Cache You Later: Post-Compression KV Repair for Long-Context Agentic LLM Inference

Andrew Rusli [*1] **Shreyan Paliwal** [*1 2] **Michael Jiao** [1]

Abstract

As large language models (LLMs) support longer contexts and more capable agentic sessions, what matters in the accumulated context can change with each new query, tool output, or test result. At the same time, inference still depends on a limited active key-value (KV) cache, and many existing KV-cache compression methods decide which past tokens stay active from the current prefix. As a result, tokens that matter in later turns may already have been evicted or compressed away in earlier compression decisions. We argue the KV cache should be multi-tiered and governed by a two-way mechanism: both eviction and repair, not just eviction. We propose REPAIRKV, a runtime operator for multi-turn agentic inference that reevaluates which evicted KV rows are currently important and restores a budgeted subset from host memory back into the GPU KV cache. Controlled experiments show that REPAIRKV's restoration of evicted tokens improves retrieval accuracy, with the effect persisting across relevance changes and different initial eviction policies, and a preliminary diagnostic on open-source repositories shows the same pattern; on Qwen2.5-7B-Instruct at 32K context, REPAIRKV achieves 91.0% retrieval on a four-query needle-in-a-haystack task versus 24.5% for the matched no-repair baseline at the same active-cache budget, with only 96 promoted tokens.

1. Introduction

Modern language models support context lengths from tens of thousands to over a million tokens for retrieval-augmented and agentic workflows (Hsieh et al., 2024; Bai et al., 2024; Zhang et al., 2024), but the KV cache holding past token keys and values grows linearly with sequence length, making it a primary GPU memory bottleneck and an increasing runtime cost during decoding. KV-cache compression and offload (Li et al., 2024a; Xiao et al., 2024; Zhang et al., 2023; Kwon et al., 2023) manage this cost by deciding which context remains active on GPU and which part is evicted or moved to a slower tier. This decision is usually made with information available at the current turn, and compression is often treated as a one-way operation: evict now, decode later from the retained active set.

This view misses a central feature of multi-turn inference: new turn signals from retrieved documents, tool outputs, test failures, or file changes can make previously low-priority context newly important. These workflows also create non-decoding intervals while the model waits for external results. Together, the new signal and the pause create an opportunity to repair the active KV state before the next decode, but no existing work studies whether an already-compressed cache can be revised in runtime under a matched active-cache budget.

This paper studies this gap. Our contributions are: (1) a matched resumed active-cache-budget protocol for testing whether repair helps without simply keeping more active tokens; (2) REPAIRKV, an operator conditioned on the next turn's signal that scores evicted KV rows and promotes a small budgeted subset back into the active cache before decoding resumes; (3) controlled evidence across synthetic relevance shifts, different initial eviction policies, preliminary real-repository diagnostics, and runtime feasibility probes. On Qwen2.5-7B-Instruct at 32K context, REPAIRKV achieves 91.0% retrieval on a four-query needle-in-a-haystack task versus 24.5% for the matched no-repair baseline at the same active-cache budget, with only 96 promoted tokens.

2. Related Work

2.1. KV-cache compression and eviction

KV-cache compression and eviction methods reduce active KV state by retaining a fixed budget of tokens or blocks based on prefix-time, recency, or attention signals (Li et al.,

[*]Equal contribution [1]Harvard College, Cambridge, MA, USA [2]Spring Silicon. Correspondence to: Andrew Rusli <andrewrusli@college.harvard.edu>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

2024a; Xiao et al., 2024; Zhang et al., 2023; Liu et al., 2023; Li et al., 2024b), but this retention decision is usually one-way: rows dropped from the active set cannot re-enter when later turns make them important. This failure mode is observed in recent analyses of instruction and fact degradation under KV compression (Chen et al., 2025).

2.2. Query-aware retrieval and sparse attention

Query-aware methods such as QUEST emphasize the premise behind repair: token criticality depends on current information need, not only on the prefix that produced the compressed cache (Tang et al., 2024). Fine-grained KV retrieval systems use this signal during active decoding to select or fetch query-relevant rows from a retained or indexed KV store (Tang et al., 2024; Wang et al., 2025; Qi et al., 2026; Sun et al., 2025). Attention analyses likewise show that token importance can be query-dependent or recurrent rather than fixed once compressed (Wu et al., 2024; Zhang et al., 2025). REPAIRKV builds on this query-dependence, but applies it at a different lifecycle point: after compression and before the next decode.

