More Than a Quick Glance: Overcoming the Greedy Bias in KV-Cache Compression

Anonymous ACL submission

Abstract

While Large Language Models (LLMs) can theoretically support extensive context windows, deployment remains constrained by the quadratic growth of Key-Value (KV) cache memory and inference cost. Prevailing compression strategies reduce memory through attention-guided eviction, but rely on a flawed assumption: that current attention scores are a sufficient proxy for token utility. High-attention tokens cluster around the same semantic region; greedy Top-K eviction selects redundantly from that cluster and permanently discards structurally relevant bridging tokens. We present LASER-KV (Layer Accumulated Selection with Exact-LSH Recall), a hybrid framework combining exact attention scoring with MagicPIG LSH recall to recover tokens outside the exact spike distribution. A protection divisor n governs cache budget partitioning, and an append-only accumulate policy ensures token selections are permanent, preventing early pruning errors from compounding across blocks. On BABILong at long context lengths such as 128k, LASER-KV achieves 47% average accuracy while FINCH collapses to 0% and SnapKV degrades to 37%, with 1.99× inference speedup and 70% peak VRAM reduction relative to full attention.

1 Introduction

Large Language Models have demonstrated remarkable capability on long-context applications such as summarizing documents, answering questions over entire codebases, and performing multi-hop reasoning across long narratives. However, deployment remains constrained by a fundamental memory bottleneck: the KV cache grows linearly with sequence length, and at long contexts its footprint alone exceeds that of the model weights.

The dominant response has been selective cache eviction, retaining highly attended *heavy hitter* tokens while permanently discarding low-scoring ones. These strategies rely on a greedy criterion: attention scores reflect relevance to the *current* query and cannot predict utility to future queries. At 128k tokens, this compounds fatally as methods like FINCH collapses to 0% accuracy on BABILong.

We present **LASER-KV** (Layer Accumulated Selection with Exact-LSH Recall), a training-free framework to mitigate this limitation through:

1. **Accumulative Memory Policy:** An append-only policy controlled by the protection divisor (n) partitions the cache budget between global syntactic anchors, a local sliding window, and a stable long-term memory pool.

2. **Exact-LSH Selection Policy:** Within the long-term pool, exact multi-head attention scoring combines with LSH probabilities to recover tokens that lack high attention spikes.

On BABILong up to 128k tokens, LASER-KV outperforms all baselines by up to 10% while reducing peak VRAM by 70%.

2 Related Work

2.1 KV Cache Compression

Zhang et al. (2023) identify that a small fraction of tokens receive the majority of attention mass. H2O retains these alongside a recency window. SnapKV (Li et al., 2024) scores tokens using the instruction-segment query at prefill. PyramidKV (Cai et al., 2025) allocates larger budgets to lower layers and smaller to upper. All three inherit the greedy selection criterion. Thus, tokens lost to early eviction are unrecoverable at later queries. Quest (Tang et al., 2024) avoids static eviction by estimating page criticality per decoding step, but still scores against a single query. StreamingLLM (Xiao et al., 2024b) retains attention-sink tokens plus a recency window, discarding all non-recent, non-sink context.

FINCH (Corallo and Papotti, 2024) applies recursive compression over sequential chunks. Therefore early-evicted tokens are unavailable to later chunks, culminating in complete collapse at 128k (Table 2). LASER-KV adopts attention sinks for the syntactic partition and an append-only policy preventing re-eviction, isolating selection quality from progressive information loss.

2.2 LSH Approximation and External Memory

Reformer (Kitaev et al., 2020) reduces self-attention to $\mathcal{O}(n \log n)$ via hash-bucket attention but requires training from scratch. MagicPIG (Chen et al., 2024) applies LSH sampling to deployed decoder-only models, ranking tokens by hash collision probability and achieving up to $5\times$ throughput at 96k context, but offloads hash tables to CPU. InfLLM (Xiao et al., 2024a) extends StreamingLLM with block-level retrieval from a CPU-resident KV store at the cost of per-step CPU–GPU transfer latency. LASER-KV adopts MagicPIG's probabilistic scoring as the recall stage of Exact-LSH, keeping computation on-GPU and maintaining a compressed cache throughout generation with zero retrieval latency.

3 Methodology

LASER-KV processes context in sequential blocks during prefilling. Unlike prior methods which maintain a fixed-size cache summary after each block (Zhang et al., 2023; Li et al., 2024; Cai et al., 2025), LASER-KV employs an append-only policy therefore once a token enters the long-term memory pool, it is never evicted. This helps avoid the irreversible loss of historically significant tokens.

