RECAP-KV: SINGLE-PREFILL APPROXIMATION OF RECONSTRUCTION IMPORTANCE FOR REUSABLE KV CACHE EVICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Key–value (KV) caching avoids recomputing the attention keys and values of processed tokens during generation. It also lets a long document, once processed, serve many subsequent queries. However, the cache grows with context length, limiting batch size and the number of caches that can remain resident in GPU memory. KV eviction addresses this by retaining only a subset of the cached KV pairs. Yet identifying KV pairs that preserve answer quality for unknown future queries is challenging, and reliable importance estimation is expensive for long contexts. We present RECAP-KV, a query-agnostic eviction method that reduces this cost while maintaining competitive answer quality. It applies the model's existing Q/K projections and normalization to attention-input states captured during prefill and uses rotary positional embeddings at virtual positions to approximate the attention that cached keys would receive during context reconstruction, without an additional transformer pass or a learned importance predictor. RECAP-KV allocates the cache budget across layers, heads, and context blocks, and its Lookback Reranking refines token selection without changing the number of KV pairs retained in each block. On 12 tasks and 6 models from the Qwen, Llama, and GPT-OSS families, RECAP-KV achieves the best or near-best mean task scores among query-agnostic methods at 15% and 30% KV retention, while its prefill-and-scoring time at 128K tokens is 37–46% lower than that of the strongest baseline under matched settings on B300 GPUs.

1 INTRODUCTION

Key–value (KV) caching reduces the cost of large language model inference by reusing previously computed attention keys and values. When multiple requests share a document or knowledge source, reusing its context cache also avoids repeated prefill computation. However, cache memory grows with context length, limiting batch size and the number of shared caches that can remain resident in GPU memory (Kwon et al., 2023). This memory constraint motivates KV cache compression. KV eviction is one such approach: it reduces the cache size by retaining only a subset of cached pairs (Zhang et al., 2023).

Choosing this subset is challenging when the cache is reused across requests. For a single request, eviction can use the current query or attention observed during decoding. A reusable context cache, however, is compressed before the future queries it will serve are available. Different queries may depend on different parts of the context, so a subset selected for one query may discard information needed by another. This creates a need for query-agnostic importance estimates that preserve answer quality across diverse future queries while keeping the cost of scoring the context low.

Context reconstruction, which appends the context again after its cache and measures the attention each re-fed token assigns to the cached keys, provides one way to obtain such an importance signal. The context itself serves as the reconstruction target, allowing cached KV pairs to be scored without specifying a downstream task. KVzip, which introduced this approach, scores each pair by the maximum of these attention weights and shows that retaining pairs selected by this criterion can preserve performance across diverse downstream tasks (Kim et al., 2025). Obtaining these scores directly, however, requires additional forward processing through the model, increasing cache-

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.

construction cost for long contexts. Learned alternatives avoid this additional pass during cache construction by using predictors trained offline on reconstruction-derived importance targets (Kim et al., 2026; Jégou & Jeblick, 2026), but each model then requires its own trained predictor. These limitations motivate estimating reconstruction importance from computation already performed during prefill, without an additional reconstruction pass or a separately trained predictor.

To this end, we propose RECAP-KV, a query-agnostic eviction method designed to reduce scoring cost while maintaining competitive answer quality. Our key idea is to reuse states captured during the original prefill and evaluate attention at virtual positions corresponding to later copies of context blocks. We apply the model's existing Q/K projections and normalization to these states, then use RoPE (Su et al., 2024) to rotate the resulting query and key vectors according to their assigned virtual positions. Attention to the original cached keys then provides importance estimates without rerunning the context through the full transformer.

Our contributions are:

1. **Single-prefill approximation of reconstruction importance.** We approximate reconstruction-based KV importance using virtual copies of context blocks built from captured prefill attention-input states. Scoring requires additional attention computation within the same prefill, without an online reconstruction pass or a trained importance predictor.
2. **Reference-guided budget allocation and selection.** To allocate a shared KV budget across heads with different score scales, we map value-weighted Base ranks to values from offline reconstruction-based reference curves. Lookback Reranking (LR) then refines token selection while preserving the allocated counts in every layer, head, and block.
3. **Quality and efficiency across models.** Across 12 tasks and 6 models spanning dense, MoE, and hybrid-attention configurations, RECAP-KV achieves competitive mean task scores at 15% and 30% KV retention, with 37–46% less prefill-and-scoring time at 128K compared with direct reconstruction. Ablations on two models show benefits from both budget allocation and reranking.

2 Related Work

KV cache compression and reusable-context eviction. KV cache compression methods differ in how they represent cached states, select KV pairs for retention, and allocate retention budgets. Palu (Chang et al., 2025) and xKV (Chang et al., 2026) use low-rank representations, and Ada-KV (Feng et al., 2025) adaptively allocates the eviction budget across attention heads. H_2O (Zhang et al., 2023) and Scissorhands (Liu et al., 2023) exploit accumulated attention and persistent token importance, respectively, while Keyformer (Adnan et al., 2024) uses regularized attention-based scores. FastGen (Ge et al., 2024) selects head-specific policies through attention profiling, SnapKV (Li et al., 2024) scores tokens using a prompt-end observation window, and VATP (Guo et al., 2024) incorporates value norms into attention-based importance. Other methods use position or context statistics: StreamingLLM (Xiao et al., 2024) preserves initial attention sinks and a recent-token window for streaming inference, Compactor (Chari & Van Durme, 2025) combines approximate key leverage scores with noncausal attention within the context, and NestedKV (Chen et al., 2026a) scores key distinctiveness using global, block-level, and sliding-window key anchors. For reusable contexts, compression precedes downstream requests, and observed attention may not reveal tokens that become relevant to a later query. For budget allocation, RECAP-KV maps each head's ranking onto per-model reference curves, so that a single pooled threshold divides the budget across layers, heads, and context blocks.

Estimating future attention. Expected Attention (Devoto et al., 2025) estimates future attention under a modeled query distribution and combines the resulting scores with the norms of output-projected value vectors. Lookahead Q-Cache (Wang et al., 2025) obtains pseudo queries through brief lookahead decoding with the target model. SpecKV (Galim et al., 2026) instead generates lookahead tokens with a smaller draft model and processes them with the target model to estimate KV importance. DapQ (Tian et al., 2026) avoids autoregressive lookahead by processing synthetic tokens at future decoding positions to obtain pseudo queries. Unlike these future-attention estimators,

RECAP-KV approximates the attention of context reconstruction rather than of future decoding, using the context's own prefill states at virtual positions instead of additional tokens.

Reconstruction-derived scoring. KVzip (Kim et al., 2025) scores each KV pair by the maximum attention it receives during context reconstruction, which requires an additional transformer pass over the context. HYBRIDKV (Baek & Kong, 2026) selects heads offline and discards the other heads' KV caches, using reconstruction-based token pruning within retained heads only when further compression is needed. ContrastKV (Chen et al., 2026b) fuses attention scores from context reconstruction and random-string probes. KVgrad (Kwak et al., 2026) augments reconstruction scoring with gradient-based sensitivity and also provides a learned MLP surrogate. Fast KVzip (Kim et al., 2026) trains lightweight sink-attention gates to predict KVzip reconstruction scores from hidden states, while KVzap (Jégou & Jeblick, 2026) trains linear or MLP predictors on value-aware KVzip+ scores. These learned approximations reduce online reconstruction cost but require separate training and model-specific predictor weights.

3 Proposed Method

3.1 RECAP-KV Scoring

RECAP-KV estimates KV importance with a virtual copy of a context block: it constructs query and key vectors from hidden states captured during the original prefill and applies RoPE at positions corresponding to a later occurrence of the block. These query vectors assign attention weights to the original cached keys. For each key in that block, the maximum weight received across the virtual queries becomes the importance score of its KV pair. Taking the maximum allows a pair to receive a high score when even one query attends strongly to its key, following reconstruction-based KV importance scoring (Kim et al., 2025). Because the scoring queries come from the context itself, this criterion does not require a future task query.

Virtual-copy construction. We divide the C-token context into blocks $\mathcal{B}_b = [x_b, e_b)$ of at most B tokens, where $x_b = bB$, $e_b = \min((b+1)B, C)$, and $n_b = e_b - x_b$. The preserved system-prompt prefix has length σ, so context token j has position $\sigma + j$. For layer l and KV head h, let $(k_{l,h,j}, v_{l,h,j})$ denote the original cached pair. Under grouped-query attention (GQA) (Ainslie et al., 2023), the query heads in $\mathcal{Q}(h)$ share this KV head. We call the hidden states supplied to the model's Q/K/V projections attention-input states. For token i in block b, $0 \le i < n_b$, its state $\mathbf{h}_{l,x_b+i}$ gives a query vector and a key vector for the virtual copy at position p:

$$q_{l,h,g,i}(p) = R_p\, \nu_q(W_q \mathbf{h}_{l,x_b+i})\,, \qquad k^{\mathrm{virt}}_{l,h,i}(p) = R_p\, \nu_k(W_k \mathbf{h}_{l,x_b+i})\,, \tag{1}$$

Here, W_q and W_k are the projections for query head $g \in \mathcal{Q}(h)$ and KV head h; ν_q and ν_k are the model's Q/K normalization functions, or the identity when absent; and R_p is the model's RoPE rotation at position p. The virtual queries q and keys k^{virt} are used only for scoring; k^{virt} is not inserted into the original KV cache. The original cached keys retain their positions $\sigma + j$. Fig. 1 shows how the captured states and original cache supply the inputs for scoring during the original prefill. Sections 3.1–3.2 follow steps ①–⑥ of Fig. 1 in order.

Two scoring views. The Base score is used to distribute each head's KV budget across context blocks (Section 3.2), while Lookback Reranking uses the reranking score to refine selection within the eligible earlier blocks defined in Section 3.2, preserving their allocated retained counts. We compute these scores using different virtual positions and different original context ranges in the attention softmax. For virtual query i, both views include the preserved system-prompt keys and virtual keys $0, \ldots, i$ in the softmax normalization set.

(1) Base scoring. The query vectors and virtual keys are placed immediately after their source block, at $\sigma + e_b + i$. The original context keys included in the softmax are those in the prefix $[0, e_b)$.

(2) Reranking. The query vectors and virtual keys are placed after the full context and a reconstruction prefix, at $\sigma + C + i$ plus the prefix length (Fig. 2A; the prefix length is specified in Appendix C.1). These are the block-copy positions of the reconstruction reference T, defined in the *Relation to reconstruction* paragraph below. We construct these vectors from captured prefill states without

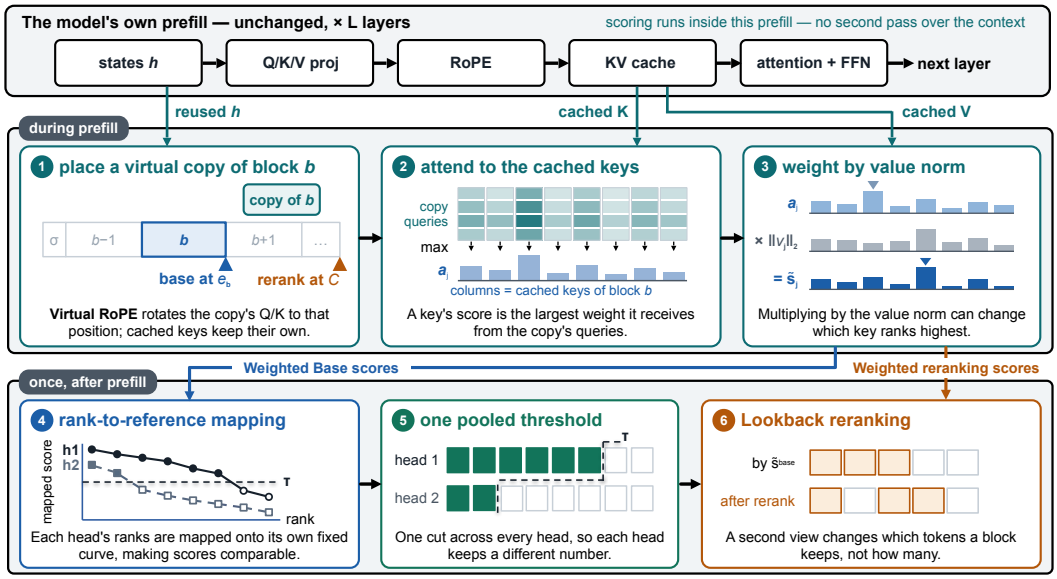

Figure 1: **RECAP-KV overview.** The original prefill supplies the KV cache and attention-input states reused for scoring. The Base and Reranking views derive queries and keys from these states and assign them their respective virtual RoPE positions; original cached keys retain their original positions. Per-key attention maxima (a_j in the schematic), weighted by value norms, estimate importance without an additional context-reconstruction transformer pass. Reference mapping makes scores comparable across heads, and a pooled threshold allocates the retention budget. Lookback Reranking changes the selected tokens while preserving each block's retained count. The resulting mask selects the original KV pairs retained in the reusable compressed cache. The diagram uses context-relative coordinates and omits the reconstruction-prefix length; the implemented virtual positions are specified in Appendix C.1.

processing the reconstruction-prefix tokens through the transformer. The scoring softmax excludes reconstruction-prefix keys and restricts the original context keys to the target block $[x_b, e_b)$.