2.3. Long-context benchmarks

Long-context benchmarks stress whether models can use information across long inputs, from ordinary long documents to 100k-token settings, showing that application-visible context length does not remove the need for memory-management mechanisms (Bai et al., 2024; Zhang et al., 2024; Hsieh et al., 2024). SCBench directly evaluates tasks around the KV-cache lifecycle itself (Li et al., 2025). However, these benchmarks leave open what happens after a compression decision has already removed future-relevant tokens from the active cache.

2.4. KV serving and tiering

Serving systems address the systems side of long-context KV cache management. PagedAttention/vLLM improves GPU KV allocation and block management during serving (Kwon et al., 2023), while prefix-caching systems reduce repeated prefill when requests share or resume prefixes (Liu et al., 2025; Dzikanyanga et al., 2026). KV-tiering systems extend this idea across GPU, host memory, storage, and network tiers (Liu et al., 2025; Dzikanyanga et al., 2026; Zhang et al., 2026). Together, these systems determine where KV state lives and when it can be reused, but they do not address the complementary decision of which offloaded rows should re-enter the active cache for the next turn.

2.5. Multi-turn LLM serving

Multi-turn agent workloads pause decoding for external actions such as tool calls, web interaction, repository edits, and tests (Qin et al., 2024; Zhou et al., 2024; Jimenez et al., 2024; Liu et al., 2024). Serving systems make these pauses operationally visible: Continuum schedules multi-turn agent requests around tool-call gaps and KV-cache lifetimes, while AgentServe targets local agent serving under tight GPU memory budgets (Li et al., 2026; Zhang et al., 2026). Pause-aware systems preserve paused-request KV to avoid repeated prefill (Abhyankar et al., 2024; Li et al., 2026; Gao et al., 2024). These same pauses often contain new relevance signals, creating the setting for repair.

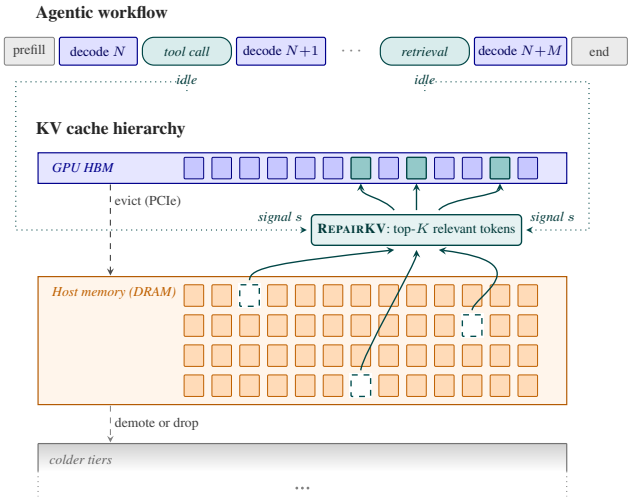

Figure 1. REPAIRKV system overview. During agent decode pauses (tool calls, retrievals), a selected signal s (such as a new user query, a tool result, retrieved content, or the model's recent generation) is used to rank host KV rows; the top-K are lifted back into GPU HBM at their original sequence positions.

3. REPAIRKV

3.1. Overview

Each compression step can only choose the active KV set with the information available at that point. REPAIRKV assumes a tiered KV cache: evicted rows are offloaded to a slower tier rather than discarded. It then treats the resulting compressed state as revisable: when a new turn signal becomes available, the runtime can score offloaded evicted KV rows and promote a small subset before decoding resumes. Figure 1 summarizes this system view.

At the pause boundary, the abstraction has three objects:

- C_{base}: active evictable KV, with $|C_{\text{base}}| = B_{\text{base}}$ (base active-cache budget);
- W_N: offloaded evicted KV, hidden from decoding unless promoted;
- s: the pause boundary signal used to score evicted KV, such as a new user query, a tool result, the model's recent generation, or other tokens that indicate upcoming significance.