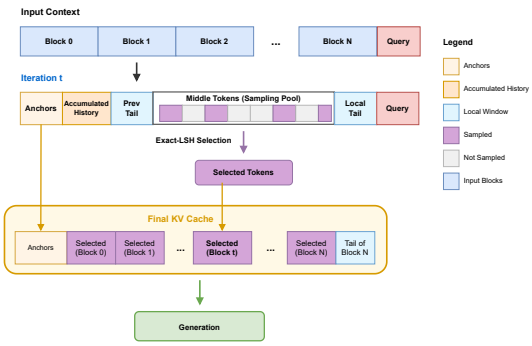

Figure 1: The LASER-KV Selection Pipeline. The input context is divided into blocks. Our Exact-LSH mechanism then selects tokens using the two criteria.

3.1 Budget Allocation via The Protection Divisor

The protection divisor n governs allocation between syntactic stability and long-term recall. Smaller n increases syntactic retention; larger n allocates more to long-term memory. We set $n = 4$, inducing a quarter-quarter-half split. Given total per-block budget B:

- **Syntactic Set ($2B/n$):** B/n tokens for Global Anchors preserving initial prompt tokens as attention sinks (Xiao et al., 2024b); B/n tokens for a Local Sliding Window maintaining grammatical coherence.
- **Recall Budget ($B - 2B/n$), denoted B_{long}:** Remaining capacity allocated to the Long-Term Memory Pool under Exact-LSH selection. Append-only constraint means selection quality directly determines recall fidelity.

3.2 Block Processing and Boundary Smoothing

Sequential fixed-size blocks handle arbitrarily long contexts within constant memory (Xiao et al., 2024a; Acharya et al., 2025). Block edges accumulate fewer attention scores, causing systematic under-scoring and premature pruning. We apply a look-back strategy (Figure 1) (Li et al., 2024): the scoring window for block $\mathcal{B}_t$ extends into the tail of $\mathcal{B}_{t-1}$, ensuring cross-block bridging tokens are evaluated with sufficient context before selection.

3.3 Exact-LSH Selection Algorithm

Single-metric selection produces brittle memory structures. Exact attention spikes preserve currently-attended tokens but miss structurally relevant tokens outside the local query distribution. We employ a hybrid Exact-LSH policy (Algorithm 1).

Precision (Global Consensus): Exact attention scores are summed across all L layers and H heads:

$$S_{\text{exact}}(k) = \sum_{l=1}^{L} \sum_{h=1}^{H} \text{Attn}_{l,h}(Q, k) \quad \forall k \in \mathcal{C} \quad (1)$$

Layer-head aggregation preserves tokens critical to specialized heads (Olsson et al., 2022). Top-αB_{long} tokens selected via a custom CUDA kernel fusing $Q \cdot K^\top$, softmax, and cross-layer accumulation in a single pass. Complexity: $\mathcal{O}(L_q \cdot |\mathcal{C}| \cdot d_h)$.

Recall (MagicPIG): Remaining $(1 - \alpha)B_{long}$ budget filled via LSH. Tokens ranked by hash colli-

sion probability with the query, recovering semantically relevant tokens excluded by the precision stage. Cost: $\mathcal{O}(|\mathcal{C}| \cdot R \cdot d_h)$ for R hash rounds.

This decomposition is essential under append-only accumulation. Exact attention alone clusters redundantly around the same semantic region (Zhang et al., 2023); LSH fills residual budget with structurally distinct tokens, improving coverage. Since selected tokens cannot be revisited, diversity at selection time directly determines downstream recall fidelity.

Algorithm 1 Exact-LSH Selection Policy

Require: Query Q, Candidate Set $\mathcal{C}$, Budget B_{long}, Ratio $\alpha \in [0, 1]$, Hash Functions $\{h_r\}_{r=1}^{R}$
Ensure: Selected Token Indices $\mathcal{K}$

