraction at 16K. Moving these keys to the always-attend set reduces the fraction to 94.8%; traffic matching in L2 supplies the larger reduction to 16.4%. The paired control changes only whether sink and norm-outlier keys enter the crossed residual matrix or the always-attend set. Reported budgets charge dense-routed heads as 1; Section 5 accounts separately for the sink/recent/norm-outlier union.

Rank-m top-k miss rates. Full-score evaluation puts the rank-m true-top-k miss rate between 0.11 and 0.44. On one 16K continuation run, the observed rate is 0.44 and the clipped pair-level expression gives 0.13. The expression assumes the sampling conditions of Theorem 2; it is not a certified miss-rate bound for the crossed calibration matrix used by CALIBKV. These measurements assess the screened path and motivate the traffic- and position-matching controls above. Exact-score CALIBKV instead selects the exact top B_h.

D EXTENDED EXPERIMENT RESULTS

D.1 CALIBRATION VARIANTS AND MAIN-TABLE PROVENANCE

Table 4 collects disjoint-split Llama-3.1-8B variants omitted from the main table: a lower α, a length-conditioned subspace, median-γ variants, and the strengthened position-matched setting. Each ΔNLL uses one segment unless the row identifies War and Peace (WP) or Moby-Dick (MD).

Table 4: **Calibration variants on Llama-3.1-8B.** Candidate fractions and ΔNLL relative to Dense use disjoint fitting and calibration queries. Each ΔNLL uses one text segment unless noted.

Point	Candidate frac.	ΔNLL	Notes
16K, α=0.999	16.4%	$+0.087$	needle 100%; 16K reference
16K, α=0.99	5.7%	$+0.150$	needle 100%
64K, α=0.999	6.1%	$+0.124$	Oracle $+0.117$ at 0.2%
64K, length-conditioned U, α=0.9999	15.6%	$+0.147$	mixed-fit U: $+0.179$
γ-median variants	3-11%	$+0.12$-0.18	range over the variant family
64K-strengthened (L3)	2.0-2.2%	$+0.098$ (WP) $+0.123$ (MD)	position-matched

For the 32K rank-m row in Table 1, replacing the position-matched subspace with a 16K-fit subspace changes ΔNLL from $+0.139$ to $+0.261$ and the budget from 3.2% to 12.7%. This intervention holds the evaluation segment and α fixed. Figures 4 and 5 show the Llama-3.1-8B context-length trend and rank-m relationship between quality and candidate fraction at the default $k = 128$.

Context-dependent α gives candidate fractions of 16.4%, 4.2%, 3.2%, and 3.3% from 16K to 128K, at most 1.6 points above the measured budgets. Conditioning the quantile on relevant pairs lowers the measured miss rate from 0.18 to 0.11 at 64K and from 0.44 to 0.37 at 16K. Table 5 compares cross-request, self-request, and combined calibration at 64K ($m = 16$, $\alpha = 0.9999$); Table 6 varies the MATH-500 generation cap.

Table 5: **Calibration sources at 64K.** True-top-k recall, candidate fraction, and measured residual violation rate for $m = 16$ and $\alpha = 0.9999$.

Calibration source	Recall	Candidate frac.	Eval. viol. rate
cross-request only	0.9130	4.3%	1.5×10^{-2}
prefill (self)	0.9961	13.3%	8.3×10^{-4}
both	**0.9985**	12.8%	$\mathbf{3.0 \times 10^{-4}}$