Fig. 2A compares the virtual positions and original keys included in normalization. In both views, importance scores are assigned only to original keys in the target block. For scoring, the virtual keys affect only the causal softmax denominator, where virtual query i can attend to virtual keys $0, \ldots, i$.

From attention to importance. For each original key in the target block, we compute its attention weight and then take the maximum over the virtual query positions and the query heads sharing its KV head. Let d be the per-head dimension, $q^{\text{base}}_{l,h,g,i} = q_{l,h,g,i}(\sigma + e_b + i)$ the query at the Base position, and $\mathcal{K}^{\text{base}}_{l,h,b}(i)$ the key vectors included in its attention softmax (Table C1). For $j \in \mathcal{B}_b$,

$$a^{\text{base}}_{l,h,g,i,j} = \frac{\overbrace{\exp\left(q^{\text{base}}_{l,h,g,i} k^{\top}_{l,h,j}/\sqrt{d}\right)}^{\text{query–key match}}}{\underbrace{\sum_{\kappa \in \mathcal{K}^{\text{base}}_{l,h,b}(i)} \exp\left(q^{\text{base}}_{l,h,g,i} \kappa^{\top}/\sqrt{d}\right)}_{\text{normalization over keys}}}, \qquad s^{\text{base}}_{l,h,j} = \underbrace{\max_{\substack{g \in \mathcal{Q}(h) \\ 0 \le i < n_b}} a^{\text{base}}_{l,h,g,i,j}}_{\text{max over queries}}. \tag{2}$$

Here, a is an individual attention weight, and s is the importance score of the corresponding KV pair. The same expressions define a^{rerank} and s^{rerank} using the reranking positions and token ranges (Table C1).

Relation to reconstruction. An explicit reconstruction pass appends a reconstruction prefix and a copy of the target block to the full cached context and recomputes the copy's attention-input

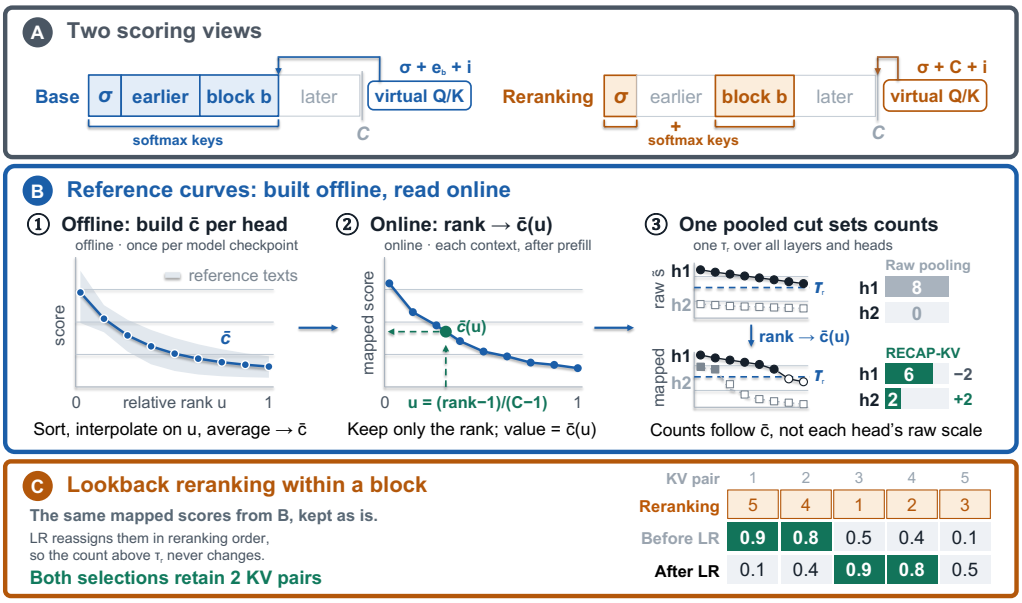

Figure 2: **Scoring views, reference-curve allocation, and Lookback Reranking.** (A) Base places virtual queries and keys immediately after the target block; Reranking places them after the full context. Normalization uses the preserved system-prompt prefix, causal virtual keys, and original context keys through the target block for Base or only within it for Reranking; only target-block keys receive scores. (B) Offline reference curves, averaged over reference texts on a shared relative-rank grid, map each token's value-weighted Base rank to a fixed value; one pooled threshold then keeps 6+2 instead of the 8+0 that raw-score pooling would keep. (C) Reassigning a block's mapped values by reranking order changes the selected tokens while retaining two pairs. Counts are schematic. Panel A omits the reconstruction-prefix length in the reranking coordinates.

states. We define the reconstruction reference T by taking the attention maximum over block-copy queries, with normalization restricted to the preserved system-prompt keys, original target-block keys, and causal block-copy keys. Reconstruction-prefix queries and keys are excluded from this reference score. Reranking differs from T only in using captured prefill states; Base also changes the positions and the normalization set. The reused states retain their original causal history; assigning virtual positions does not recompute them under the reconstruction context. Appendix C.1 measures the agreement of the Base and Reranking scores with T and with the score of the released KVzip implementation.

3.2 Budget Allocation and Token Selection

Given a cache budget, we determine how many KV pairs to retain in each head and context block, and which pairs to keep. Let L be the number of layers subject to eviction, H the number of KV heads per layer, and $r \in (0, 1]$ the requested retained fraction of the LHC context pairs.

Value weighting. Each attention head forms its output as a weighted sum of value vectors. The magnitude of an individual term depends on both its attention weight and its value norm. We therefore weight each Base score by the corresponding cached value's ℓ_2 norm, $\tilde{s}^{\text{base}}_{l,h,j} = s^{\text{base}}_{l,h,j} \|v_{l,h,j}\|_2$. The resulting score is used for ranking; the cached value vectors remain unchanged.

Rank-to-reference mapping. Weighted scores can have different scales across heads. We use a fixed reference curve $\bar{c}_{l,h}$ for each layer and KV head to define the scores used for cross-head selection. Each curve specifies the score assigned to each relative rank within that head. These curves are computed once per model and reused across contexts: the reconstruction-based importance scores of several reference texts are sorted within the head, interpolated onto a shared grid of relative ranks,

and averaged (Fig. 2B①). Sensitivity to the reference texts is evaluated in Appendix B.3. We assign a reference value to each token according to its value-weighted Base rank:

$$s_{l,h,j}^{\text{mapped}} = \bar{c}_{l,h}\left(\frac{\text{rank}_{\downarrow}(\widetilde{s}_{l,h,j}^{\text{base}}) - 1}{C-1}\right). \tag{3}$$

Within each layer and head, tokens are ranked in descending score order from 1 to C, with a fixed ordering for ties. The highest-ranked token receives $\bar{c}_{l,h}(0)$, and the lowest-ranked token receives $\bar{c}_{l,h}(1)$. For each head, the collection of mapped values is therefore fixed at a given context length C. The context-specific ranking determines which tokens receive those values.

Pooled threshold. We pool the mapped scores across all layers, KV heads, and context tokens and define τ_r as the $\lfloor rLHC \rfloor$-th largest score. For fixed C and r, the reference curves determine each head's retained count, so RECAP-KV combines an offline per-head budget with an online, context-specific ranking. The value-weighted Base ranking determines the initial token selection and, through the tokens' locations, how that count is distributed across blocks. Fig. 2B③ shows why the threshold is applied to mapped rather than raw scores: pooling raw weighted scores would allocate the budget by each head's score scale, whereas mapped scores are comparable across heads. The retained count in block b is

$$N_{l,h,b} = \left|\{\, j \in \mathcal{B}_b : s_{l,h,j}^{\text{mapped}} > \tau_r \,\}\right|. \tag{4}$$

The strict inequality retains fewer than $\lfloor rLHC \rfloor$ pairs, with ties potentially reducing the count further, so the requested budget is an upper bound.

Lookback Reranking. We use the reranking score to refine token selection while preserving these blockwise retained counts (Fig. 2C). The evaluated configuration applies this refinement to blocks lying entirely within the first $3C/4$ context tokens (Appendix C.1). We weight the reranking score by the same value norm, $\widetilde{s}_{l,h,j}^{\text{rerank}} = s_{l,h,j}^{\text{rerank}} \|v_{l,h,j}\|_2$, and use its ordering to reassign the existing mapped values. Within each eligible block, the largest mapped value is assigned to the highest-ranked token under reranking, the second-largest value to the next token, and so on.

For a fixed layer, head, and eligible block, let j_k^{mapped} and j_k^{rerank} be the token indices at rank $k = 1, \ldots, n_b$ in descending mapped-score and value-weighted reranking order, respectively. The reassignment and final retention mask are

$$s_{l,h,j_k^{\text{rerank}}}^{\text{final}} = s_{l,h,j_k^{\text{mapped}}}^{\text{mapped}}, \qquad M_{l,h,j} = \mathbf{1}\left[s_{l,h,j}^{\text{final}} > \tau_r\right]. \tag{5}$$

Here, j indexes tokens and k indexes ranks. Reassignment permutes the block's existing mapped values, so the number above τ_r remains $N_{l,h,b}$, while the selected token identities can change. Blocks outside the eligible region keep $s^{\text{final}} = s^{\text{mapped}}$. Applying M to the original KV cache produces the reusable compressed cache.

3.3 Efficient Cache Construction

Layer-wise score computation. During the original context prefill, we temporarily retain the attention-input states of each layer. Once the layer's attention has updated its KV cache, we use these captured states and cached keys to compute Base scores followed by reranking scores for the available context blocks. For each view, we reapply the model's Q/K projections and normalization and perform the specified attention calculation. We store the resulting scores, release the captured input states, and continue with the remaining computations of the layer. The same transformer pass therefore supplies both scoring views, while captured states need not accumulate across layers. The KV cache and score tensors remain available for final selection.

Batched attention and maximum reduction. For each scoring calculation, we first compute and retain query-wise log-sum-exp (LSE) normalizers over the full key set specified for that view. On supported tensor shapes, a dedicated LSE-only kernel preserves the FlashAttention-2 (Dao, 2024) normalizer computation while omitting the unused value reads and probability–value product; other shapes use the original kernel. We then recompute QK products only for the target block's original keys and normalize them using the retained normalizers. In Base scoring, earlier-context keys contribute to normalization, but their QK products are not recomputed for target attention. For each query-row tile, we compute target attention, update the running per-key maxima, and then release that tile's temporary attention weights. After all query rows and associated query heads have been processed, the maxima give the importance scores defined in Section 3.1, without storing the full attention matrix. For reranking, available blocks of equal length are evaluated as a batch, with each block normalized independently. Appendix C.2 gives operation counts, and Appendix C.3 compares this implementation with a dense implementation of the same scoring equations.

4 Experiments

4.1 Experimental Setup

We compare RECAP-KV with KVzip, StreamingLLM, query-agnostic SnapKV, Expected Attention, Compactor, and DapQ on 12 tasks and 6 models (Table 1). Each model is evaluated on 873 contexts, and we report task-equal mean scores on a 0–100 scale. Eviction precedes downstream queries. RECAP-KV uses 2,000-token scoring blocks; Appendix B.2 varies the block size and reranking range, and Appendix B.3 details how its reference curves are constructed. We report 15% and 30% here[1], with additional results in Appendix B. Timing covers prefill and importance scoring, excluding pruning and generation. Each run uses a single B300 GPU, except Llama-4-Scout-17B-16E, which uses 4 for every method; we report medians of 5 runs after 2 warm-ups. RECAP-KV timing uses the implementation of Section 3.3. GPT-OSS-120B uses native MXFP4 experts, and the other models use BF16. Full settings are in Appendix A.

4.2 Experimental Results

Table 1 compares answer quality and prefill-and-scoring time across 6 models. At 128K, StreamingLLM, SnapKV, Expected Attention, Compactor, and DapQ add at most a third to the prefill time, but even the best of them trails RECAP-KV by 10.8–34.9 points in mean task score. KVzip is accurate but slow, taking 1.6–1.8$\times$ as long as RECAP-KV at 128K. RECAP-KV matches or exceeds KVzip's mean score on all 6 models at 30% retention; at 15%, it outperforms KVzip on 3 models and trails by at most 2.0 points on the others. A separate Qwen2.5 cache-ready measurement at 128K and 30% retention retains a 1.63$\times$ speedup, including selection and compaction (Appendix A.3).

Fig. 3 presents our long-context retrieval evaluation using Needle-in-a-Haystack (NIAH)[2], which tests whether a model can recover information inserted into a long context. All compared methods use the same context–query instances, decoding settings, and answer-scoring rule. Fig. 3(a) evaluates retrieval from contexts up to 384K tokens on Qwen2.5-7B-Instruct-1M at 15% KV retention. At 384K, RECAP-KV achieves higher NIAH accuracy than KVzip, showing that captured prefill states can support effective long-context cache selection without an additional reconstruction pass.