$S_{exact}(k) \leftarrow \sum_{l=1}^{L} \sum_{h=1}^{H} \text{Attn}(Q, k) \quad \forall k \in \mathcal{C}$
$\mathcal{K}_{exact} \leftarrow \text{Top-K}(\mathcal{C}, \text{score} = S_{exact}, k = \alpha B_{long})$
$\mathcal{C}_{residual} \leftarrow \mathcal{C} \setminus \mathcal{K}_{exact}$
$S_{lsh}(k) \leftarrow \frac{1}{R} \sum_{r=1}^{R} \mathbb{1}[h_r(Q) = h_r(k)] \quad \forall k \in \mathcal{C}_{residual}$
$\mathcal{K}_{lsh} \leftarrow \text{Top-K}(\mathcal{C}_{residual}, \text{score} = S_{lsh}, k = (1 - \alpha) B_{long})$
return $\mathcal{K}_{exact} \cup \mathcal{K}_{lsh}$

4 Experiments

We evaluate LASER-KV on BABILong (Kuratov et al., 2024) across ten tasks and three context lengths. Section 4.1 describes setup. Section 4.2 reports main results. Section 4.3 analyzes computational overhead.

4.1 Experimental Setup

We evaluate on two models: Llama-3.1-8B-Instruct (Meta AI, 2024) for 16k and 64k, and Llama-3-8B-Instruct-Gradient-1048k (Gradient AI, 2024) for 128k, which extends context via frequency scaling similar to YaRN (Peng et al., 2023). We use 8B models to isolate selection policy contribution rather than model capacity. Baselines are Full Attention, FINCH (Corallo and Papotti, 2024), PyramidKV (Cai et al., 2025), and SnapKV (Li et al., 2024) at compression ratio r=0.25 at 16k and r=0.125 at 64k and 128k. BABILong task descriptions are in Appendix D; full implementation details in Appendix A.

4.2 Main Results

Table 1 summarises average accuracy across all ten tasks at each context length; Table 2 provides the 128k per-task breakdown.

Method	16k	64k	128k
Full Attention	52	39	38
FINCH	**48**	31	0
PyramidKV	42	35	38
SnapKV	41	27	37
LASER-KV (ours)	45	**40**	**47**

Table 1: Average accuracy (%) on BABILong QA1–QA10. Bold: best among compression methods per column.

Short to long context scaling. At 16k, FINCH leads at 48% with LASER-KV at 45%, consistent with greedy eviction remaining effective when evidence density is high. At 64k, LASER-KV leads at 40%, surpassing PyramidKV (35%), FINCH (31%), and SnapKV (27%). FINCH loses 17% as recursive eviction compounds across blocks, while SnapKV degrades as its observation window covers a smaller fraction of the context.

128k behaviour. At 128k, FINCH collapses to 0% across all tasks due to irreversible evictions. SnapKV and PyramidKV degrade to 37% and 38%, whereas LASER-KV maintains 47%, outperforming all baselines by at least 9%. This supports the hypothesis that append-only accumulation isolates selection quality from progressive information loss.

Task-level trends. QA5 saturates at 84% across all surviving methods, while the largest LASER-KV gains occur on tasks requiring aggregation of non-adjacent evidence such as QA7. The LSH recall stage recovers structurally relevant tokens outside the exact-attention spike region (Chen et al., 2024). QA2 and QA7 remain difficult across all methods, indicating persistent challenges in multi-hop retrieval and entity counting at 128k.

Figure 2 and Figure 3 report relative inference speedup and peak VRAM reduction.

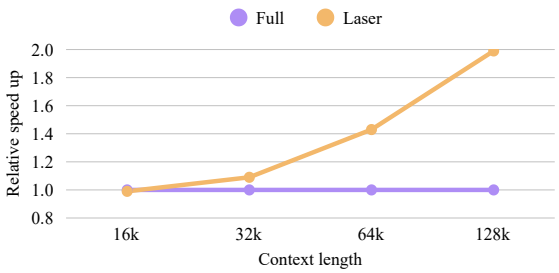

Figure 2: Relative speedup as context length increases.

Method	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Avg
Full Attention	31	6	15	46	80	55	15	14	69	53	38
FINCH	0	0	0	0	0	0	0	0	0	0	0
PyramidKV	25	**10**	19	45	84	59	1	15	67	54	38
SnapKV	22	5	19	45	84	57	2	18	68	52	37
LASER-KV (ours)	**38**	7	**25**	**57**	**84**	**66**	**30**	**30**	**74**	**58**	**47**

Table 2: Accuracy (%) on BABILong QA1–QA10 at 128k. FINCH collapses to 0% across all tasks. Bold: best among compression methods per task.