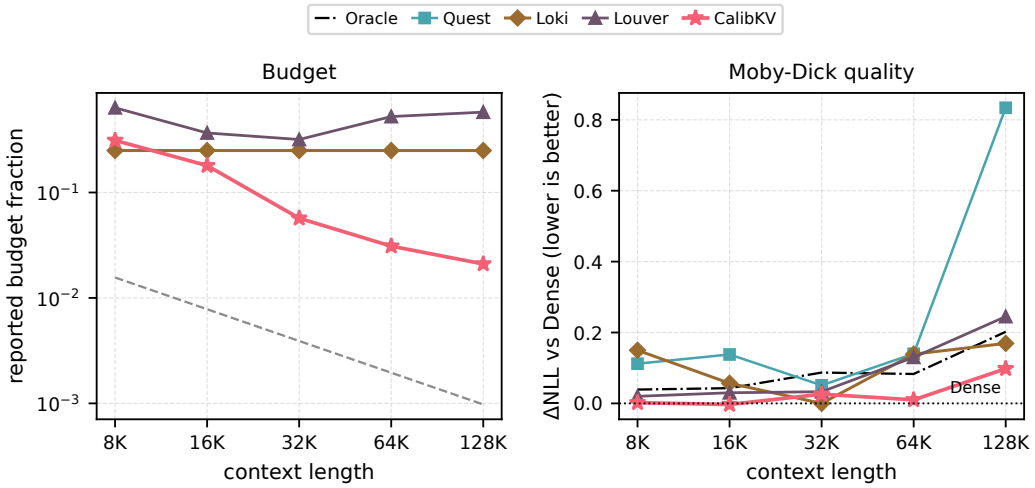

Figure 4: **Evaluated budgets across context lengths.** Left: normalized budget; the fixed-k reference follows k/n, Loki remains at 25%, and the adaptive methods follow their measured counts. Right: ΔNLL on the Moby-Dick segment at the same evaluated operating points.

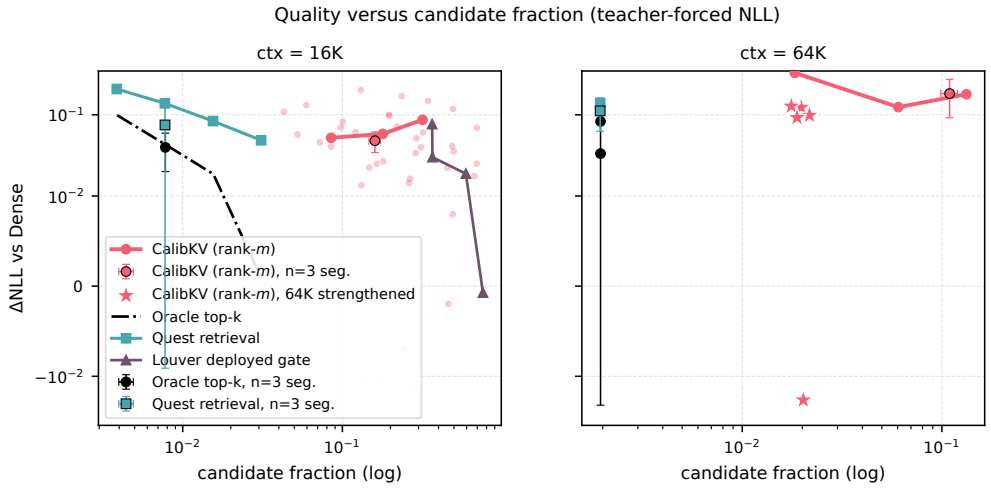

Figure 5: **Quality and candidate fraction.** The CALIBKV curves show rank-m screening. Teacher-forced token ΔNLL relative to Dense at 16K (left) and 64K (right). Three-segment means are marked in the legend; error bars show one standard deviation.

Table 6 reports normalized budget fractions. "Hit 4096 cap" counts examples that reach 4096 generated tokens without a final answer. Wall-clock values use the 100-example run at that limit on one A100; the CALIBKV budget 0.9987 is the unrounded mean from the 500-example run. Loki uses rank 32 and its fixed 25% budget.

D.2 SOUND-BOUND ADMITTED FRACTIONS

Figure 1 uses Llama-3.1-8B and Moby-Dick at token offset 40,000, with a 16K or 64K prefill and 128 teacher-forced decode steps. Each bar averages 8,192 observations: all 32 query heads in layers $0, 4, \ldots, 28$ at decode steps $0, 4, \ldots, 124$. The three sound-bound references share dense query/key captures, including keys appended up to each step. C-S uses an independently fitted rank-64 query basis from a disjoint span at offset 110,000. Its threshold is the 128th exact score among its 768 highest approximate-score keys. Quest uses 16-key pages and takes the 128th exact score among keys

Table 6: **MATH-500 quality and runtime.** Accuracy is measured at the stated generation cap and sample size; "Hit 4096 cap" and wall-clock time use 100 examples. Budgets are normalized fractions. Bold marks the best sparse-method result in each column.