The repair map chooses $S_K(s) \subseteq W_N$ with $|S_K| \leq K$, where K is the restore budget, and resumes from $C_{\text{repair}} = C_{\text{base}} \cup S_K$ at resumed active-cache budget $|C_{\text{repair}}| \leq B_{\text{base}} + K$.

3.2. Two-turn protocol

We isolate multi-turn repair with a protocol for the simplest controlled case: one relevance shift across two turns. A five-turn revisit diagnostic (Figure 3) confirms that the effect persists across repeated shifts.

At turn N, the model receives a long distractor document followed by prompt Q_1 and generates the response to Q_1 from the full uncompressed cache. The runtime then compresses the distractor document KV to a base budget B_{base} on GPU and offloads the evicted rows to host memory as W_N. The Q_1 prompt and the generated response are preserved exactly. During the subsequent idle window, the turn-$(N{+}1)$ prompt Q_2 becomes known and is preserved verbatim.

At turn $N{+}1$, the distractor document is not re-presented. REPAIRKV scores the evicted store W_N against the active post-Q_1 cache and the Q_2 signal, promotes a K-budgeted subset S_K back to GPU before Q_2 decoding, and the model answers Q_2 from C_{repair}.

The primary comparison excludes full-prefill recompute because it reprocesses the long context after Q_2 is known, while our comparison asks whether the already-compressed cache can be improved under the same resumed active-cache budget. The controlled prototype evaluates one paused request on one GPU with a single host-memory tier, a prematerialized evicted-KV store, and the full Q_2 signal known before resumed decoding.

3.3. Repair operation

Intuitively, REPAIRKV uses the next-turn signal to identify which evicted KV rows should be active before decoding resumes. Algorithm 1 describes the general framework; the prototype is specified by two main choices: SCORE and SELECT, with $\mathcal{C} = C_{\text{base}}$ and $\mathcal{W} = W_N$.

For SCORE, post-RoPE Q_2 query tensors $\{q_{2,t}^{(\ell,h)}\}$ are scored against keys in the active and offloaded sets via

$$A^{(\ell,h)} = \text{softmax}\left(\frac{q_{2,:}^{(\ell,h)}\,[K_{C_{\text{base}}},\, K_{W_N}]^\top}{\sqrt{d_k}}\right), \quad (1)$$

restricted to columns in W_N, then aggregated as $a_i = \text{mean}_{\ell,h} \max_t A_{t,i}^{(\ell,h)}$. Because the softmax normalizes over the active and offloaded keys, a_i is the attention mass row i would receive given C_{base} rather than a raw inner product. Ties break by the per-token score from the turn-N initial compression, then by ascending position.

Algorithm 1 General REPAIRKV Execution Framework

Input: turn signals $\{s_N\}_{N=1}^{T}$, eviction policy $\mathcal{P}$, base budget B_{base}, restore budget K.
Initialize: active cache $\mathcal{C}$, evicted store $\mathcal{W} \leftarrow \emptyset$.
for $N = 1$ to T **do**
  $\text{ans}_N \leftarrow \text{DECODE}(\mathcal{C}, s_N)$ // Decoding
  $\mathcal{C}, \mathcal{W} \leftarrow \mathcal{P}(\mathcal{C}, \mathcal{W}, B_{\text{base}})$ // Compression
  if $N < T$ **then**
    obtain s_{N+1} // Idle window begins
    for $i \in \mathcal{W}$ **do**
      $a_i \leftarrow \text{SCORE}(i \mid \mathcal{C}, s_{N+1})$ // Scoring
    end for
    rank $\mathcal{W}$ by a_i // Ranking
    $S_K \leftarrow \text{SELECT}(\mathcal{W}, K)$, $|S_K| \leq K$ // Selection
    $\mathcal{C} \leftarrow \mathcal{C} \cup S_K$; $\mathcal{W} \leftarrow \mathcal{W} \setminus S_K$ // Repair
  end if
end for

For SELECT, each top-ranked anchor i is expanded into a local burst $\{i{-}L, \ldots, i{+}R\} \cap (W_N \setminus S_K)$ under a strict $|S_K| \leq K$ budget. The right-biased $(L, R) = (2, 20)$ reflects the local structure: high-scoring anchors often occur near queried keys, and answer-bearing value tokens usually follow nearby. If anchor bursts do not exhaust K, remaining slots backfill from single ranked rows.