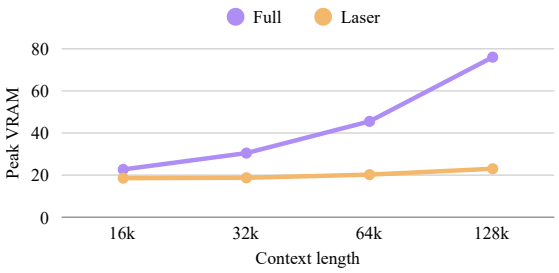

Figure 3: Peak VRAM as context length increases.

4.3 Efficiency Analysis

At 16k, LASER-KV is marginally slower (0.92×) as selection overhead is not yet amortized. Beyond 32k, Full Attention's quadratic scaling dominates, and LASER-KV reaches 1.99× speedup at 128k.

LASER-KV maintains a near-constant memory footprint because the accumulate policy caps cache size at B regardless of processed context (Kwon et al., 2023). At 128k, this reduces peak VRAM by 70%, enabling inference on a single 24 GB GPU, whereas Full Attention would require multi-GPU.

5 Ablation Studies

All ablations run on BABILong-16k with Llama-3.1-8B-Instruct (Meta AI, 2024), following the configuration in Appendix A unless specified. Full per-task breakdowns are in Appendix E.

5.1 Hybrid Ratio α

Figure 4 sweeps $\alpha \in \{1.0, 0.75, 0.50, 0.25\}$. All ratios achieve similar average accuracy (45–46%). Exact-only performs best on QA2 and QA8, where a single decisive token dominates, while lower α improves multi-hop tasks (QA3, QA4, QA9) by allocating more budget to LSH recall (Olsson et al., 2022). At longer contexts, exact attention would over-select semantically clustered tokens, whereas LSH improves diversity. We select α=0.75 for its stronger precision allocation at scale.

5.2 Protection Divisor n

Figure 5 sweeps $n \in \{3, 4, 8, 16\}$. Accuracy varies by at most 1 point, indicating robustness. Smaller n improves local reasoning, while larger n increases recall capacity. The n=4 split achieves the best average performance.

5.3 Boundary Smoothing

Figure 6 compares variants with and without boundary smoothing. Smoothing improves average accuracy by 2%, especially on tasks where evidence spans block boundaries (QA6, QA10, QA8, QA1), while adding no extra forward passes.

6 Conclusion

Attention scores are an insufficient proxy for token utility in long contexts: high-attention tokens cluster within the same semantic region, causing greedy Top-K eviction to discard structurally relevant tokens. LASER-KV addresses this through append-only accumulation and Exact-LSH selection, combining exact attention precision with LSH recall. On BABILong, LASER-KV maintains 47% accuracy at 128k while FINCH collapses to 0%, with 1.99× speedup and 70% VRAM reduction enabling inference within a single 24 GB GPU budget.

7 Limitations

Evaluation is limited to BABILong and two model scales; broader benchmarks are needed to establish generalisation. The protection divisor n and mixing ratio α are fixed globally, and adaptive layer-wise policies could improve performance. At contexts beyond 512k, a coarse LSH stage pruning $|\mathcal{C}|$ before exact scoring may better scale to million-token regimes. Extending the accumulate policy to streaming settings remains future work.

References

Shantanu Acharya, Fei Jia, and Boris Ginsburg. 2025. Star attention: Efficient llm inference over long sequences. *Preprint*, arXiv:2411.17116.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. 2025. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *Preprint*, arXiv:2406.02069.

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

Giulio Corallo and Paolo Papotti. 2024. Finch: Prompt-guided key-value cache compression. *Preprint*, arXiv:2408.00167.

Gradient AI. 2024. Llama-3-8b-instruct-gradient-1048k. https://huggingface.co/gradientai/Llama-3-8B-Instruct-Gradient-1048k. Accessed: 2026.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. 2024. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Preprint*, arXiv:2406.10149.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *Preprint*, arXiv:2309.06180.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: Llm knows what you are looking for before generation. *Preprint*, arXiv:2404.14469.

Meta AI. 2024. Llama 3.1: Advanced open-source language model. https://ai.meta.com/blog/meta-llama-3-1/. Accessed: 2026.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, and 7 others. 2022. In-context learning and induction heads. *Preprint*, arXiv:2209.11895.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *Preprint*, arXiv:2309.00071.

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

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. 2024a. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. *Preprint*, arXiv:2402.04617.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024b. Efficient streaming language models with attention sinks. *Preprint*, arXiv:2309.17453.