Fig. 3(b) shows that RECAP-KV's absolute savings in prefill-and-scoring time over KVzip increase with context length on Qwen2.5-7B-Instruct-1M. The advantage persists up to 1M tokens, indicating that reusing prefill computation remains beneficial as longer inputs make context processing more expensive.

Fig. 3(c) highlights the importance of cache selection at low retention rates. On Qwen3-30B-A3B-Instruct-2507, with results pooled over 8K–128K, RECAP-KV and KVzip achieve similarly high NIAH accuracy at larger budgets, but RECAP-KV remains more accurate at 7% and 5% retention.

[1] Retention ratios apply only to eviction-eligible context KV pairs in full-attention layers. Recurrent states and KV caches in other layers are retained.

[2] See Appendix A.1 for the protocol used in Fig. 3(a,c).

Table 1: Quality and prefill-and-scoring cost across 6 models.

Model	FullKV (100%)	Streaming LLM	SnapKV	Expected Attention	KVzip	Compactor	DapQ	**RECAP-KV**
A. 12-task mean score ↑, 30% KV retained (best compressed score in bold)								
Qwen2.5-7B-Instruct-1M	65.7	27.4	26.0	31.3	61.4	35.6	27.5	**62.8**
Llama-3.1-8B-Instruct	61.2	26.9	24.0	31.4	56.5	30.0	26.7	**58.8**
Qwen3-30B-A3B-Instruct-2507	73.2	31.9	32.6	34.5	**73.0**	41.5	36.0	**73.0**
Qwen3.8-27B	81.1	34.7	42.3	44.2	76.1	48.2	43.5	**77.0**
GPT-OSS-120B	54.8	24.7	21.0	21.3	53.9	27.1	21.0	**54.4**
Llama-4-Scout-17B-16E	62.1	38.6	38.7	38.2	61.6	48.2	42.6	**62.1**
B. 12-task mean score ↑, 15% KV retained (best compressed score in bold)								
Qwen2.5-7B-Instruct-1M	65.7	20.7	21.5	23.0	50.4	28.7	21.5	**54.4**
Llama-3.1-8B-Instruct	61.2	20.3	19.5	23.5	49.2	22.1	23.6	**51.9**
Qwen3-30B-A3B-Instruct-2507	73.2	25.5	25.2	24.8	**68.4**	33.1	29.0	68.0
Qwen3.8-27B	81.1	29.2	36.7	38.0	49.7	40.5	37.3	**51.3**
GPT-OSS-120B	54.8	20.5	16.9	17.5	**44.5**	23.3	17.7	42.5
Llama-4-Scout-17B-16E	62.1	34.9	35.4	33.2	**60.1**	43.4	39.2	59.3
C. Time at 128K (s) ↓								
Qwen2.5-7B-Instruct-1M	9.8	9.8	10.0	11.0	38.0	10.7	10.0	22.7
Llama-3.1-8B-Instruct	12.9	13.0	13.1	14.5	47.2	14.4	13.1	29.6
Qwen3-30B-A3B-Instruct-2507	20.2	20.5	21.0	25.4	71.1	26.9	20.6	44.3
Qwen3.8-27B	16.6	16.7	16.7	18.3	48.5	17.3	16.8	27.8
GPT-OSS-120B	11.5	11.5	11.8	12.9	45.4	13.2	11.7	24.7
Llama-4-Scout-17B-16E	41.8	41.8	41.9	42.5	92.0	42.5	42.0	49.9

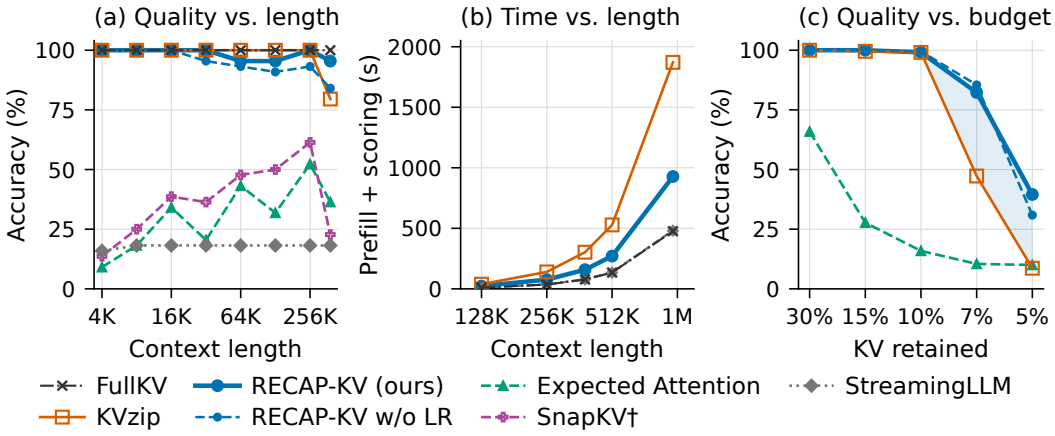

Figure 3: **Quality and cost across context lengths and KV budgets.** On Qwen2.5-7B-Instruct-1M, (a) shows NIAH accuracy at 15% retention and (b) shows prefill-and-scoring time. (c) shows NIAH accuracy versus retention on Qwen3-30B-A3B-Instruct-2507, pooled over 5 context lengths from 8K to 128K. † Query-agnostic SnapKV.

The gap at these budgets suggests that RECAP-KV preserves retrieval-relevant information more effectively under aggressive eviction.

4.3 Ablation Studies

Table 2A reports the change in mean score from enabling Lookback Reranking, overall and by task family. Because LR preserves the number of pairs retained in every layer, head, and block, the two runs differ only in which pairs they keep; LR still raises the overall mean by 1.04–2.99 points on Qwen2.5-7B-Instruct-1M and Llama-3.1-8B-Instruct. The gains therefore reflect ranking information from the Reranking view beyond the Base ordering. The largest gains shift from reasoning/aggregation at 30% retention to retrieval/copy at 15%, and some task families decline, so the gains from LR depend on the task type and retention level.

Table 2: Effects of Lookback Reranking (LR) and budget allocation on the Table 1 benchmark (12 tasks, 873 contexts per model). **A:** Score changes from enabling LR (on minus off) by task family and overall; both runs retain identical counts in every layer, head, and block. **B:** Absolute scores under different head/layer budget allocations at the same total KV budget (30% or 15%); changes in parentheses use unrounded scores.

A. Average LR gain

Model / KV kept	Retrieval/copy	QA	Reasoning/aggregation	Generation/ICL	Overall
Qwen2.5-7B-Instruct-1M, 30%	+1.74	+0.53	**+1.88**	+0.99	+1.35
Qwen2.5-7B-Instruct-1M, 15%	**+2.18**	+0.76	−0.01	+0.78	+1.04
Llama-3.1-8B-Instruct, 30%	+0.73	+3.96	**+4.47**	+3.82	+2.99
Llama-3.1-8B-Instruct, 15%	**+9.73**	−0.75	−1.87	−0.02	+2.58

B. Budget allocation

	Qwen2.5-7B-Instruct-1M		Llama-3.1-8B-Instruct	
Configuration	30%	15%	30%	15%
RECAP-KV	**62.8**	**54.4**	**58.8**	**51.9**
w/ equal per-head budgets	43.6 (−19.2)	32.8 (−21.5)	39.9 (−18.9)	28.6 (−23.3)
w/ equal per-layer budgets	59.8 (−3.0)	36.2 (−18.2)	60.0 (+1.3)	47.7 (−4.2)

Table 2B shows the contribution of RECAP-KV's reference-curve allocation. Replacing it with equal per-head budgets lowers the mean task score by 18.9–23.3 points on the two models, with scoring and reranking unchanged. RECAP-KV also outperforms equal per-layer budgets in 3 of the 4 settings; equal per-layer budgets score 1.3 points higher on Llama-3.1-8B-Instruct at 30% retention.

At 15% retention, Table 3 separates allocation from ranking using identical budgets in each layer and head within each row; Expected Attention and random ranking replace only the scoring rule. Reference-curve allocation raises Expected Attention from 23.0 to 44.6. With this allocation, RECAP-KV without LR outperforms random ranking and exceeds Expected Attention by 8.7 points, so allocation and scoring each contribute. LR improves both allocation settings while preserving blockwise retained counts. When LR instead reorders the eligible blocks by the Base score computed at the tokens' original positions, without virtual positions, the mean falls to 50.5, below RECAP-KV with and without LR. Appendix B.2 reports paired confidence intervals for these comparisons.

Table 3: Ranking and per-head budget on Qwen2.5-7B-Instruct-1M: 12-task mean score over 873 contexts at 15% retention.

Per-head budget	Random	Expected Attention	RECAP-KV w/o LR	**RECAP-KV**
Equal per head	–	23.0	31.9	32.8
RECAP-KV	37.3	44.6	53.3	54.4

5 Conclusion

We presented RECAP-KV, a query-agnostic KV eviction method that approximates reconstruction importance from the states a prefill already computes. A virtual copy of each context block, scored at virtual positions, turns these states into a selection signal for queries that are not yet known, at the cost of additional attention computation inside the prefill rather than a second pass over the context. The ablations isolate two further contributions: reference-curve allocation across layers and heads, which affects accuracy as much as the ranking rule itself, and Lookback Reranking, which improves the retained pairs without changing the allocated counts. Across 12 tasks and 6 models spanning dense, MoE, and hybrid-attention architectures, RECAP-KV attains the best or near-best mean task scores among query-agnostic methods at 15% and 30% retention while requiring substantially less prefill-and-scoring time than explicit reconstruction on long contexts, and the time saved grows with context length. A long context can thus be compressed during its only prefill into a cache that later queries reuse, and our vLLM prototype decodes directly from such caches. Since the preferred split across layers varies with the model and retention level, making this allocation adaptive is a natural next step.

AI USE STATEMENT

In this work, we used generative AI tools (OpenAI Codex and Anthropic Claude Code) to implement methods, design and provide feedback on experiments, interpret results, and assist with translation. We have not used generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, propose or refine hypotheses, or clean and reformat datasets, and the remaining required-disclosure tasks (mathematical claims, proofs, and qualitative data analysis) are not applicable to this work. Additionally, we used generative AI tools to draft and edit the paper, summarize related literature, format references, and create figure and analysis code. We have reviewed all AI-assisted work: the authors verified every reported number against the archived experiment outputs and reviewed the code used for the reported results. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Our code, submitted as supplementary material, includes an entry point for each evaluated model, covering scoring, cache construction, and the 12-task evaluation, together with the vLLM integration of Appendix D. Appendix A specifies the evaluation and timing protocol, and Appendix C gives the scoring definitions and the cache-construction algorithm.

REFERENCES

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J. Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: KV Cache Reduction through Key Tokens Selection for Efficient Generative Inference. In *Proceedings of Machine Learning and Systems*, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/48fecef47b19fe501d27d338b6d52582-Abstract-Conference.html.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://arxiv.org/abs/2305.13245.

Changwoo Baek and Kyeongbo Kong. HYBRIDKV: Exploiting Head-Dominant Reconstruction for Efficient Query-Agnostic KV Cache Compression. In *ICML 2026 Workshop on Resource-Adaptive Foundation Model Inference (AdaptFM)*, 2026. URL https://openreview.net/forum?id=sWF4OAXw7u.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S. Abdelfattah, and Kai-Chiang Wu. Palu: KV-Cache Compression with Low-Rank Projection. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2407.21118.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Hung-Yueh Chiang, Yash Akhauri, Xilai Dai, Huiqiang Jiang, Yucheng Li, Luis Ceze, Kai-Chiang Wu, and Mohamed S. Abdelfattah. xKV: Cross-Layer KV-Cache Compression via Aligned Singular Vector Extraction. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2503.18893.

Vivek Chari and Benjamin Van Durme. Compactor: Calibrated Query-Agnostic KV Cache Compression with Approximate Leverage Scores. *arXiv preprint arXiv:2507.08143*, 2025. URL https://arxiv.org/abs/2507.08143.

Hong Chen, Xiang Liu, Yubo Gao, Yuxuan Fan, Bo Wang, Yuanlin Chu, Yuanguo Lin, and Xuming Hu. NestedKV: Nested Memory Routing for Long-Context KV Cache Compression. *arXiv preprint arXiv:2605.26678*, 2026a. URL https://arxiv.org/abs/2605.26678.

Xingchi Chen, Peiyuan Zong, Ziqiang Gao, Qing Li, Yong Jiang, Fa Zhu, and Hui Li. ContrastKV: Robust KV Cache Eviction via Contrastive Signal Fusion for Multi-Query Generalization. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026b. URL https://doi.org/10.18653/v1/2026.acl-long.417.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.08691.

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. URL https://arxiv.org/abs/2510.00636.

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. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2407.11550.