Method	Budget	Acc.@2048 (100)	Acc.@4096 (100)	Acc.@2048 (500)	Hit 4096 cap	Wall-clock
Dense	1.0	48.0%	67.0%	44.6%	22/100	1.4 h
Oracle	0.08-0.09	41.0%	50.0%	38.6%	39/100	6.7 h
Quest	0.08-0.09	41.0%	47.0%	37.2%	41/100	10.7 h
Loki	0.25	41.0%	50.0%	39.2%	44/100	**2.1 h**
Louver	0.88-0.92	45.0%	55.0%	41.0%	34/100	11.7 h
CALIBKV	0.9987	**50.0%**	**62.0%**	**46.6%**	**27/100**	**2.1 h**

in its 48 highest-bound pages. These are screening reference protocols, not fixed-budget inference rules. Neither uses a dense cutoff or an added sink/recent union.

Louver uses the enclosing-ball formula and threshold scan in its paper's Algorithm 4. We reuse the released balanced PCA-tree assignments with four coordinate subspaces and four keys per cluster, and compute enclosing centers and radii in float64. The threshold comes from its released 512-key SampleMax reservoir, with seed 0. The count includes the visited-cluster union before exact post-filtering and all appended keys in the native buffer. This is a paper-formula reference, not the released centroid-only TA kernel. All three references had zero above-threshold omissions in the measured captures. Soundness is relative to each method's threshold; the reservoir threshold need not retain the exact top-128 set.

The exact-score CALIBKV bars use the same text and sampling grid, including their own sparse-attention trajectory. They count the union of exact top-B_h, sink/recent, and norm-outlier keys; dense-routed heads contribute one. The existing 16K and 64K artifacts have actual ranks 27 and 56, respectively. The selected fractions are 19.50% and 3.66%; before the always-attend union, the corresponding budgets are 19.39% and 3.56%. Averaging all layers and decode steps instead gives budgets of 17.99% and 3.13%.

Table 7 retains the earlier fixed-threshold rank sweep and its original median aggregation. Its block-ball row sums coordinate-block ball bounds at supplied thresholds. Figure 1 instead measures the paper's cluster-list scan at reservoir thresholds. These protocols and their aggregation differ; the table is not a breakdown of the figure. The 16K/64K labels denote context length.

Table 7: **Sound-bound admitted fractions.** Median percentages at top-k and fixed-budget thresholds; 100.0% denotes the full cache. This earlier threshold sweep is separate from Figure 1.

				top-k threshold		budget threshold		
Model	Method	Ctx	Config	$k = 32$	$k = 128$	2%	5%	10%
Llama-3.1-8B	C-S band	16K	m=8	94.1%	98.8%	99.7%	99.9%	100.0%
Llama-3.1-8B	C-S band	16K	m=16	78.4%	92.5%	97.2%	99.1%	99.7%
Llama-3.1-8B	Quest bound	16K	page 16	100.0%	100.0%	100.0%	100.0%	100.0%
Llama-3.1-8B	Block-ball bound	16K	S=4	98.4%	99.6%	99.9%	100.0%	100.0%
Qwen2.5-7B	C-S band	16K	m=8	88.7%	97.2%	99.2%	99.8%	100.0%
Qwen2.5-7B	C-S band	16K	m=16	68.8%	87.6%	95.2%	98.4%	99.5%
Qwen2.5-7B	Quest bound	16K	page 16	100.0%	100.0%	100.0%	100.0%	100.0%
Qwen2.5-7B	Block-ball bound	16K	S=4	98.9%	99.8%	100.0%	100.0%	100.0%
Llama-3.1-8B	C-S band	64K	$m = 8$	76.4%	90.1%	99.1%	99.8%	100.0%
Llama-3.1-8B	C-S band	64K	$m = 16$	55.3%	74.9%	95.2%	98.4%	99.5%
Llama-3.1-8B	Quest bound	64K	page 16	99.4%	99.9%	100.0%	100.0%	100.0%
Llama-3.1-8B	Block-ball bound	64K	$S = 4$	81.6%	91.6%	98.8%	99.6%	99.9%

The 16K rows summarize three requests, with 1024 head units for Llama and 784 for Qwen. The 64K Llama sweep uses three requests and five sampled layers. The 16K Llama Quest row uses regenerated tensors under the original configuration. The 64K Quest row uses recaptures of the same three texts and 16 sampled decode queries per head, giving 7,680 observations per threshold. Figure 1 instead reports the 98.36% mean under its per-method seed-threshold protocol.