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. 2023. H_2o: Heavy-hitter oracle for efficient generative inference of large language models. *Preprint*, arXiv:2306.14048.

Appendix A Implementation and Configuration Details

Appendix A.1 Configuration Details

All experiments use bfloat16 precision with greedy decoding. LASER-KV is evaluated with block size 4096, protection divisor n=4, and mixing ratio α=0.75, inducing a $B/4/B/4/B/2$ cache partition across global anchors, local window, and long-term recall respectively. Compression ratio is r=0.25 at 16k and r=0.125 at 64k and 128k. FINCH uses block size 1024 per its original implementation; all other baselines operate without block decomposition. Models are Llama-3.1-8B-Instruct (Meta AI, 2024) at 16k and 64k, and Llama-3-8B-Instruct-Gradient-1048k (Gradient AI, 2024) at 128k.

Appendix A.2 Global Budget via Harmonic Mean

Standard block-wise compression fixes the per-block budget as a flat fraction of block size. This over-allocates budget at short context lengths (where the block covers a large fraction of total context) and under-allocates at long ones. To ensure fair, length-invariant comparison across the 16k–128k evaluation range, we compute the global memory budget B_{global} via the harmonic mean of block size S_{block} and total context length T, scaled by compression ratio r:

$$B_{\text{global}} = \left\lfloor \frac{2 \cdot r \cdot S_{\text{block}} \cdot T}{S_{\text{block}} + T} \right\rfloor \tag{2}$$

When $T \ll S_{\text{block}}$, the formula approaches $2rT$, scaling budget with context length and avoiding over-allocation. When $T \gg S_{\text{block}}$, it approaches $2rS_{\text{block}}$, capping budget at a fixed fraction of block size. B_{global} is evenly divided across all blocks; within each block, per-block budget B follows the protection divisor partitioning described in Section 3.1.

Appendix A.3 MagicPIG Configuration

Standalone MagicPIG variants score 19–21% below the hybrid, with little difference across configurations (26% vs. 24%), indicating that pure LSH is limited by retrieval quality rather than hash-table tuning. LSH captures structural similarity but cannot distinguish causally necessary tokens, a signal provided by exact attention (Zhang et al., 2023; Olsson et al., 2022). The 19% hybrid gain confirms the complementarity between precision and recall.

Appendix A.4 Absolute Position Indexing

A key correctness requirement in compressed KV cache inference is that tokens must retain their original absolute position indices for Rotary Position Embedding (RoPE) computation. When tokens are selected from non-contiguous positions and concatenated into a compressed cache, naive re-indexing from zero corrupts the positional encodings applied during key and query projection.

LASER-KV preserves original absolute positions throughout the pipeline. Each token in the compressed cache retains its position index from the full sequence at prefill time. During generation, query tokens are assigned positions starting from the total original context length, ensuring no positional overlap with prefill tokens. This guarantees that RoPE rotations are applied at semantically correct positions regardless of physical cache location.

Appendix A.5 Flash Attention Compatibility

Flash Attention 2 infers sequence structure from gaps in position ID sequences, treating non-contiguous positions as separate sequences in variable-length (varlen) mode. A compressed KV cache contains tokens drawn from disjoint positions, producing exactly such gaps. Without correction, Flash Attention would split the compressed cache into multiple independent subsequences, breaking cross-token attention.

We patch the relevant Flash Attention position preparation function to force single-sequence mode regardless of position ID contiguity. This ensures all tokens in the compressed cache participate in a single unified attention computation. The patch introduces no overhead and does not alter attention outputs beyond enforcing the correct sequence interpretation.

Appendix B Exact Attention Kernel Details

Appendix B.1 Fused Scoring Kernel

Standard exact attention scoring using framework-level attention hooks requires one forward pass per layer to extract attention weights, followed by post-hoc accumulation. This introduces three sources of overhead: repeated GPU memory allocation for full attention weight matrices of shape $[B, H, L_q, |\mathcal{C}|]$ across all layers, synchronization between the attention computation and the hook callback, and separate reduction passes to sum across layers and heads.

Our custom CUDA kernel eliminates all three overheads by fusing the following operations into a single GPU kernel launch per block:

1. **QK product:** Compute $\mathbf{Q}\mathbf{K}^\top/\sqrt{d_h}$ for query tokens against the candidate set $\mathcal{C}$ at each layer and head.

2. **Softmax:** Apply numerically stable softmax over the candidate dimension.