Kevin Galim, Ethan Ewer, Wonjun Kang, Minjae Lee, Hyung Il Koo, and Kangwook Lee. Draft-based Approximate Inference for LLMs. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2506.08373.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model Tells You What to Discard: Adaptive KV Cache Compression for LLMs. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.01801.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Zhiyu Guo, Hidetaka Kamigaito, and Taro Watanabe. Attention Score is not All You Need for Token Importance Indicator in KV Cache Reduction: Value Also Matters. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024. URL https://arxiv.org/abs/2406.12335.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the Real Context Size of Your Long-Context Language Models? In *Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2404.06654.

Simon Jégou and Maximilian Jeblick. KVzap: Fast, Adaptive, and Faithful KV Cache Pruning. *arXiv preprint arXiv:2601.07891*, 2026. URL https://arxiv.org/abs/2601.07891.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.23416.

Jang-Hyun Kim, Dongyoon Han, and Sangdoo Yun. Fast KVzip: Efficient and Accurate LLM Inference with Gated KV Eviction. *arXiv preprint arXiv:2601.17668*, 2026. URL https://arxiv.org/abs/2601.17668.

Jihwan Kwak, Sunghwan Joo, Jung Yoon Hwang, Jaeseok Byun, and Taesup Moon. KVgrad: Query-Agnostic KV Cache Eviction via Gradient-based Global Importance Scoring. In *ICML 2026 Workshop on Resource-Adaptive Foundation Model Inference (AdaptFM)*, 2026. URL https://openreview.net/forum?id=cg1wTCJjjk.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023. URL https://arxiv.org/abs/2309.06180.

Yucheng Li, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir H. Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, and Lili Qiu. SCBench: A KV Cache-Centric Analysis of Long-Context Methods. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2412.10319.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM Knows What You are Looking for Before Generation. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2404.14469.

Jiawei Liu, Jia Le Tian, Vijay Daita, Yuxiang Wei, Yifeng Ding, Yuhan Katherine Wang, Jun Yang, and Lingming Zhang. RepoQA: Evaluating Long Context Code Understanding. *arXiv preprint arXiv:2406.06025*, 2024. URL https://arxiv.org/abs/2406.06025.

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. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.17118.

Meta AI. The Llama 4 Herd: The Beginning of a New Era of Natively Multimodal AI Innovation. Blog post, 2025. URL https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

OpenAI. gpt-oss-120b & gpt-oss-20b Model Card. *arXiv preprint arXiv:2508.10925*, 2025. URL https://arxiv.org/abs/2508.10925.

Qwen Team. Qwen3.8-Max: A New Bar for Coding and Cowork. Blog post, 2026. URL https://qwen.ai/blog?id=qwen3.8.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016. URL https://aclanthology.org/D16-1264/.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024. URL https://doi.org/10.1016/j.neucom.2023.127063.

Zhenxu Tian, Yi Su, Juntao Li, and Min Zhang. Where Matters More Than What: Decoding-aligned KV Cache Compression via Position-aware Pseudo Queries. *arXiv preprint arXiv:2603.11564*, 2026. URL https://arxiv.org/abs/2603.11564.

Yixuan Wang, Shiyu Ji, Yijun Liu, Yuzhuang Xu, Yang Xu, Qingfu Zhu, and Wanxiang Che. Lookahead Q-Cache: Achieving More Consistent KV Cache Eviction via Pseudo Query. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025. URL https://doi.org/10.18653/v1/2025.emnlp-main.1732.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient Streaming Language Models with Attention Sinks. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.17453.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025a. URL https://arxiv.org/abs/2505.09388.

An Yang, Bowen Yu, Chengyuan Li, et al. Qwen2.5-1M Technical Report. *arXiv preprint arXiv:2501.15383*, 2025b. URL https://arxiv.org/abs/2501.15383.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.14048.

A Evaluation Protocol

A.1 Tasks, Queries, and Scoring

Each example consists of a context and one or more queries about it. The cache is built and compressed from the context alone. Each query is then answered from the same compressed cache, and the tokens of one query and its answer are not carried over to the next.

The main comparison uses the 12 tasks in Table A1. The tasks follow the definitions of SCBench, RepoQA, SQuAD, and GSM8K (Li et al., 2025; Liu et al., 2024; Rajpurkar et al., 2016; Cobbe et al., 2021). For the main comparison, NIAH uses the generator released with KVzip (Kim et al., 2025), with Paul Graham essays, nominal context lengths of 500, 2,000, and 8,000 tokens, and 11 insertion depths from 0% to 100%, yielding 33 contexts.

All methods, including FullKV, use greedy decoding and the same reference answers and scoring rules. NIAH, key–value retrieval, prefix–suffix copying, and SQuAD use normalized answer inclusion, assigning one if the normalized reference answer occurs in the normalized response and zero otherwise. English QA uses the maximum of token-level F1 and answer inclusion, while English multiple choice applies inclusion scoring to the first response line. Normalization lowercases text, removes punctuation and articles, collapses whitespace, and converts number words zero–nine to digits.

Variable tracking uses the fraction of reference variables found in the response without normalization, and math finding uses exact string match. Summarization uses ROUGE-L F1, and RepoQA uses official pass@1 at a similarity threshold of 0.8. GSM8K checks whether the final reference answer occurs in the response after the last `the answer is` marker, or in the full response if the marker is absent. Many-shot learning uses exact match between parenthesized option labels when both the response and reference contain them; otherwise, it uses normalized inclusion of the reference answer text.

Scores are averaged over queries within each context, contexts within each task, and then equally across the 12 tasks. All scores are reported on a 0–100 scale.

Table A1: Main-comparison tasks, their families in Table 2A, and the number of contexts per model.

Code	Task	Family	Contexts
Ndl	Needle-in-a-Haystack (NIAH)	Retrieval/copy	33
KV	SCBench key–value retrieval (short)	Retrieval/copy	100
PS	SCBench prefix–suffix copying (short)	Retrieval/copy	100
RQA	SCBench RepoQA	Retrieval/copy	88
SQD	SQuAD	QA	100
QA	SCBench English QA	QA	20
Ch	SCBench English multiple choice	QA	18
GSM	GSM8K	Reasoning/aggregation	100
MF	SCBench math finding	Reasoning/aggregation	100
VT	SCBench variable tracking	Reasoning/aggregation	90
Sum	SCBench summarization	Generation/ICL	70
MS	SCBench many-shot learning	Generation/ICL	54

Long-context NIAH. For Fig. 3(a,c), we use the public NIAH implementation at `https://github.com/snu-mllab/KVzip/tree/main/data/needle`. The haystack concatenates 16 public-domain novels without repetition. We evaluate 11 equally spaced insertion depths from 0% to 100% and 4 seeds per context length, using up to 64 greedily generated tokens and normalized needle inclusion for scoring.

A.2 Models and Baselines

Table A2 lists the model configurations. Eviction is restricted to the listed layers, which are the full-attention layers on hybrid models. Retention fractions apply only to eligible context KV pairs; system-prompt KV pairs and KV pairs in other layers are preserved. Requested retention is an upper bound under the strict pooled threshold of Section 3.2; the realized fraction is the number of

retained eligible pairs divided by the number eligible before eviction. Across the 873 contexts of each evaluated model, the realized retention differs from the requested 15% or 30% by less than 0.1 percentage points.

For GPT-OSS-120B, all methods use the Harmony template with generation starting in the final channel. KVzip and RECAP-KV include the model's learned attention-sink logits in their scoring softmax. The evicted layers of Llama-4-Scout-17B-16E do not use RoPE, so virtual-position rotation is the identity, and their position-dependent query temperature is retained.

StreamingLLM, query-agnostic SnapKV, and Expected Attention use the scoring rules of kvpress[3] within the same evaluation code. StreamingLLM keeps 4 sink tokens and the most recent tokens without position re-rotation. SnapKV observes the last 64 context tokens without the query, with the window capped at one token less than the context length. Expected Attention uses `n_future`=512, 4 sink tokens, covariance, and value weighting.

For KVzip, we use the released reconstruction-scoring implementation with adaptations for the evaluated model architectures (Kim et al., 2025); Compactor uses its official scoring implementation. DapQ is implemented following the scoring procedure described in the original paper (Tian et al., 2026).

StreamingLLM, SnapKV, Expected Attention, and DapQ select KV pairs within each head. Compactor selects across the heads of each layer. KVzip and RECAP-KV pool scores across heads and layers, allowing nonuniform per-head budgets.

Table A2: Evaluated models, weight formats, and layers subject to KV eviction.

Model	KV eviction layers	Weights
Qwen2.5-7B-Instruct-1M (Yang et al., 2025b)	all 28	BF16
Llama-3.1-8B-Instruct (Grattafiori et al., 2024)	all 32	BF16
Qwen3-30B-A3B-Instruct-2507 (Yang et al., 2025a)	all 48	BF16
Qwen3.8-27B (Qwen Team, 2026)	16 full-attention of 64	BF16
GPT-OSS-120B (OpenAI, 2025)	18 full-attention of 36	MXFP4 experts
Llama-4-Scout-17B-16E (Meta AI, 2025)	12 full-attention of 48	BF16

A.3 Timing Protocol

We follow the hardware and repetition settings in Section 4.1. Prefill-and-scoring time starts with input token IDs on the GPU and ends when the context KV cache and final importance scores are ready; FullKV includes prefill only. We synchronize participating GPUs at both boundaries and rotate method order within each run. Model loading, tokenization, KV selection, cache compaction, and generation are excluded. For Table 1C, all methods of a model are timed on the same prefill, in 8,000-token pieces through the same model code, attention kernels, and GPU placement; Qwen3-30B-A3B-Instruct-2507 uses a grouped-GEMM MoE forward with the same weights and routing.

Dense and optimized implementations in Table C4 alternate within each setting and use the same prefill-and-scoring interval.

Time to a usable compressed cache. A separate measurement on Qwen2.5-7B-Instruct-1M at 30% requested retention times the full path from GPU-resident context token IDs to a compacted cache: prefill and scoring, reference mapping and LR reassignment (RECAP-KV only), KV selection, and physical compaction. On a single B300 GPU with no concurrent workload, the median of 5 runs after 2 warm-ups, timing the entire path as one interval, is 3.92/3.27 s at 32K, 11.61/8.18 s at 64K, and 38.76/23.81 s at 128K for KVzip/RECAP-KV. This run uses 8,000-token prefill pieces for RECAP-KV and 16,000-token chunks for KVzip, whereas Table 1C uses 8,000-token pieces for all methods. Model loading, tokenization, and generation are excluded; these measurements cover cache construction rather than end-to-end serving.

[3] `https://github.com/NVIDIA/kvpress`

B Additional Results

B.1 Task-Level Scores

Tables B1 and B2 break down the scores of Table 1 by task.

Table B1: Task-level scores (0–100). Overall is the equal-weight mean of the 12 tasks and reproduces Table 1. Task codes are defined in Table A1. †Query-agnostic SnapKV.