The current prototype scores token rows with fixed burst widths. A systems implementation would promote cached blocks or pages.

3.4. Matched-budget evaluation

All main claims compare REPAIRKV to a no-repair baseline at the same resumed active-cache budget. In both conditions, Q_2 decoding sees the same number of active evictable-context tokens in addition to the identical preserved Q_1 prompt and answer tail. The match is only on active GPU KV: extra offloaded rows, scorer time, and transfer time are not part of the matched budget; they are reported separately as service costs.

Let B_{match} denote the no-repair condition, which applies the base eviction policy with active evictable-context budget $B_{\text{base}} + K$. REPAIRKV reaches the same active budget by starting from B_{base} retained rows and promoting up to K rows from the host-memory store.

The main restore-budget frontier reports raw answer score. For effect size comparisons, we also report gain over matched no-repair,

$$\Delta_{\text{repair}} = \text{Score}(\text{RepairKV}) - \text{Score}(B_{\text{match}}) \quad (2)$$

where Score is the mean fraction of held-out Q_2 targets recovered correctly per example over the fixed evaluation set shared across all conditions. Stricter exact-set and precision-

sensitive scoring serve as confirmatory checks.

4. Experimental Setup

4.1. Benchmark family

The main benchmark is a split-query version of multi-query needle-in-a-haystack (MQ-NIAH) derived from RULER (Hsieh et al., 2024). We use this family because it exposes the failure mode repair is meant to address: tokens that are unimportant for the first turn can become critical post-compression.

Each example contains key-value needles inside a long distractor context. The first turn queries one subset of keys, the cache is compressed, and the second turn queries a different subset. This gives a controlled relevance shift with annotated future-relevant spans and a shared turn-N history across conditions.

We evaluate 2Q, 4Q, 6Q, and 8Q variants, which use the same task template but increase the number of queried key-value pairs. For each variant, we pool balanced partitions that avoid a recency shortcut: the keys requested in Q_2 are not the needles nearest to the end of the context. Appendix Figure 7 summarizes the exact splits.

4.2. Evaluation configuration

All main experiments use Qwen2.5-7B-Instruct (Qwen, 2024) at 32K rendered context on a single RTX PRO 6000 Blackwell GPU. The initial post-Q_1 compression is context-only SnapKV compression (Li et al., 2024a): it is applied only to the evictable document context, while the Q_1 prompt and generated Q_1 response are preserved exactly.

Per-task base budgets are calibrated so matched no-repair is neither already near full-cache accuracy nor near zero, leaving room to measure whether repair improves the resumed cache. Unless stated otherwise, the fixed sweep is

- compressor: context-only SnapKV with $R_{\text{ctx}} = 128$ (SnapKV observation-window length);
- base budgets: $B_{\text{base}} = 8192$ for 2Q, $B_{\text{base}} = 16384$ for 4Q, and $B_{\text{base}} = 18432$ for 6Q/8Q;
- restore budgets: $K = 8, 16, 24, 32, 48, 64, 80, 96, 128$;
- replay: one shared full-cache-generated Q_1 response per setting and condition.

This replay removes turn-N generation differences, so the frontier isolates resumed-cache selection quality at turn $N{+}1$. MQ-NIAH-2Q/4Q/6Q use fixed 100-example evaluation sets per partition, while MQ-NIAH-8Q uses 24 examples per partition over five partitions. Prompt and decode settings are reported in the appendix.

4.3. Scorers

For the initial turn-N compression, we score context tokens using the generated Q_1 response rather than a generic last-32-token suffix. This makes the post-Q_1 compressed cache reflect the actual first turn computation.

For repair, the main quality curves use an exact Q_2 scorer. We append Q_2 to the compressed active cache, extract the model's post-RoPE Q_2 query projections, and score active and offloaded key rows under those projections before selecting evicted rows for promotion. This isolates whether the newly revealed turn signal identifies rows worth restoring.