3. **Cross-layer accumulation:** Accumulate per-candidate attention mass across all L layers and H heads, producing a single importance score $S_{\text{exact}}(k)$ per candidate.

The fused kernel writes only the final $|\mathcal{C}|$-dimensional score vector, avoiding materialisation of full per-layer attention weight matrices. Theoretical FLOPs are $O(L \cdot H \cdot L_q \cdot |\mathcal{C}| \cdot d_h)$, identical to standard exact attention, but memory traffic is reduced by a factor of $L \cdot H$ because intermediate matrices are never written to global memory.

Appendix B.2 Query Reconstruction for Prefilled Layers

During prefill, query vectors are not retained in memory after the attention operation completes. To avoid a second forward pass solely to extract queries, we reconstruct query vectors post-hoc from captured hidden states. For each layer, pre-attention hidden states are intercepted via a registered forward hook. The query is then reconstructed as:

$$\hat{\mathbf{Q}} = \mathrm{RoPE}(\mathrm{reshape}(\mathbf{W}_Q\, \mathbf{h}_{\text{query}})\,,\ p_{\text{query}}) \quad (3)$$

where $\mathbf{W}_Q$ is the query projection matrix, $\mathbf{h}_{\text{query}}$ are the captured hidden states at query token positions, and p_{query} is the original absolute position index. Keys in the compressed cache already carry post-RoPE encodings from their original prefill positions, ensuring $\hat{\mathbf{Q}}$ and the cached keys are in a consistent positional space without re-running any attention layers.

Appendix C Limitations

Benchmark scope. All main results are reported on BABILong with two model scales. BABILong tasks are synthetic and use a controlled distractor format; gains may not transfer uniformly to open-ended or retrieval-augmented generation benchmarks. Evaluation on additional long-context benchmarks and needle-in-a-haystack variants would establish broader generalisability.

Model scale. Experiments use 8B-parameter models. Larger models (70B scale) exhibit stronger native long-context ability and may show different attention spike patterns. The relative contribution of the LSH recall stage could shift at larger scale, where exact attention spikes may already cover a broader portion of the relevant token distribution.

Fixed global hyperparameters. The protection divisor n and mixing ratio α are set as global constants across all layers. Attention patterns vary systematically by layer (Cai et al., 2025): lower layers encode syntactic structure with diffuse attention; upper layers concentrate on semantic content. Layer-adaptive n and α could improve over the uniform default, but would require a principled adaptation mechanism to avoid overfitting to a specific benchmark distribution.

Candidate pool growth at extreme lengths. The exact attention scoring stage has complexity $O(L_q \cdot |\mathcal{C}| \cdot d_h)$. At context lengths beyond 512k, the candidate pool $|\mathcal{C}|$ grows large enough that even a fused kernel imposes non-trivial per-block overhead. A hierarchical approach—coarse LSH filtering to prune $|\mathcal{C}|$ before the exact kernel—could extend the method to million-token regimes without sacrificing the precision guarantees of the exact stage.

Prefill-only compression. LASER-KV selects tokens during prefill under the accumulate policy. The framework does not currently support online streaming settings where context arrives incrementally and no future block is available when selection decisions must be made. Extending accumulate-mode selection to the streaming regime is a natural direction for agent and dialogue applications.

Training-free regime. LASER-KV is entirely training-free. Joint optimisation of the selection policy and model weights could yield a stronger precision-recall trade-off. However, training-based methods sacrifice zero-shot deployability across arbitrary models and task distributions, which is a primary design goal of the current framework.

Appendix D BABILong Benchmark Task Descriptions

To rigorously evaluate long-context retrieval and reasoning capabilities, we utilize the BABILong benchmark (Kuratov et al., 2024). This suite embeds synthetic reasoning chains inside long distractor text drawn from PG-19 books. Ten tasks test increasing reasoning depth and breadth. Table 3 provides detailed descriptions of all ten tasks, including the reasoning type required and representative query structures. Each task contains 100 samples per context length; accuracy is measured by substring match.

Appendix E Ablations

All ablations run on BABILong-16k with Llama-3.1-8B-Instruct (Meta AI, 2024), r=0.25, block size 4096, varying one component while fixing all others at defaults (α=0.75, n=4, boundary smoothing enabled).