Method	KV retained	Ndl	KV	PS	RQA	SQD	QA	Ch	GSM	MF	VT	Sum	MS	Overall
Qwen2.5-7B-1M														
FullKV	100%	100.0	85.0	89.4	59.1	93.5	41.6	79.2	74.0	32.7	41.2	36.7	55.9	65.7
StreamingLLM	30%	27.3	22.5	23.6	15.7	31.4	21.4	56.5	4.0	24.2	12.9	29.8	59.3	27.4
SnapKV†	30%	15.2	1.2	4.0	9.1	32.4	34.4	68.1	6.0	29.7	24.1	32.4	55.9	26.0
Expected Attention	30%	18.2	0.7	1.4	14.1	51.0	36.5	76.4	35.0	31.5	21.7	33.4	56.3	31.3
Compactor	30%	72.7	0.0	0.0	17.5	37.8	44.0	73.6	4.0	38.7	45.8	34.7	57.8	35.6
DapQ	30%	21.2	0.2	0.2	9.8	37.2	35.1	69.4	14.0	30.5	19.3	34.9	58.1	27.5
KVzip	30%	100.0	81.0	55.0	58.0	94.2	43.9	77.8	63.0	34.8	40.7	36.1	53.0	61.4
RECAP-KV	30%	100.0	83.0	60.2	56.6	93.4	43.0	77.8	69.0	34.3	46.6	36.1	53.3	62.8
StreamingLLM	15%	15.2	11.0	11.2	8.4	20.7	17.3	46.8	3.0	19.7	9.9	28.0	57.4	20.7
SnapKV†	15%	9.1	0.3	0.8	2.5	22.5	28.7	60.7	3.0	25.7	18.7	29.3	57.0	21.5
Expected Attention	15%	9.1	0.0	0.0	4.1	31.6	30.4	62.0	14.0	28.5	10.7	28.9	56.3	23.0
Compactor	15%	24.2	0.0	0.0	3.2	23.9	41.4	75.0	5.0	36.0	46.8	31.5	57.0	28.7
DapQ	15%	9.1	0.0	0.0	2.7	28.2	24.9	55.6	9.0	26.5	12.1	31.6	58.5	21.5
KVzip	15%	100.0	46.9	7.8	45.2	88.9	35.6	72.2	47.0	33.5	41.6	34.9	50.7	50.4
RECAP-KV	15%	100.0	63.1	34.0	42.3	83.4	40.0	75.0	47.0	34.5	47.0	35.0	51.1	54.4
Llama-3.1-8B														
FullKV	100%	100.0	91.3	93.4	41.8	92.6	46.9	75.0	67.0	15.3	27.9	39.0	43.7	61.2
StreamingLLM	30%	27.3	26.9	28.6	10.7	34.5	23.2	49.5	12.0	16.0	20.4	31.1	42.6	26.9
SnapKV†	30%	9.1	5.7	3.2	6.1	38.0	37.5	64.8	6.0	21.5	20.5	34.5	41.1	24.0
Expected Attention	30%	6.1	2.9	2.6	11.1	73.0	43.8	70.8	52.0	6.5	19.5	33.3	55.2	31.4
Compactor	30%	57.6	0.6	0.0	11.4	40.9	41.6	72.2	8.0	7.5	37.0	37.5	46.3	30.0
DapQ	30%	15.2	1.6	1.4	7.3	48.0	36.3	60.6	28.0	6.2	22.1	36.3	57.4	26.7
KVzip	30%	100.0	93.7	56.0	26.6	92.3	36.7	57.9	70.0	14.0	31.2	40.3	59.3	56.5
RECAP-KV	30%	100.0	94.8	91.8	27.3	92.4	39.0	60.2	67.0	14.7	25.6	38.6	53.7	58.8
StreamingLLM	15%	15.2	12.0	13.6	5.5	26.4	21.0	42.1	9.0	13.5	15.8	29.5	40.4	20.3
SnapKV†	15%	9.1	1.7	0.6	2.5	29.0	30.0	51.8	6.0	19.2	9.0	30.9	44.1	19.5
Expected Attention	15%	0.0	0.3	0.4	3.2	52.1	36.4	57.4	27.0	7.5	9.4	30.5	58.1	23.5
Compactor	15%	9.1	0.0	0.0	1.8	29.9	37.2	68.1	3.0	7.0	29.9	34.2	44.8	22.1
DapQ	15%	12.1	0.2	0.0	1.4	35.3	33.0	48.6	20.0	6.5	18.3	33.8	74.1	23.6
KVzip	15%	100.0	50.4	58.0	16.8	88.5	36.5	58.3	65.0	6.3	32.5	35.1	43.3	49.2
RECAP-KV	15%	100.0	88.7	78.6	21.6	88.8	33.1	61.1	50.0	10.8	8.2	37.3	44.8	51.9
Qwen3-30B-A3B														
FullKV	100%	100.0	97.0	87.6	68.4	93.8	52.3	87.5	82.0	43.7	75.9	36.0	54.1	73.2
StreamingLLM	30%	24.2	29.0	27.4	18.2	37.1	29.0	74.5	7.0	26.2	16.9	30.6	62.6	31.9
SnapKV†	30%	48.5	1.3	4.8	12.3	38.0	42.7	68.1	11.0	41.8	32.1	33.1	57.0	32.6
Expected Attention	30%	0.0	0.3	3.6	20.0	57.0	37.5	77.8	42.0	41.3	41.1	35.2	57.8	34.5
Compactor	30%	69.7	0.0	0.0	23.0	41.1	48.3	87.5	17.0	45.3	73.8	34.4	57.4	41.5
DapQ	30%	60.6	0.1	1.2	14.3	41.7	40.6	68.1	18.0	44.7	50.0	35.7	57.0	36.0
KVzip	30%	100.0	95.8	86.8	68.2	93.9	50.3	86.1	83.0	42.2	77.3	36.2	56.7	73.0
RECAP-KV	30%	100.0	97.6	87.2	67.3	93.5	48.9	84.7	83.0	45.8	75.3	36.4	56.3	73.0
StreamingLLM	15%	15.2	13.8	14.2	10.2	28.3	22.2	64.8	10.0	24.7	13.8	29.1	59.3	25.5
SnapKV†	15%	39.4	0.2	1.0	3.6	28.6	26.5	59.3	5.0	38.3	11.9	31.1	57.4	25.2
Expected Attention	15%	0.0	0.1	0.6	4.8	40.0	32.4	65.7	14.0	37.0	10.6	32.3	60.0	24.8
Compactor	15%	21.2	0.0	0.0	8.0	32.9	47.3	77.8	7.0	41.2	67.9	32.9	60.7	33.1
DapQ	15%	36.4	0.0	0.2	4.3	34.4	36.9	65.3	12.0	43.3	25.5	33.6	56.3	29.0
KVzip	15%	100.0	87.4	74.0	65.5	87.5	46.5	80.6	68.0	39.5	77.7	35.1	58.9	68.4
RECAP-KV	15%	93.9	89.9	78.4	59.1	85.4	45.4	80.6	74.0	44.7	71.2	34.4	58.9	68.0

Table B2: Task-level scores (0–100). Overall is the equal-weight mean of the 12 tasks and reproduces Table 1. Task codes are defined in Table A1. †Query-agnostic SnapKV.

Method	KV retained	Ndl	KV	PS	RQA	SQD	QA	Ch	GSM	MF	VT	Sum	MS	Overall
Qwen3.8-27B														
FullKV	100%	100.0	99.4	100.0	76.1	94.4	58.1	88.9	73.0	79.3	99.9	40.1	64.1	81.1
StreamingLLM	30%	27.3	29.4	29.8	20.2	29.8	34.3	56.5	14.0	61.5	12.8	29.7	71.5	34.7
SnapKV†	30%	12.1	11.4	13.6	21.8	31.8	51.2	72.2	14.0	75.8	99.7	37.4	65.9	42.3
Expected Attention	30%	0.0	0.7	1.2	30.4	45.8	53.1	77.8	38.0	79.8	99.5	39.5	64.4	44.2
Compactor	30%	54.5	0.0	0.0	44.8	37.0	57.4	86.1	17.0	76.8	99.6	38.0	66.7	48.2
DapQ	30%	9.1	3.1	6.6	25.2	35.1	56.7	79.2	28.0	76.5	99.8	38.1	64.4	43.5
KVzip	30%	93.9	93.9	100.0	74.8	86.2	54.2	72.2	61.0	71.3	99.8	39.6	65.6	76.1
RECAP-KV	30%	100.0	92.2	98.6	67.7	89.9	54.3	79.2	63.0	74.8	99.9	39.3	64.8	77.0
StreamingLLM	15%	18.2	14.0	15.0	10.9	19.4	30.7	56.9	11.0	61.0	8.6	28.4	75.9	29.2
SnapKV†	15%	12.1	2.5	4.6	8.6	20.3	46.4	68.1	11.0	71.3	99.0	33.8	63.0	36.7
Expected Attention	15%	0.0	0.0	0.0	12.5	28.8	46.3	75.0	17.0	76.8	95.7	38.2	65.9	38.0
Compactor	15%	15.2	0.0	0.0	18.2	23.1	55.0	80.6	15.0	74.5	99.1	36.0	68.9	40.5
DapQ	15%	9.1	0.1	1.4	7.5	24.7	48.8	66.7	18.0	74.3	98.7	36.1	62.6	37.3
KVzip	15%	3.0	51.1	83.0	66.6	40.7	26.0	59.3	15.0	51.5	99.8	36.3	64.4	49.7
RECAP-KV	15%	42.4	51.2	58.8	39.8	54.2	38.9	59.7	23.0	64.2	85.0	34.6	64.1	51.3
GPT-OSS-120B														
FullKV	100%	97.0	58.2	66.6	61.6	77.8	36.5	63.9	40.0	24.2	56.4	27.4	48.1	54.8
StreamingLLM	30%	27.3	27.9	12.2	15.9	29.2	24.2	57.9	2.0	11.5	2.9	24.9	60.0	24.7
SnapKV†	30%	18.2	1.2	1.0	11.4	30.0	26.7	58.3	7.0	22.7	4.1	25.6	45.6	21.0
Expected Attention	30%	6.1	0.3	0.4	10.9	35.0	30.1	54.2	18.0	20.3	9.2	26.0	44.8	21.3
Compactor	30%	9.1	0.0	0.0	19.8	30.0	37.0	62.5	9.0	23.7	55.3	27.1	51.5	27.1
DapQ	30%	12.1	0.3	0.4	8.2	27.1	27.5	54.2	6.0	25.0	12.4	26.5	52.2	21.0
KVzip	30%	90.9	83.1	41.0	60.2	77.2	34.5	61.1	46.0	21.8	58.1	28.1	45.2	53.9
RECAP-KV	30%	90.9	69.9	47.2	59.1	79.8	34.5	66.7	55.0	24.7	51.2	27.6	46.7	54.4
StreamingLLM	15%	18.2	13.0	6.6	8.2	21.4	24.7	52.3	6.0	7.7	1.8	24.1	61.9	20.5
SnapKV†	15%	9.1	0.1	0.6	3.4	19.5	28.4	49.5	3.0	18.0	1.2	24.9	44.8	16.9
Expected Attention	15%	0.0	0.0	0.0	3.9	24.8	23.4	52.3	14.0	17.8	4.4	25.0	44.8	17.5
Compactor	15%	9.1	0.0	0.0	5.2	20.2	35.3	61.1	4.0	22.3	43.5	26.1	53.0	23.3
DapQ	15%	9.1	0.0	0.0	2.0	22.4	22.3	47.2	8.0	24.2	2.0	25.2	50.4	17.7
KVzip	15%	63.6	50.5	15.2	53.9	67.0	33.4	68.1	30.0	20.8	60.6	27.7	43.3	44.5
RECAP-KV	15%	51.5	27.6	18.0	43.0	69.1	34.2	63.9	47.0	22.7	53.5	26.7	53.0	42.5
Llama-4-Scout														
FullKV	100%	100.0	81.3	55.6	64.5	93.5	48.7	20.8	83.0	52.5	45.0	40.1	60.4	62.1
StreamingLLM	30%	48.5	23.2	25.8	17.3	87.5	22.8	8.3	86.0	33.3	18.6	31.5	60.7	38.6
SnapKV†	30%	48.5	4.4	4.2	13.0	87.7	34.8	12.5	88.0	48.3	25.1	35.3	62.2	38.7
Expected Attention	30%	60.6	0.9	3.0	14.3	90.7	43.2	5.6	86.0	52.7	10.3	31.9	58.9	38.2
Compactor	30%	97.0	10.2	2.0	31.1	89.5	48.5	16.7	86.0	53.0	45.1	38.5	60.7	48.2
DapQ	30%	48.5	4.6	1.6	15.9	89.5	44.3	19.4	85.0	53.0	45.3	38.9	65.6	42.6
KVzip	30%	100.0	82.5	51.8	64.3	93.7	47.3	19.4	85.0	51.5	45.0	40.1	58.5	61.6
RECAP-KV	30%	100.0	78.4	52.6	64.5	93.1	45.7	28.2	86.0	53.2	44.8	40.2	58.1	62.1
StreamingLLM	15%	42.4	11.4	13.2	9.1	87.4	22.4	9.7	88.0	31.0	16.1	29.5	58.9	34.9
SnapKV†	15%	45.5	0.6	1.0	3.2	87.5	34.4	11.1	86.0	46.7	18.2	32.7	58.1	35.4
Expected Attention	15%	33.3	0.1	0.2	3.9	89.5	33.9	6.9	87.0	49.2	7.9	30.5	55.6	33.2
Compactor	15%	84.8	0.5	0.0	8.9	87.2	47.1	13.9	87.0	53.0	43.6	34.9	60.4	43.4
DapQ	15%	42.4	1.1	0.4	4.8	86.9	39.1	16.7	86.0	52.8	43.3	36.5	60.0	39.2
KVzip	15%	100.0	61.0	51.6	62.1	93.3	46.8	24.1	85.0	50.0	48.1	40.2	58.9	60.1
RECAP-KV	15%	100.0	62.8	36.4	62.1	93.4	46.6	29.2	85.0	55.0	44.4	39.0	57.8	59.3

The paired 95% interval of the difference between RECAP-KV and KVzip in the 12-task mean of Table 1 lies above zero on Qwen2.5-7B-Instruct-1M and Llama-3.1-8B-Instruct at both retention levels and includes zero on the other 4 models, using 10,000 context resamples within each task (seed 0).

Table B3 repeats 2 retrieval/copy tasks at longer contexts on Qwen2.5-7B-Instruct-1M, with 100 contexts per length and the same reference curves as in the main comparison; the main comparison uses the short versions of these two tasks, and the longer lengths use their medium- and full-length versions. At 15% retention, RECAP-KV exceeds KVzip at every length. At 30%, its observed scores are higher on key–value retrieval at every length and on prefix–suffix copying at 18.9K tokens, and lower on prefix–suffix copying at 56.3K and 112.4K tokens, by 1.4 and 6.8 points.

Table B3: Longer contexts on Qwen2.5-7B-Instruct-1M: task score (0–100) by median context length, 100 contexts per length. The shortest length of each task is the cohort of Table 1.