4.4. Baselines and references

We compare the full-cache reference, the base compressed cache before repair, matched no-repair B_{match}, Random-K and Oldest-K restore controls, and RepairKV. Random-K and Oldest-K promote from the same evicted KV store as RepairKV without scoring with Q_2, testing whether gains come from query-aware repair or generic reinsertion.

The specificity study adds StaleQ-K (score with the previous turn query), WrongQ-K (score with another example's Q_2) and Refresh-buffered, which reselects the whole $B_{\text{base}}+K$ context budget from active and offloaded rows after Q_2 is known, without full-prefix recomputation. Refresh-buffered exposes selector headroom rather than acting as a main baseline.

5. Results

5.1. Matched-budget frontier

Across MQ-NIAH-2Q/4Q/6Q/8Q, RepairKV separates from matched no-repair at moderate to large restore budgets. At $K = 96$, RepairKV scores 0.910 on 4Q, 0.989 on 6Q, and 0.960 on 8Q, while matched no-repair scores 0.245, 0.422, and 0.546, respectively. Figure 2 reports the full raw Q_2 score frontier. Random-K and Oldest-K stay near matched no-repair, showing that generic reinsertion from the same evicted KV store does not explain the gains (Appendix Fig. 8). MQ-NIAH-2Q saturates by $K = 80$, while matched no-repair remains near 0.06. Across all twelve balanced partitions at $K = 128$, RepairKV beats B_{match} and both content-agnostic controls. Even the weakest partition-level gains are 0.585, 0.387, and 0.292 for 4Q, 6Q, and 8Q, respectively. (Appendix Figure 7).

5.2. Next-turn signal specificity

Repair gains depend on using the correct newly revealed signal. At $K = 48$, StaleQ-K and WrongQ-K stay near matched no-repair, with gains 0.028 $[-0.021, 0.083]$ and 0.028 $[-0.014, 0.069]$, while RepairKV reaches 0.326

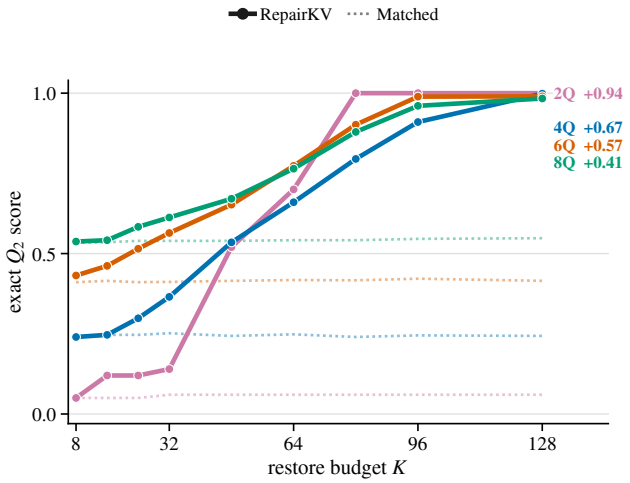

Figure 2. Raw Q_2 score under matched resumed-cache budgets. Solid: REPAIRKV; dotted: matched no-repair. Right labels: query count and Δ_{96} (gain at K=96).

$[0.243, 0.410]$. Thus, the improvement is not explained by scoring with any query-like text; the signal must match the upcoming turn. Refresh-buffered reaches 1.000, showing that relevant rows remain available for stronger selectors to exploit, motivating an open research direction.

Table 1. Next-turn signal specificity at $K = 48$ on MQ-NIAH-4Q (72 paired example-partition cases). Stale and donor controls test whether any query-like signal suffices; Refresh is a full-budget reselector over active and offloaded KV.

Condition	Q_2 score	Δ vs. matched [95% CI]
Matched no-repair	0.215	–
StaleQ-K	0.243	$+0.028$ $[-0.021, 0.083]$
WrongQ-K	0.243	$+0.028$ $[-0.014, 0.069]$
REPAIRKV	0.542	$+0.326$ $[0.243, 0.410]$
Refresh-buffered	1.000	$+0.785$ $[0.722, 0.847]$

5.3. Repeated relevance shifts

The repair effect persists when relevance changes repeatedly. On a fixed five-turn MQ-NIAH-8Q revisit schedule ($K = 80$, $n = 24$), REPAIRKV gains 0.542 over matched no-repair on non-initial turns (95% CI $[0.458, 0.620]$). Random-K and Oldest-K gain only 0.010 and 0.021, and REPAIRKV remains above StaleQ-K, suggesting that repair can track the current turn signal across repeated controlled shifts.