D.3 OPERATING-POINT GRIDS (m, α, k)

Rank m. Table 8 summarizes the 16K Llama-3.1-8B in-sample rank sweep. Candidate fractions come from the needle grid used in Figure 3; ΔNLL comes from separate teacher-forced language-modeling evaluations. The candidate column uses a 64-query bank for $m = 8, 16, 32, 64$ and a 256-query bank at the full-rank endpoint $m = d = 128$. Table 4 gives the disjoint-split reference results. Increasing the bank from 64 to 1024 queries changes the candidate fraction by at most two percentage points across the plotted grid.

Table 8: **16K rank sensitivity on Llama-3.1-8B.** Candidate fractions use the in-sample needle grid; ΔNLL relative to Dense comes from separate language-modeling evaluations.

m	Candidate frac.	ΔNLL	Notes
8	77%	$+0.092$	large residuals
16	57%	$+0.059$	
32	20%	$+0.044$	varying α gives 9.6-34% at $+0.041$-0.070
64	4.1%	$+0.043$	close to Oracle on this segment; reference config
128 ($= d$)	0.8%	$+0.046$	near-exact endpoint

Component-level recall/budget grid. The component-level grid uses captured 16K decode tensors, $m = 16$, $\alpha = 0.9999$, and an oracle threshold. Entries are medians over 784 to 1024 head units. On Qwen2.5-7B, top-128 and top-32 thresholds give candidate fractions of 7.2% and 1.8%, with recalls of 0.994 and 0.989. A threshold based on a 2% budget gives a 17.4% candidate fraction and recall of 0.996. The subspace-transfer ratio is 0.90.

On Llama-3.1-8B, the top-128 threshold gives an 8.0% candidate fraction at recall 0.993. A threshold based on a 2% budget gives 18.7% at recall 0.996, with transfer ratios of 0.96 to 0.98. In the Qwen threshold-estimation control, replacing the oracle threshold with the estimated threshold changes the candidate fraction from 8.0% to 14.2% and recall from 0.994 to 0.993.

Residual-quantile level α. At 64K, $\alpha = 0.9999$ with combined self- and cross-request inputs gives a measured violation rate of 3.0×10^{-4} (Table 5). Figure 3 reports the budget and miss-rate sweep.

D.4 LONGBENCH AND RULER PER-TASK TABLES

LongBench runs use the official templates, generation lengths, and metrics: qa_f1 for question answering, Rouge-L F for gov_report, classification accuracy for trec, and edit similarity for lcc. Each subtask contains 100 examples. Table 9 reports ten subtasks for each of three models. Table 10 reports Llama-3.1-8B-Instruct RULER cells as EM (all answer strings correct) and the official string_match_all score; each variable-tracking cell has 100 examples and each needle cell has 25. Parenthesized values are normalized mean budgets, with Dense equal to 1; rank-m runs use the defaults from Section 5. Bold marks the best non-Dense score, including ties, and pink columns mark CALIBKV.

Qwen RULER. On Qwen2.5, the rank-m diagnostic uses normalized budgets of 0.05 to 0.12. Dense/rank-m variable-tracking EM is 52/36 at 16K and 64/52 at 32K on 7B; on 14B it is 52/48 and 36/16. The four-cell Dense/rank-m EM macro is 77/70 on 7B and 72/66 on 14B. Loki's fixed 0.25 budget gives 96/76/32/56 EM on the 7B niah@16K, niah@32K, vt@16K, and vt@32K cells, and 100/100/42/35 on 14B.

Table 9: **LongBench per-task results.** Each task has 100 examples. Scores are followed by normalized mean key fractions in gray; macro averages the ten task scores. Bold marks the best sparse-method score.