Task ID	Task Name	Description & Reasoning Requirement
QA1	Single Supporting Fact	*Fact Retrieval.* The model must locate a single specific fact hidden within the context to answer the query. *Example:* "Where is Mary?" → "In the kitchen."
QA2	Two Supporting Facts	*Multi-hop Reasoning.* Requires chaining two separate pieces of information. The model must identify an intermediate entity to locate the final target. *Example:* "Where is the football?" (implied: Who had it last? → Where are they now?)
QA3	Three Supporting Facts	*Complex Chaining.* A higher-order multi-hop task requiring retrieval and synthesis of three distinct facts spread across the context to derive the correct conclusion.
QA4	Two Argument Relations	*Path Finding.* The model must determine a spatial or logical path between two entities by traversing a chain of pairwise relational statements (e.g., "A is north of B; B is east of C"). *Example:* "How do you get from the garden to the office?"
QA5	Three Argument Relations	*Complex Relations.* Requires reasoning over relational dependencies among three entities. The model must track and integrate multiple interrelated facts to resolve relative positions or interactions.
QA6	Yes/No Questions	*Verification.* The model must affirm or negate a statement based on the strict presence or absence of supporting facts in the context. Correct negation requires confirming the absence of contradicting evidence, not merely its presence.
QA7	Counting	*Entity Counting.* The model must count all instances of a given entity or event distributed across the full document. Every occurrence must be retrieved; missing even one instance produces an incorrect answer. This task is maximally sensitive to token eviction. *Example:* "How many objects is Daniel holding?"
QA8	Lists and Sets	*Set Tracking.* The model must maintain an accurate set of objects held by an entity across a sequence of acquisition and transfer events. State updates can appear anywhere in the context. *Example:* "What is Daniel holding?" → "{apple, football}"
QA9	Simple Negation	*Negation Handling.* The model must correctly interpret negative statements ("X never visited Y") and integrate them with positive evidence to answer queries. Requires distinguishing confirmed absence from unmentioned states.
QA10	Indefinite Knowledge	*Motivation and State Tracking.* The model must track the motivations or internal states of entities, which are often stated indirectly or updated by subsequent events. Requires integrating sparse, non-adjacent causal statements. *Example:* "Why did Sandra go to the office?"

Table 3: Detailed descriptions of all ten BABILong tasks. Tasks increase in reasoning complexity from single-fact retrieval (QA1) to multi-hop chaining (QA2–QA5), set tracking (QA8), and negation reasoning (QA9–QA10). Difficulty scales with evidence dispersion: tasks requiring aggregation of multiple non-adjacent tokens are disproportionately harmed by KV cache eviction.

Hybrid Ratio α. Figure 4 sweeps the long-term budget split between exact attention and MagicPIG LSH recall. All ratios yield near-identical average accuracy (45–46%); task-level differences motivate α=0.75 for its stronger precision allocation at longer contexts.

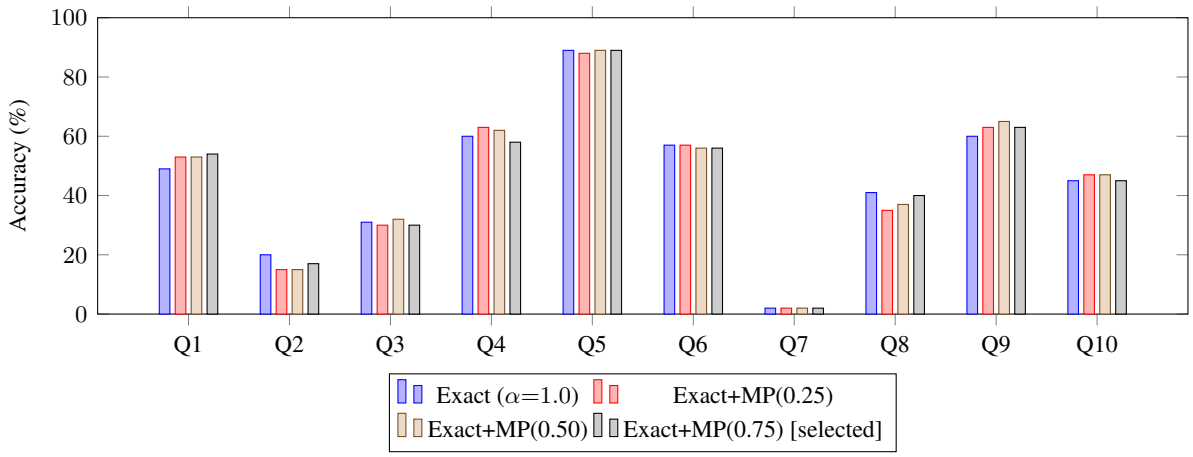

Figure 4: Effect of hybrid ratio α on BABILong-16k accuracy (%). Shaded bars: selected configuration.