			30% retention		15% retention	
Task	Tokens	FullKV	KVzip	RECAP-KV	KVzip	RECAP-KV
Key–value retrieval	24.6K	85.0	81.0	83.0	46.9	63.1
	84.8K	68.6	66.2	67.5	31.2	52.5
	169.1K	67.0	68.6	73.8	21.2	56.2
Prefix–suffix copying	18.9K	89.4	55.0	60.2	7.8	34.0
	56.3K	69.2	42.4	41.0	2.8	23.8
	112.4K	50.6	23.0	16.2	0.8	10.2

B.2 Ranking Sources, Block Size, and Reranking Range

Ranking sources at fixed per-head budgets. We isolate the effect of token ranking from budget allocation on Qwen2.5-7B-Instruct-1M at 15% retention. Within each allocation setting of Table 3, rankings retain identical counts in every layer and head; the selected tokens and their distribution across context blocks may differ. Table B4 reports paired comparisons for the 12 tasks and 873 contexts using per-head budgets verified to match Base for all compressed variants. Table 3 uses LR in the RECAP-KV column, and Table B4 additionally includes an LR variant whose order comes from Base scores at the tokens' original positions.

Table B4: Ranking-source comparisons at identical RECAP-KV per-head budgets on Qwen2.5-7B-Instruct-1M at 15% retention. Scores are on a 0–100 scale; Δ is the row's score minus Base's, in points, with paired 95% confidence intervals. FullKV is an uncompressed reference.

Ranking rule	Score	Δ [95% CI]
FullKV (reference)	65.68	–
RECAP-KV without LR (Base)	53.33	0
Random ranking	37.35	$-15.98\,[-17.95, -14.07]$
Expected Attention ranking	44.58	$-8.74\,[-11.01, -6.56]$
RECAP-KV with LR	54.37	$+1.04\,[+0.37, +1.72]$
LR with original-position scores	50.54	$-2.79\,[-3.47, -2.13]$

Scores average queries within contexts, contexts within tasks, and tasks equally. Intervals are unadjusted 95% percentile intervals from 10,000 paired context-bootstrap resamples within each task (seed 0). Differences use unrounded scores. Random ranking uses seed 7,500,000.

Block size and reranking range. Table B5 varies the block size B and the fraction of the context eligible for Lookback Reranking on Qwen2.5-7B-Instruct-1M, with the reference curves and per-head budgets unchanged. With Lookback Reranking enabled, every setting stays within 0.4 points of the evaluated configuration at 30% retention and 0.7 points at 15%, and disabling it lowers the mean at every block size.

Table B5: Block size and reranking range on Qwen2.5-7B-Instruct-1M (12 tasks, 873 contexts). Δ is the change from the evaluated configuration in points, with paired 95% confidence intervals. GSM8K and SQuAD contexts fit in one block at every size.

	30% retention		15% retention	
Setting	Score	Δ [95% CI]	Score	Δ [95% CI]
Evaluated: $B = 2{,}000$, LR on the first 75%	62.78	–	54.37	–
LR off	61.44	$-1.35\,[-1.92, -0.80]$	53.33	$-1.04\,[-1.72, -0.37]$
LR on the first 50%	62.45	$-0.33\,[-0.67, +0.04]$	53.76	$-0.61\,[-1.18, -0.09]$
LR on all blocks	62.40	$-0.38\,[-1.07, +0.29]$	54.25	$-0.12\,[-1.15, +0.90]$
$B = 1{,}000$	62.87	$+0.08\,[-0.45, +0.61]$	54.57	$+0.20\,[-0.43, +0.80]$
$B = 1{,}000$, LR off	61.92	$-0.87\,[-1.40, -0.33]$	53.76	$-0.61\,[-1.50, +0.31]$
$B = 4{,}000$	63.16	$+0.38\,[-0.09, +0.88]$	54.57	$+0.20\,[-0.57, +0.95]$
$B = 4{,}000$, LR off	62.24	$-0.55\,[-1.16, +0.07]$	53.86	$-0.51\,[-1.30, +0.28]$

B.3 Reference Calibration and Sensitivity

For each reference text, we compute reconstruction-based importance scores, sort them in descending order within each layer and KV head, and interpolate them onto a common relative-rank grid. Averaging the curves across texts yields the fixed reference curves used in Section 3.2; we call these the default curves and use them for all reported results.

For Qwen2.5-7B-Instruct-1M, fitting the reference curves from 6 texts (10,339 tokens in total, at most 1,800 per text) takes a median of 1.08 seconds on one B300 GPU, excluding model loading.

As a check, we also fit the curves of Qwen2.5-7B-Instruct-1M from texts that share no complete evaluation context or contiguous sequence of 32 normalized words with any of the 873 evaluation contexts: prose from a SQuAD training article absent from evaluation, with the same synthetic key–value inputs (excerpts below). Compared with the default curves, the 12-task mean changes by -0.16 points at 30% retention [95% CI: -0.68, $+0.35$] and -0.01 at 15% [-0.87, $+0.85$].

Prose (SQuAD). The 2008 Sichuan earthquake or the Great Sichuan earthquake, measured at 8.0 Ms and 7.9 Mw, . . .

Synthetic key–value. `JSON data:`
`{"c8e04639-e752-411e-9f42-3a9018720192":`
`"d0542e96-3b94-4e59-a82a-9f5a3095aa56", ...}`

Ellipses mark omissions; line breaks are adjusted for display.

Table B6 reports a separate sensitivity study on 200 contexts, taking the first 50 each from SQuAD, key–value retrieval, prefix–suffix copying, and variable tracking. LR remains enabled, and only the reference texts change. The 5 disjoint corpora replace both the prose and synthetic key–value texts, with no overlap between their prose and any evaluation context.

The 5 disjoint corpora with the same composition change the mean by -0.42 to $+0.83$ points and preserve high selection overlap (Table B6). On the full cohort of 873 contexts, 2 of these corpora change the mean by -0.36 to -0.17 points at 30% and $+0.08$ to $+0.13$ at 15%. This supports limited sensitivity to the specific reference texts in this setting. Changing the number of prose texts has larger effects at 15% retention: 1 text lowers the mean by 1.96 points, while 10 texts raise it by 1.48 points. Prose-only references yield smaller score changes despite lower selection overlap, showing that different retained sets can achieve similar observed scores.

Table B6: Sensitivity to the reference texts (Qwen2.5-7B-Instruct-1M, 4 tasks, 200 contexts, LR on). The first row uses the default reference texts; other rows give changes relative to this row in points. Jaccard: overlap of the retained pairs with the first row's selection at 15%.

Reference texts	Prose + KV	30%	15%	Jaccard
Default	3 + 3	71.36	59.12	–
Same texts, refitted	3 + 3	−0.14	+0.33	0.998
Five disjoint corpora	3 + 3	−0.28 to +0.83	−0.42 to +0.53	0.960–0.976
One prose text	1 + 3	−0.15	−1.96	0.904
Ten prose texts	10 + 3	+0.29	+1.48	0.907
Prose only	1 / 3 / 10 + 0	+0.33 / +0.33 / +0.33	+0.29 / −0.10 / +0.68	0.845–0.855

B.4 Learned KV-Eviction Predictors

Fast KVzip (Kim et al., 2026) and KVzap (Jégou & Jeblick, 2026) learn the importance score itself. Both first collect teacher scores on the target model—Fast KVzip the KVzip reconstruction importance, KVzap the KVzip+ score obtained by feeding the context a second time—and then train a small predictor from hidden states to these scores: a low-rank gate on the query-projection input of every layer (4096→512 with 16 sink tokens on Qwen3-8B) for Fast KVzip, and a per-layer linear or MLP head (hidden size 512) for KVzap. At inference the predictor scores the cache during the single prefill, so the teacher is never recomputed. The upstream studies report Fast KVzip distillation within one H100 hour and KVzap training on 750–1,250-token Nemotron samples; our retraining settings are specified below. Released weights cover 7 Fast KVzip gates (Qwen2.5-7B/14B-Instruct-1M, Qwen3-4B/8B/8B-FP8/14B, Gemma-3-12B-it) and KVzap heads for 3 models (Llama-3.1-8B-Instruct, Qwen3-8B, Qwen3-32B; linear and MLP).

Learned predictors estimate KV importance during prefill without online reconstruction. We evaluate their sensitivity to context length, input distribution, model changes, and retention policy. Across the evaluated configurations, the predictors transfer in some settings and lose accuracy in others. We distinguish changes in score quality from the effects of threshold calibration and protected windows, and specify the training budget and adaptation allowed in each comparison.

B.4.1 Setup

Table B7 lists the released predictors for the evaluated models, and Table B8 what each method prepares before compressing a new model. For comparisons at the same requested retention budget we apply the pooled rule of Section 3.2 (one threshold over all layers and heads, as KVzip does) to all three methods. The methods' native policies—Fast KVzip's window (16K-token chunked prefill, the last 4,096 context tokens protected for contexts of at least 16,000 tokens and the last 2% below that, and the window shrunk to the budget when the budget is smaller) and KVzap's DMS threshold (a threshold τ calibrated on RULER-4K development contexts to hit the target fraction on average, with a 128-token window)—are evaluated separately with their realized fractions. The shared strict pooled rule provides an upper bound on retained pairs, rather than exact retention; threshold ties can reduce the count further. Retraining uses the upstream predictor implementations on Qwen3-8B, the only model with released weights for both methods (131K YaRN profile; its reference curves are fitted as in Appendix B.3), with the data and budgets specified here: FineWeb-Edu views of 2K or 32K tokens, 262,144 supervised positions per condition (one quarter of the upstream protocol), 5,000 steps (Fast KVzip) or 15 epochs (KVzap-MLP), and initialization seeds 0–2; the corpus study of Table B11 uses 8K views, 131,072 positions, and seed 0. Held-out fidelity on the corresponding training-data distribution is Spearman .70 (Fast KVzip) to .81 (KVzap-MLP) per head. Cohorts: the 12-task cohort of Table A1 (873 contexts); RULER-4K and RULER-16K (Hsieh et al., 2024) (52, 104, or 650 contexts); and multi-query retrieval scenarios (multi-key, two-hop, and code-lookup; 3 queries per context; 2K–120K tokens; 24 contexts per length; exact containment of the gold code). Confidence intervals are cluster bootstraps over document groups or RULER items (2,000 resamples).

Table B7: Released predictors for the evaluated models (upstream releases as of 2026-09-24) and what a model without one requires.

Model	Fast KVzip gate	KVzap head	Without a released predictor
Qwen2.5-7B-Instruct-1M	released	–	KVzap: teacher labels + head training
Llama-3.1-8B-Instruct	–	linear, MLP	Fast KVzip: teacher labels + gate training
Qwen3-30B-A3B-Instruct-2507	–	–	both methods
Qwen3.8-27B	–	–	both methods
GPT-OSS-120B	–	–	both methods
Llama-4-Scout-17B-16E	–	–	both methods
Qwen3-8B (this appendix only)	released (+ FP8 gate)	linear, MLP	–

Table B8: Preparation requirements for a new model. Times are estimates for Qwen3-8B on one B300 GPU from measured unit costs (teacher passes, optimizer steps, feature hashing); positions are supervised training positions.

	RECAP-KV	Fast KVzip	KVzap
Per-model artifact	reference curves (0.6 MB; Appendix B.3)	trained gate (378 MB, FP32)	trained head (4.7 MB linear; 303 MB MLP)
Offline use of reconstruction	fitting the reference curves	training labels from KVzip scores	training labels from KVzip+ scores (context fed twice)
Preparation, 262,144 positions	one pass per reference text (1.08 s in total for Qwen2.5-7B-Instruct-1M; Appendix B.3)	≈33 min; 72 GB of features	≈52 min; 72 GB
Preparation, 1M positions	same	≈47 min; 288 GB	≈3.5 h; 288 GB
Model variant	validate curves; refit if needed	validate gate and policy (a separate FP8 gate is released)	validate head and threshold; adapt if needed
Longest directly labeled context (131K window)	no training labels	$\leq$128K	≈65K (two-copy labeling)
Retention policy	pooled threshold; requested fraction is an upper bound	pooled rule here; native window policy	pooled rule here; native DMS threshold

B.4.2 Released predictors and evaluation at the training length

Released-predictor quality varies across the evaluated models and workloads (Table B9): on Qwen3-8B RULER the KVzap-linear head is within 2 points of KVzip, and the Fast KVzip gate is 1.7–2.1 points below KVzip; on the Llama-3.1-8B 12-task cohort the KVzap-linear head is 0.6–2.0 points below KVzip and the KVzap-MLP head within 1.5 points of it; on the Qwen2.5-7B-Instruct-1M 12-task cohort the Fast KVzip gate is 0.9–2.5 points below KVzip. For predictors retrained here, evaluation at the training length gives the following results: the 32K-trained KVzap-MLP head matches KVzip on 32K scenarios in all 3 seeds (94.9 ± 1.7 vs. 94.4), the 32K-trained Fast KVzip gate scores 88.9 ± 1.1, and the 2K-trained predictors reach 95.8 ± 1.1 and 100.0 at 2K (KVzip 100.0). The released KVzap-MLP head under the pooled rule also transfers to variants of its model: it loses 1.0 point on the W4-quantized model at 15%, and on the base model its loss (8.7–12.6 points) is somewhat larger than KVzip's (6.0–9.1). These predictors score prefill states with their learned modules and avoid the online reconstruction pass used by KVzip (Kim et al., 2026; Jégou & Jeblick, 2026).