Model	Subtask	Dense	Oracle	Quest	Loki	Louver	CALIBKV	CALIBKV (rank-m)
Llama-3.1-8B	narrativeqa	23.54	23.70 (.009)	22.76 (.009)	23.71 (.25)	23.69 (.38)	23.30 (.18)	**23.98** (.21)
	qasper	42.11	**41.92** (.029)	38.12 (.029)	41.61 (.25)	41.65 (.69)	41.51 (.33)	40.34 (.41)
	multifieldqa_en	56.10	55.04 (.027)	55.19 (.027)	55.91 (.25)	55.64 (.62)	55.87 (.32)	**56.33** (.37)
	hotpotqa	52.88	52.80 (.011)	51.09 (.011)	52.86 (.25)	**53.09** (.41)	52.80 (.20)	52.67 (.22)
	2wikimqa	47.12	46.24 (.023)	46.94 (.022)	46.54 (.25)	47.48 (.61)	46.76 (.29)	**48.59** (.33)
	musique	33.46	33.16 (.009)	31.88 (.009)	**33.78** (.25)	33.41 (.32)	33.66 (.17)	33.77 (.19)
	gov_report	19.79	19.71 (.016)	17.54 (.016)	19.58 (.25)	**19.93** (.49)	19.55 (.24)	19.25 (.25)
	trec	67.00	**67.00** (.024)	65.00 (.024)	**67.00** (.25)	**67.00** (.65)	**67.00** (.46)	**67.00** (.48)
	triviaqa	17.73	17.99 (.017)	17.96 (.017)	**18.48** (.25)	17.87 (.52)	18.04 (.26)	17.89 (.28)
	lcc	62.22	62.58 (.055)	61.91 (.055)	61.88 (.25)	**62.83** (.78)	62.29 (.57)	62.64 (.60)
	macro	42.20	42.01	40.84	42.14	**42.26**	42.08	42.25
Qwen2.5-7B	narrativeqa	26.58	26.14 (.009)	24.95 (.009)	23.37 (.25)	26.70 (.38)	**27.14** (.09)	26.43 (.07)
	qasper	43.07	**43.67** (.029)	39.70 (.028)	41.93 (.25)	43.56 (.65)	43.28 (.18)	42.98 (.15)
	multifieldqa_en	52.58	**52.13** (.026)	50.36 (.026)	51.11 (.25)	50.80 (.59)	51.96 (.17)	51.09 (.14)
	hotpotqa	53.38	52.98 (.011)	50.40 (.011)	50.28 (.25)	52.36 (.40)	**53.33** (.11)	52.33 (.08)
	2wikimqa	48.57	48.26 (.023)	44.55 (.022)	45.39 (.25)	48.40 (.59)	48.24 (.16)	**49.24** (.13)
	musique	28.53	28.60 (.009)	19.82 (.008)	24.90 (.25)	**29.07** (.33)	28.62 (.09)	28.88 (.07)
	gov_report	18.01	17.68 (.015)	16.49 (.015)	**18.85** (.25)	18.14 (.48)	17.98 (.13)	17.56 (.09)
	trec	68.00	**68.00** (.024)	64.00 (.024)	65.50 (.25)	**68.00** (.61)	**68.00** (.18)	**68.00** (.16)
	triviaqa	18.42	18.23 (.016)	19.20 (.016)	17.54 (.25)	**21.87** (.49)	18.93 (.13)	18.75 (.10)
	lcc	61.78	60.69 (.055)	56.62 (.054)	58.43 (.25)	59.72 (.73)	**61.18** (.29)	59.76 (.26)
	macro	41.89	41.64	38.61	39.73	41.86	**41.87**	41.50
Qwen2.5-14B	narrativeqa	29.83	29.03 (.009)	24.91 (.009)	29.09 (.25)	**29.68** (.40)	29.61 (.09)	28.76 (.12)
	qasper	46.05	**47.64** (.029)	43.80 (.029)	46.74 (.25)	47.47 (.67)	46.51 (.19)	43.52 (.20)
	multifieldqa_en	51.75	50.33 (.026)	45.07 (.026)	50.35 (.25)	49.53 (.61)	**50.93** (.18)	47.49 (.21)
	hotpotqa	58.05	57.60 (.011)	51.89 (.011)	56.71 (.25)	**57.87** (.42)	57.43 (.10)	56.07 (.11)
	2wikimqa	53.85	53.35 (.023)	50.64 (.023)	**54.68** (.25)	53.35 (.61)	53.35 (.17)	53.97 (.17)
	musique	35.96	35.93 (.009)	31.34 (.008)	**36.86** (.25)	36.07 (.36)	35.70 (.09)	34.93 (.10)
	gov_report	16.94	17.19 (.016)	15.38 (.016)	**17.31** (.25)	16.92 (.49)	16.97 (.12)	13.72 (.16)
	trec	71.00	**71.00** (.024)	69.00 (.024)	**71.00** (.25)	**71.00** (.63)	**71.00** (.22)	**71.00** (.27)
	triviaqa	19.28	18.92 (.016)	18.55 (.016)	19.09 (.25)	**19.86** (.50)	18.31 (.14)	19.57 (.15)
	lcc	58.31	**60.86** (.055)	58.69 (.054)	59.30 (.25)	60.31 (.74)	60.03 (.35)	58.67 (.34)
	macro	44.10	44.19	40.93	44.11	**44.21**	43.98	42.77