Protection Divisor n. Figure 5 sweeps cache budget partitioning. Average accuracy varies by at most 1 point; n=4 achieves best average while balancing anchor stability and recall capacity.

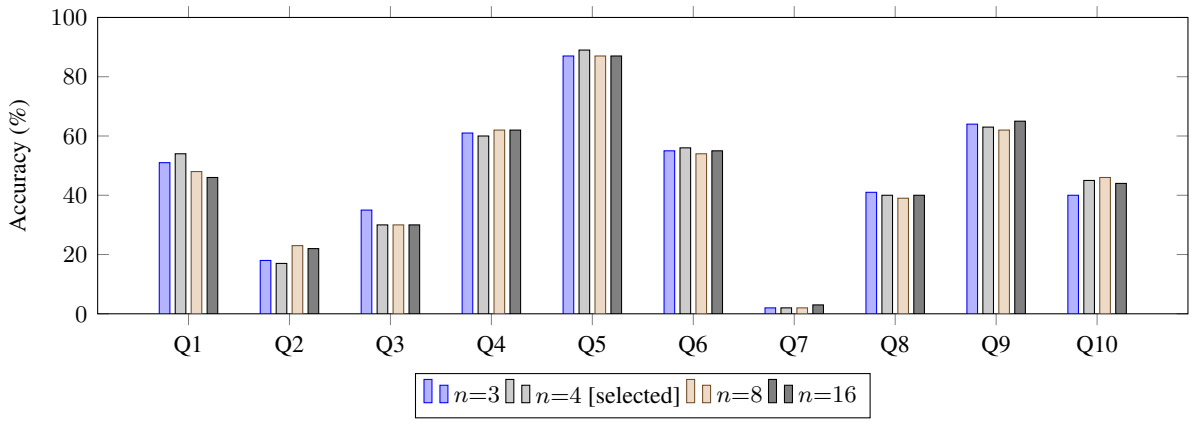

Figure 5: Effect of protection divisor n on BABILong-16k accuracy (%). α=0.75. Shaded bars: selected configuration.

Boundary Smoothing. Figure 6 compares variants with and without look-back boundary correction. Gains concentrate on tasks where facts straddle block boundaries: QA6 (+6), QA10 (+5), QA8 (+3), QA1 (+3).

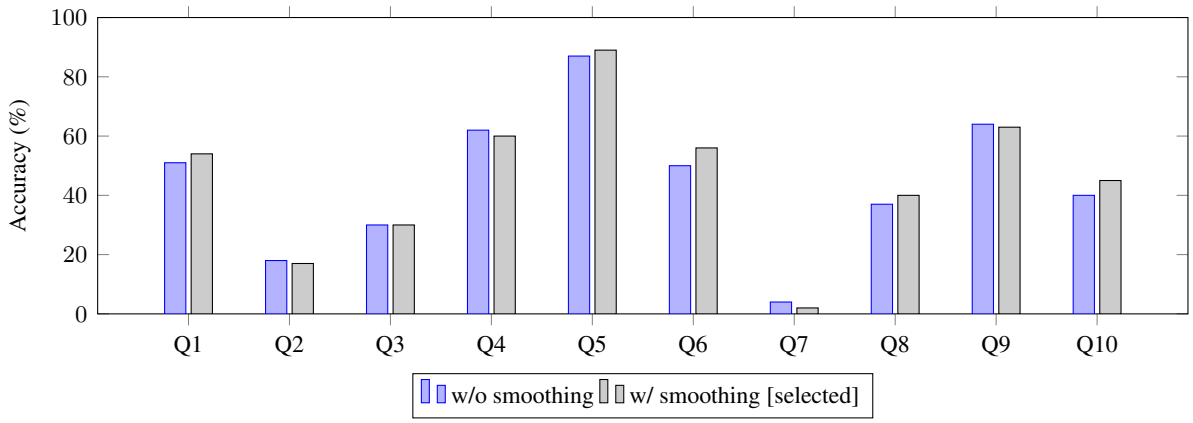

Figure 6: Effect of boundary smoothing on BABILong-16k accuracy (%). α=0.75, n=4. Shaded bars: selected configuration.