Table B9: Released predictors under the pooled rule (contexts in parentheses). Qwen2.5-7B-Instruct-1M and Llama-3.1-8B-Instruct: RECAP-KV and KVzip from Table 1; the released predictors are evaluated on the same 873 contexts, and the KVzap column gives the linear / MLP heads. Qwen3-8B: RULER with 50 contexts per task; its KVzap head is the linear one. –: no released predictor for the model (Table B7).

Model	Cohort	KV retained	RECAP-KV	KVzip	KVzap (linear / MLP)	Fast KVzip
Qwen2.5-7B-Instruct-1M	12 tasks (873)	30%	62.8	61.4	–	58.9
		15%	54.4	50.4	–	49.5
Llama-3.1-8B-Instruct	12 tasks (873)	30%	58.8	56.5	54.5 / 55.0	–
		15%	51.9	49.2	48.6 / 49.6	–
Qwen3-8B	RULER-4K (650)	15%	92.0	90.7	92.5	89.0
	RULER-16K (650)	15%	84.5	86.7	86.1	84.6

B.4.3 Sensitivity to deployment changes

Contexts longer than the training length. In the evaluated Qwen3-8B retraining configurations, all 3 seeds lose accuracy at 120K relative to their training-length evaluations (Table B10). The 2K-trained predictors score 29.6 $\pm$ 2.9 (Fast KVzip) and 28.2 $\pm$ 2.6 (KVzap-MLP) at 120K, and the 32K-trained predictors 73.6 $\pm$ 4.1 and 59.7 $\pm$ 2.0, whereas KVzip scores 84.7 at that length. Held-out score fidelity of a 2K-trained head falls from Spearman .75–.79 on 2K contexts to .54–.62 on 32K contexts, and uniform per-layer budgets do not recover the observed downstream loss. For a given model, RECAP-KV reuses the same reference curves across the tested context lengths. With the 131K-window configuration used here, KVzap's two-copy labeling procedure limits directly labeled contexts to approximately 65K tokens. Evaluation at 120K tests transfer of predictors trained on shorter contexts. In the evaluated 32K-context training sweep, increasing supervision from 32K to 262K positions improves held-out score fidelity (Spearman .61→.70 for Fast KVzip and .75→.81 for KVzap-MLP). KVzap-MLP's downstream accuracy falls within KVzip's interval in 15 of 16 cells, while Fast KVzip scores below KVzip in 10 of 16 cells.

Table B10: Accuracy beyond the training length (Qwen3-8B, multi-query scenarios, 15% KV retained, 24 contexts per length; mean $\pm$ sd over 3 initialization seeds).

	2K	8K	32K	120K
Fast KVzip, trained on 2K	95.8 $\pm$ 1.1	94.9 $\pm$ 2.6	84.7 $\pm$ 3.0	29.6 $\pm$ 2.9
Fast KVzip, trained on 32K	99.1 $\pm$ 1.3	98.1 $\pm$ 0.7	88.9 $\pm$ 1.1	73.6 $\pm$ 4.1
KVzap-MLP, trained on 2K	100.0 $\pm$ 0.0	94.0 $\pm$ 2.4	87.5 $\pm$ 1.1	28.2 $\pm$ 2.6
KVzap-MLP, trained on 32K	98.1 $\pm$ 0.7	97.2 $\pm$ 1.1	94.9 $\pm$ 1.7	59.7 $\pm$ 2.0
KVzip (teacher)	100.0	97.2	94.4	84.7
RECAP-KV	97.2	100.0	84.7	87.5
FullKV	100.0	100.0	97.2	84.7

Training corpus. With the model, view length (8K), number of supervised positions, recipe and seed fixed, we retrained both predictors on 5 corpora—FineWeb-Edu prose, code, mathematics, translated multilingual QA, and synthetic key–value JSON—and evaluated each on RULER, the multi-query scenarios and SCBench key–value retrieval (Table B11). Held-out fidelity to the teacher is highest for the matching corpus in every one of the 10 (family, test corpus) columns, and the corpus that gives the best downstream accuracy differs by evaluation: prose on RULER-4K, mathematics on RULER-16K, code on the scenarios, and the translated corpus on key–value retrieval. Among the 4 natural-language corpora the choice moves the KVzap-MLP head's accuracy by up to 13 points; in the tested synthetic key–value condition, the KVzap-MLP head trained on key–value JSON scores 65.4 on RULER-4K, 41.3 on RULER-16K, and 0.0 on the 32K scenarios (the Fast KVzip gate, not shown in the table, shows the same pattern, with losses of 6–19 points on 5 of its 6 natural-text evaluations). These comparisons show training-corpus sensitivity for the tested data budget and evaluations; they do not establish a general failure of predictors on unseen domains.

Table B11: Training corpus of the KVzap-MLP head (Qwen3-8B, 8K views, 131,072 supervised positions per corpus, seed 0) on 4 evaluations at 15% KV retained (RULER-16K at 10%). RULER-4K: 104 contexts; RULER-16K: 52; scenarios at 32K: 24; SCBench key–value retrieval: 24 contexts with 10 queries each. Bold: best corpus in the column.

Training corpus	RULER-4K	RULER-16K (10%)	Scenarios 32K	Key–value retrieval
Prose (FineWeb-Edu)	**91.9**	82.8	91.7	37.9
Code	91.3	82.5	**95.8**	33.8
Mathematics	90.4	**85.6**	93.1	36.7
Translated QA	91.5	77.8	91.7	**46.7**
Synthetic key–value	65.4	41.3	0.0	40.8
KVzip (teacher)	90.4	81.7	94.4	41.7
RECAP-KV	92.2	83.6	84.7	50.0

Model variants. Model changes affect the predictors' scores and their policy calibration differently. On Qwen3-8B quantized to 4-bit weights (RTN, group 128; RULER-4K, 104 contexts), every Fast KVzip gate we tested—the released gate under either policy, the released FP8 gate, and our retrained gate—loses 9.3–10.9 points of its unquantized accuracy at 10% KV retained, whereas KVzip's score changes by less than 1.2 points. On the base model (before post-training), every method loses accuracy (FullKV by 6.3 points at 15%); the released KVzap-MLP head loses 8.7–12.6 points under the pooled rule but 17.2–18.2 points under its DMS threshold calibrated on the instruct model, which realizes fractions of .13 and .09 instead of .15 and .10. RECAP-KV's reference curves are refitted with one reference pass; on the W4 model the refit recovers 3.5 points at 10%. These comparisons are not at matched adaptation effort: RECAP-KV receives a reference refit, whereas the predictors and their policies are used as released.

Native budget policies at small budgets. The native policies control memory differently: KVzap uses a fixed threshold, while Fast KVzip reserves a protected window. KVzap's DMS threshold spreads the realized retention 2–8$\times$ across contexts for one τ (on RULER-16K at a 15% target, the 5th–95th percentiles are 0.023–0.175), and the mean drifts with length (a 5% target keeps 2.4% on RULER-16K). Fast KVzip's window policy protects the last 4,096 tokens of contexts of at least 16,000 tokens and, once the budget is smaller than that window, keeps only the last $r\,C$ tokens without using the gate; in the 12-task cohort (873 contexts, median 24,628 tokens) this applies to 254 contexts (29.1%) at 15% KV retained. Table B12 shows the consequence on RULER-16K: under the pooled rule the predictors descend together with KVzip below 15%, and the native policies score lower at every budget.

Table B12: Small budgets on Qwen3-8B RULER-16K (104 contexts). Realized fractions of the DMS rows in parentheses.

KV retained	15%	10%	5%
RECAP-KV	85.5	83.8	58.7
KVzip (teacher)	90.5	84.2	48.3
Fast KVzip gate, pooled	86.6	72.5	45.5
Fast KVzip gate, window policy	67.2	58.0	27.7
KVzap-MLP head, pooled	89.5	77.5	48.8
KVzap-MLP head, DMS threshold	86.1 (.13)	66.8 (.08)	7.7 (.02)

B.4.4 Where each design is preferable

The reusable-cache setting is query-agnostic by construction: the cache is compressed before its queries exist, and in serving the contexts themselves are often not known in advance. In the evaluated configurations, a learned predictor's accuracy depends on how closely a deployment matches its training length, data, model, and budget policy (Tables B10–B12); no single training corpus is best for every evaluation in Table B11, and a distant one is harmful. Such a match is difficult to validate for contexts that have not yet arrived. A released predictor remains attractive for a fixed workload on which its quality has been validated, since it adds only a small projection per layer during prefill and needs no new training. RECAP-KV instead derives importance from the context itself

through additional attention during prefill; its only offline input is a set of model-specific reference curves, fitted in about a second and changing the mean by less than 1 point across the reference texts tested (Appendix B.3). It can therefore be applied to models, lengths, and inputs for which no predictor has been trained or validated. Tight KV budgets provide another reason to consider RECAP-KV: at 15% retention on the 12-task cohort it scores above the released predictor available for Qwen2.5-7B-Instruct-1M (Fast KVzip) and for Llama-3.1-8B-Instruct (KVzap-MLP) under the same pooled policy (Table B9), and on Qwen3-8B RULER-16K it achieves higher accuracy than both learned predictors at 10% and 5% retention (Table B12).

C METHOD DETAIL

C.1 SCORING VIEWS AND THE RECONSTRUCTION REFERENCE

Let $\mathcal{A}(\mathbf{h}, \Gamma)$ denote the attention maximum obtained by constructing queries and virtual keys from states $\mathbf{h}$ under the positions and normalization sets Γ, with the original cached keys held fixed. The reconstruction reference and Reranking share Γ_{rerank}, so that $T = \mathcal{A}(\mathbf{h}^{\text{copy}}, \Gamma_{\text{rerank}})$, $s^{\text{rerank}} = \mathcal{A}(\mathbf{h}, \Gamma_{\text{rerank}})$, and $s^{\text{base}} = \mathcal{A}(\mathbf{h}, \Gamma_{\text{base}})$. Then

$$T - s^{\text{base}} = \underbrace{\mathcal{A}(\mathbf{h}^{\text{copy}}, \Gamma_{\text{rerank}}) - \mathcal{A}(\mathbf{h}, \Gamma_{\text{rerank}})}_{\text{states only}} + \underbrace{\mathcal{A}(\mathbf{h}, \Gamma_{\text{rerank}}) - \mathcal{A}(\mathbf{h}, \Gamma_{\text{base}})}_{\text{geometry only}}.$$

This decomposition is an identity; Table C2 measures how much each term changes the selected tokens.

The reference block copy starts at $\sigma + C$ plus the tokenized length of the reconstruction prefix, and its rows occupy consecutive positions. Reranking uses the same positions. For the first block, the prefix length is the tokenized length of the first repeat instruction plus the model-specific chat-template postfix length. For later blocks, it is the tokenized length of the later repeat instruction plus the previous block's last 8 tokens and the same postfix length, using the evaluated block size $B = 2{,}000$. Instruction strings are tokenized without adding special tokens. This prefix-length convention follows the released KVzip implementation (Kim et al., 2025). Online reranking uses only the prefix lengths and does not process the prefix tokens.

Table C1: Scoring settings. $C_+ = B\lfloor 3C/(4B) \rfloor$; at $C = 261{,}673$ and $B = 2{,}000$, the reranked blocks are the first 98 of 131.

	Reference T	Reranking	Base
Query and virtual-key states	recomputed, $\mathbf{h}^{\text{copy}}$	captured, $\mathbf{h}$	captured, $\mathbf{h}$
Position of block-copy row i	$\sigma + C + i$ plus prefix length	same as reference	$\sigma + e_b + i$
Original keys in the softmax	system prompt, $[x_b, e_b)$	system prompt, $[x_b, e_b)$	system prompt, $[0, e_b)$
Virtual keys for query i	$0, \ldots, i$	$0, \ldots, i$	$0, \ldots, i$
Blocks scored	all	all; used for $e_b \le C_+$	all

Measured agreement. Table C2 compares the scores directly on 144 contexts of Qwen2.5-7B-Instruct-1M, the first 12 of each task in Table A1, with all scores saved in FP32. s^{KVzip} denotes the score of the released KVzip implementation (Kim et al., 2025). Like T, it normalizes over the system-prompt keys, the original target-block keys, and causal block-copy keys; it differs from T by including the reconstruction-prefix rows among these queries and keys and by computing the softmax in BF16. T is the reference of the decomposition above because it differs from Reranking only in the states. Each comparison is made within one layer, KV head, and 2,000-token block at the number of pairs $k = N_{l,h,b}$ that the RECAP-KV mask keeps in that block, which Lookback Reranking preserves; entries average the cells with $k \ge 1$ and $n_b \ge 2$. Recall is the share of the reference's top-k tokens that the candidate's top-k contains, with reference ties credited; top-$\lceil k/4 \rceil$ kept is the same share for the reference's top $\lceil k/4 \rceil$ tokens; coverage is the reference-score mass of the candidate's top-k relative to that of the reference's top-k; Spearman correlation is computed over all tokens of the block. The within-block order of s^{final}, RECAP-KV's final score, follows $\widetilde{s}^{\text{rerank}}$ in the eligible blocks and $\widetilde{s}^{\text{base}}$ elsewhere. With the geometry held fixed, s^{rerank} recovers 0.659/0.529 of T's selection at 30%/15% retention; with the states held fixed, s^{base} recovers 0.510/0.411 of the selection of s^{rerank}. Against s^{KVzip}, recall is 0.623/0.486 for s^{rerank} and 0.476/0.350 for s^{base}.