Table 10: **Llama-3.1-8B RULER results.** Cells report exact match/string match, followed by normalized mean key fractions in gray. Bold marks the best sparse-method exact match.

Cell	Dense	Oracle	Quest	Loki	Louver	CALIBKV	CALIBKV (rank-m)
niah_mk@16K	100/100	**100/100** (.008)	**100/100** (.008)	**100/100** (.25)	**100/100** (.33)	**100/100** (.16)	**100/100** (.17)
niah_mk@32K	100/100	**100/100** (.004)	**100/100** (.004)	**100/100** (.25)	**100/100** (.30)	**100/100** (.12)	**100/100** (.11)
vt@16K	94/97.8	83/91.6 (.008)	26/57.4 (.008)	**91/96.0** (.25)	70/87.0 (.35)	**91/95.6** (.16)	80/90.0 (.19)
vt@32K	80/90.0	58/74.2 (.004)	16/51.2 (.004)	68/87.6 (.25)	49/73.0 (.32)	**70/87.2** (.12)	53/70.8 (.12)

D.5 LLAMA-3.1-70B RESULTS

Table 11: **Llama-3.1-70B quality.** Budget is the mean reported key fraction; Louver and rank-m CALIBKV count candidates before reranking. ΔNLL relative to Dense is shown for *Moby-Dick* (MD) and as mean$\pm$SD over three *War and Peace* (WP) segments. Needle gives sparse (Dense) retrieval accuracy.

Method	Ctx	Budget	ΔNLL (MD)	ΔNLL $\pm$ SD (WP)	Needle (Dense)
Oracle	8K	1.6%	-0.018	$+0.072 \pm 0.082$	1.00 (1.00)
Quest	8K	1.5%	$\mathbf{-0.235}$	$\mathbf{-0.096 \pm 0.061}$	1.00 (1.00)
Loki	8K	25.0%	$+0.112$	$+0.019 \pm 0.029$	1.00 (1.00)
Louver	8K	60.0%	-0.022	$+0.064 \pm 0.067$	1.00 (1.00)
CALIBKV	8K	4.5%	$+0.005$	$+0.077 \pm 0.155$	1.00 (1.00)
CALIBKV (rank-m)	8K	3.0%	$+0.030$	$+0.189 \pm 0.075$	1.00 (1.00)
Oracle	16K	0.8%	$+0.229$	-0.001 ± 0.038	1.00 (1.00)
Quest	16K	0.8%	$\mathbf{-0.065}$	$+0.019 \pm 0.051$	1.00 (1.00)
Loki	16K	25.0%	$+0.029$	$\mathbf{-0.020 \pm 0.014}$	1.00 (1.00)
Louver	16K	27.2%	$+0.097$	$+0.213 \pm 0.161$	1.00 (1.00)
CALIBKV	16K	2.2%	$+0.134$	$\mathbf{-0.020 \pm 0.019}$	1.00 (1.00)
CALIBKV (rank-m)	16K	1.7%	$+0.251$	$+0.021 \pm 0.047$	1.00 (1.00)
Oracle	32K	0.4%	$+0.345$	$+0.115 \pm 0.105$	1.00 (1.00)
Quest	32K	0.4%	$\mathbf{-0.138}$	$\mathbf{-0.108 \pm 0.065}$	1.00 (1.00)
Loki	32K	25.0%	$+0.035$	-0.008 ± 0.028	1.00 (1.00)
Louver	32K	21.1%	$+0.514$	$+0.171 \pm 0.147$	1.00 (1.00)
CALIBKV	32K	2.6%	$+0.272$	$+0.169 \pm 0.059$	1.00 (1.00)
CALIBKV (rank-m)	32K	20.1%	$+0.542$	$+0.657 \pm 0.110$	1.00 (1.00)
Oracle	64K	0.2%	$+0.074$	-0.009 ± 0.016	1.00 (1.00)
Quest	64K	0.2%	$+0.095$	$\mathbf{-0.219 \pm 0.100}$	1.00 (1.00)
Loki	64K	25.0%	$\mathbf{+0.002}$	$+0.011 \pm 0.010$	1.00 (1.00)
Louver	64K	38.3%	$+0.076$	$+0.031 \pm 0.046$	1.00 (1.00)
CALIBKV	64K	1.3%	$+0.049$	-0.037 ± 0.018	1.00 (1.00)
CALIBKV (rank-m)	64K	2.9%	$+0.243$	$+0.422 \pm 0.107$	1.00 (1.00)
Oracle	128K	0.1%	$+1.140$	$+0.816 \pm 0.307$	1.00 (1.00)
Quest	128K	0.1%	$+1.081$	$+0.613 \pm 0.155$	0.50 (1.00)
Loki	128K	25.0%	$+0.513$	$+0.257 \pm 0.046$	1.00 (1.00)
Louver	128K	47.0%	$+1.009$	$+0.690 \pm 0.252$	1.00 (1.00)
CALIBKV	128K	0.7%	$\mathbf{+0.260}$	$\mathbf{+0.153 \pm 0.043}$	1.00 (1.00)
CALIBKV (rank-m)	128K	3.6%	$+3.168$	$+1.906 \pm 0.243$	1.00 (1.00)