On strict prefixes of 3 long inputs at 8K, 16K, 32K, 64K, and 128K tokens, the Spearman correlation between s^{rerank} and T in the first four blocks, which every length contains, is 0.745/0.735 at 8K and 0.737/0.728 at 128K; recall there is 0.674/0.527 and 0.713/0.570, against chance 0.297/0.164 and 0.352/0.197. Over all blocks, Spearman is 0.733/0.724 at 8K and 0.702/0.688 at 128K, and recall is 0.682/0.534 and 0.656/0.518.

Using T in place of s^{rerank} in Lookback Reranking, that is, computing the reranking score from the reconstruction states $\mathbf{h}^{\text{copy}}$ with every per-block count unchanged, changes the 12-task mean on the 144 contexts by $-0.38\,[-1.08, +0.29]$ points at 30% retention and $+0.17\,[-0.85, +1.22]$ at 15%.

None of the SQuAD, GSM8K, and needle-retrieval contexts in this cohort has an eligible block, so the two variants coincide there; over the other 9 tasks, the changes are $-0.51\,[-1.44, +0.40]$ and $+0.23\,[-1.10, +1.64]$ (paired intervals computed as in Table B4).

Table C2: Score agreement per layer, KV head, and 2,000-token block at the evaluated per-block count k (Qwen2.5-7B-Instruct-1M, 144 contexts, 30% / 15%); chance recall k/n_b is 0.301 / 0.165.

Reference ← candidate	Recall	Top-$\lceil k/4 \rceil$ kept	Coverage	Spearman
A. Terms of the identity				
States: $T \leftarrow s^{\text{rerank}}$	0.659 / 0.529	0.812 / 0.656	0.832 / 0.724	0.720 / 0.711
Geometry: $s^{\text{rerank}} \leftarrow s^{\text{base}}$	0.510 / 0.411	0.659 / 0.549	0.696 / 0.582	0.540 / 0.552
Both: $T \leftarrow s^{\text{base}}$	0.473 / 0.348	0.601 / 0.455	0.642 / 0.507	0.470 / 0.477
States, value-weighted: $T\|v\|_2 \leftarrow \widetilde{s}^{\text{rerank}}$	0.671 / 0.543	0.826 / 0.675	0.842 / 0.737	0.743 / 0.735
B. Against the released KVzip score				
$s^{\text{KVzip}} \leftarrow T$	0.854 / 0.806	0.901 / 0.854	0.934 / 0.907	0.947 / 0.948
$s^{\text{KVzip}} \leftarrow s^{\text{rerank}}$	0.623 / 0.486	0.771 / 0.606	0.805 / 0.688	0.693 / 0.684
$s^{\text{KVzip}} \leftarrow s^{\text{base}}$	0.476 / 0.350	0.603 / 0.453	0.644 / 0.506	0.468 / 0.474
$s^{\text{KVzip}} \leftarrow s^{\text{final}}$	0.581 / 0.450	0.727 / 0.568	0.762 / 0.643	0.629 / 0.625

C.2 Algorithm and Cost

Algorithm 1 summarizes cache construction, and Table C3 counts the additional query–key evaluations for scoring per layer and query head. For fixed B and σ, Base normalization shares the leading $C^2/2$ term of causal prefill, while the remaining scoring steps scale linearly in C.

Table C3: Query–key pairs of the scorer per layer and query head. At $C = 261{,}673$ and $B = 2{,}000$, the total is about 1.07 times the query–key pairs of one causal prefill.

Step	Query–key pairs	Growth in C
Base normalization over the prefix	about $C^2/2 + CB$	quadratic, as in causal prefill
Base maxima over block keys	at most CB	linear
Reranking	at most $C(\sigma + 2.5B)$	linear

C.3 Dense and Optimized Implementations

Table C4 shows that implementation choices are central to RECAP-KV's practical efficiency. The dense implementation evaluates the same Base and Reranking equations, but uses separate prefills and materializes full FP32 attention probabilities. Sharing prefill across the two views, releasing states after scoring, and batching and tiling the attention computation together provide a 2.34–5.16× speedup over the dense implementation across 32K–256K. On Qwen2.5-7B-Instruct-1M, peak GPU allocation at 256K falls from 142.2 to 29.5 GiB, showing that avoiding large intermediate tensors substantially reduces the memory cost of the scoring computation. The timing protocol is in Appendix A.3.

Table C4: Dense / optimized prefill-and-scoring time and peak GPU allocation for RECAP-KV.

	Qwen2.5-7B-Instruct-1M			Llama-3.1-8B-Instruct			Qwen3-30B-A3B-Instruct-2507		
Context	Time (s)	**Speedup**	Peak (GiB)	Time (s)	**Speedup**	Peak (GiB)	Time (s)	**Speedup**	Peak (GiB)
32K	9.7/3.1	**3.18**×	32.1/17.3	12.9/4.1	**3.18**×	37.5/20.5	34.6/14.8	**2.34**×	78.1/61.3
64K	30.9/7.9	**3.90**×	47.5/19.0	40.3/10.4	**3.88**×	57.1/24.5	93.3/31.0	**3.01**×	96.6/64.3
128K	109.9/23.3	**4.72**×	78.2/22.5	145.2/30.1	**4.82**×	96.3/32.4	259.3/58.0	**4.47**×	133.9/70.2
256K	385.5/76.0	**5.07**×	142.2/29.5	510.2/98.9†	**5.16**×	177.7/48.8	820.5/180.1†	**4.55**×	210.9/82.4

5-run medians, 2 warm-ups; speedup = dense/optimized time. † Beyond the configured window; cost only. Qwen3-30B-A3B-Instruct-2507 uses the Transformers MoE loop here.

Algorithm 1 RECAP-KV cache construction

Require: Context of C tokens, model f_θ, reference curves $\bar{c}$, retained fraction r, block size B
Ensure: Compact reusable cache $\mathcal{KV}_r$
$C_+ \leftarrow B\lfloor 3C/(4B) \rfloor$; initialize the native KV cache $\mathcal{KV}$ and the score tensors s^{base}, s^{rerank}
for each prefill piece $\mathcal{P}$ in causal order **do**
  Execute f_θ once on $\mathcal{P}$; append its K/V and capture its attention-input states $\mathbf{h}_\mathcal{P}$
  for each block $\mathcal{B}_b$ completed by $\mathcal{P}$ **do**
    $s^{\text{base}}[\mathcal{B}_b] \leftarrow$ the score of §3.1 under the Base view (LSE computation and tiled FP32 column maxima)
    $s^{\text{rerank}}[\mathcal{B}_b] \leftarrow$ the score of §3.1 under the Reranking view
  end for
  Release $\mathbf{h}_\mathcal{P}$
end for
Weight the scores: $\widetilde{s}^{\text{base}} \leftarrow s^{\text{base}} \odot \|v\|_2$, $\widetilde{s}^{\text{rerank}} \leftarrow s^{\text{rerank}} \odot \|v\|_2$, where $\|v\|_2$ holds the per-token value norms $\|v_{l,h,j}\|_2$
Map $\widetilde{s}^{\text{base}}$ to s^{mapped} using $\bar{c}$ (§3.2)
Initialize $s^{\text{final}} \leftarrow s^{\text{mapped}}$
for each layer l, head h, and block b with $e_b \leq C_+$ **do**
  Sort block token indices by s^{mapped} and $\widetilde{s}^{\text{rerank}}$ to obtain
  j_k^{mapped} and j_k^{rerank}, respectively (§3.2)
  $s^{\text{final}}_{l,h,j_k^{\text{rerank}}} \leftarrow s^{\text{mapped}}_{l,h,j_k^{\text{mapped}}}$ for $k = 1, \ldots, n_b$
end for
$\tau_r \leftarrow$ the $\lfloor rLHC \rfloor$-th largest entry of s^{final} (equal to that of s^{mapped}, since reranking permutes values within blocks); $M \leftarrow \mathbf{1}[s^{\text{final}} > \tau_r]$
return the preserved system-prompt prefix and context pairs selected by M, compacted as $\mathcal{KV}_r$

D INTEGRATION WITH VLLM

From token eviction to paged KV storage. KV eviction selects individual KV pairs, whereas paged serving engines allocate fixed-size blocks (Kwon et al., 2023). In the standard full-attention configuration used here, vLLM shares a block table and sequence length across a request's KV heads. Different retained lengths thus require additional metadata; scattered deletions also leave blocks allocated until their surviving entries are compacted. Padding heads or layers to the largest retained cache reduces memory savings, while uniform retention budgets can alter the selected pairs. These requirements apply to RECAP-KV because its retained counts vary across layers and heads.

Applying the retention mask. Our vLLM 0.27.1 prototype maintains a block table, length, and storage offset per request, layer, and KV head, with one cache group per layer. Each physical block provides H head-sized segments of $\beta = 16$ KV pairs. Algorithm 2 packs selected entries while preserving their order, values, and original RoPE positions. Scheduler and worker updates form one logical compaction operation; the storage unit β is distinct from the scoring block size B.

Algorithm 2 Applying RECAP-KV retention in vLLM (one request)

Require: Prefilled KV cache $\mathcal{KV}$, context mask $M \in \{0,1\}^{L\times H\times C}$, system-prefix length σ, storage block size β

$P_0 \leftarrow \sigma + C$ ▷ Original prefix length at compaction
for each layer l **do**
  For every head h, form ordered indices I_h from $[0, \sigma)$ and $\{\sigma + j : M_{l,h,j} = 1\}$
  $n_h \leftarrow |I_h|$, $m_h \leftarrow \lceil n_h/\beta \rceil$ for all h; $a_l \leftarrow \lceil (\sum_h m_h)/H \rceil$
  Gather all selected K/V entries of layer l into a temporary buffer
  Reserve a_l blocks in the layer's cache group; assign m_h segments to each head h
  Pack the buffered entries and build head-specific block tables $\mathrm{BT}_{l,h}$
  Set row lengths $n_{l,h} \leftarrow n_h$ and offsets $\delta_{l,h} \leftarrow P_0 - n_h$
  Commit the new mapping and return unused blocks to the allocator
end for
for each subsequent attention-layer call at layer l and original token position t **do**
  Append K/V at row position $t - \delta_{l,h}$ for each h, extending storage as needed
  Update lengths to $t + 1 - \delta_{l,h}$; retain the original RoPE position t
  Run the existing attention kernel per request–KV-head pair with its associated query heads, using $\mathrm{BT}_{l,h}$ and the updated row lengths
end for

Validation and scope. Table D1 evaluates Qwen2.5-7B-Instruct-1M (BF16, tensor-parallel size 1) on 260 RULER contexts (Hsieh et al., 2024): 13 tasks, 10 contexts per task at each nominal length (4K and 16K). Masks use the reference curves, scoring, and LR settings of Table 1. Our Hugging Face (HF) implementation computes masks in advance; the FlashAttention-based vLLM prototype prefills its own cache and applies them at the context boundary, with prefix caching disabled. Across 6 tested retention fractions and 2 runs per context (free generation and teacher forcing), 3,120 compaction checks found no violations in per-head retained counts, copied K/V values, or block accounting. The prototype realizes RECAP-KV's nonuniform retention in paged storage while preserving the selected K/V values. Under identical masks, HF and vLLM achieve similar mean task scores, and occupied KV blocks are close to the requested retention fractions (Table D1). This experiment validates cache compatibility and occupancy; scoring is external, so it does not measure end-to-end compression time. Our code also includes a second path that imports one compressed context once as a shared prefix and prefills only the question of each later request; this path is not evaluated here.

Table D1: Quality and KV occupancy on 260 RULER contexts. Scores are task–length-equal mean recall; Δ is vLLM minus HF at the same mask. RECAP-KV uses per-layer cache groups.

Setting	HF	vLLM	Δ [95% CI]	KV blocks (%)
FullKV (stock)	92.76	92.65	-0.12 [-0.27, $+0.04$]	100.00
RECAP-KV, 30%	92.06	92.15	$+0.08$ [-0.33, $+0.49$]	30.30
RECAP-KV, 15%	87.96	88.03	$+0.08$ [-1.04, $+1.19$]	15.35

KV blocks: ratio of summed occupied prefix blocks to pre-compaction FullKV, including system tokens and padding; query/generated KV excluded. This is not process-reserved memory. CIs: 10,000 paired bootstrap draws stratified by task and nominal length.