Table 11 reports the bf16 Llama-3.1-70B sweep from 8K to 128K on four A100-80GB GPUs in one node. The lowest ΔNLL varies by method and text from 8K to 64K. Exact-score CALIBKV lowers its mean budget from 4.5% at 8K to 0.7% at 128K. At 128K, it has the lowest ΔNLL among the sparse methods on both texts: $+0.260$ on *Moby-Dick* and $+0.153$ on *War and Peace*. It matches Dense on the reported needle test, while Quest retrieves half of the needles. The rank-m path admits candidates equal to 3.6% of the cache and gives $+3.168$ and $+1.906$ ΔNLL on the two texts.

E EXPERIMENTAL PROTOCOL AND BASELINE REPRODUCTION

E.1 SHARED MEASUREMENT PROTOCOL

Quality runs use bf16 decode with batch size 1. Models up to 14B run on one A100-80GB GPU; the 70B sweep uses four A100-80GB GPUs.

The full-model GPU rows in Table 3 use bf16 Llama-3.1-8B on one A100-80GB GPU with native $H_q = 32$, $H_{\text{kv}} = 8$ GQA. Each length uses three prompts and three measured repetitions per prompt after warmup. The near-128K requests contain 130,816 prompt tokens, followed by 256 generated tokens. Prefill includes the first forward pass and generated token; decode covers the remaining 255 forwards. End-to-end time covers the complete timed request. Model loading, tokenization, compilation, warmup, and reusable fitting are outside this request clock.

Offline times measure capture and fitting once per reusable artifact, excluding model loading and tokenization.

The attention column uses a separate single-layer fp16 input with 32 independent heads and $d = 128$ on the same A100; its 128K input has 131,072 keys. It times each method's scoring, selection where applicable, and attention with CUDA events after warmup. These measurements are separate from the full-model decode times.

E.2 Quest Reproduction Details

Quest ranks 16-key pages by min/max score bounds. The book-passage comparison uses a 128-key budget.

E.3 Loki Reproduction Details

Loki uses rank 32 and a fixed 25% key budget, without a sink or recent window. A PCA basis is fitted for each layer and KV head on held-out WikiText-103 validation text, separately for each model. Cached projected keys rank the cache; selected original keys and values then enter exact attention.

E.4 Louver Reproduction Details

Louver uses a subspace-ball gate with a reservoir-based threshold estimate. Its reported quality fraction counts candidates before exact top-128 reranking. The GPU timing uses the released TA path and includes request-index construction in prefill.