5.4. Eviction-policy sensitivity

REPAIRKV also improves caches produced by other first-stage eviction policies. On MQ-NIAH-4Q, we replace the primary SnapKV-style compressor with H_2O-style accumulated attention retention and a StreamingLLM-style sink-plus-recent rule. In both cases, REPAIRKV improves over each policy's matched no-repair baseline while Random-K and Oldest-K remain near baseline (Figure 4). Under the StreamingLLM-style policy, for example, REPAIRKV gains 0.431 at $K = 128$. A separate Scissorhands-style persistence stress test also clears the matched-budget protocol (Appendix Table 3).

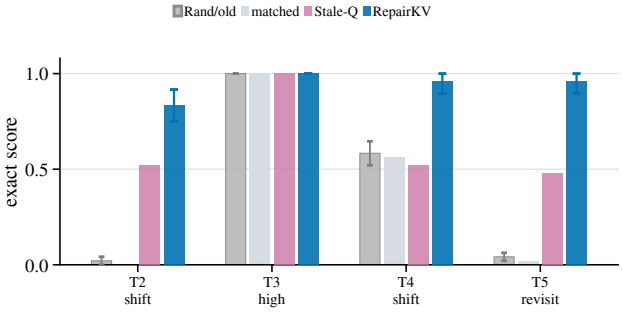

Figure 3. Five-turn relevance-shift diagnostic on MQ-NIAH-8Q (B_{base}=18,432, K=80, n=24). Groups show post-priming exact scores at equal active-cache budget. Intervals are bootstrap 95% CIs for REPAIRKV, and Stale-Q uses the previous-turn query.

5.5. Real-repository diagnostic

To test external validity beyond MQ-NIAH, we ran a controlled relevance shift diagnostic on 48 callsite examples from 12 open-source repositories drawn from the SWE-bench pool (Jimenez et al., 2024) at fixed commits. Each example loads 32K tokens of line-numbered repository context. The first turn asks about a file unrelated to the answer, encouraging compression to evict the later relevant code. The second turn asks for a target identifier at a callsite.

Between turns, an *event* arrives: a description of the callsite with the target identifier redacted. This event replaces the synthetic Q_2 signal used in MQ-NIAH, and all conditions decode the same event and question prompt. At $K = 192$, event-only REPAIRKV improves exact identifier accuracy from 18.8% matched no-repair to 72.9%. Non-query controls and ToolFile-K remain near matched (Table 2). Thus, the gain comes from using the event as a relevance signal, not from generic reinsertion or simply knowing the file name.

Label-assisted references show remaining headroom: File-gated REPAIRKV reaches 83.3%, and AnchorWindow-K reaches 89.6%. This is a small diagnostic rather than a full benchmark, but it shows some initial evidence that the repair approach improves quality beyond the synthetic MQ-NIAH setting. Appendix Figure 6 gives the full audit.

5.6. Runtime

We run a separate GPU runtime probe for the repair operation, measuring chunked scan, top-K selection, and KV

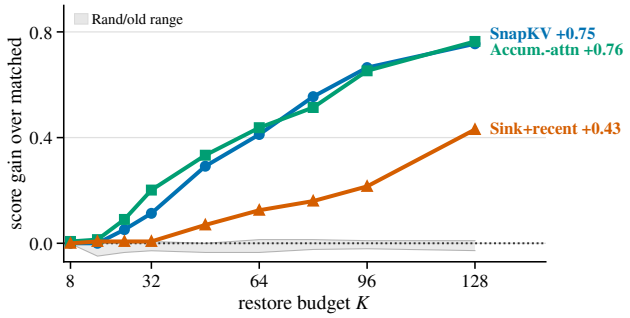

Figure 4. Eviction-policy sensitivity on MQ-NIAH-4Q (n=24, B_{base}=16,384). Curves show Q_2 gain over each policy's matched no-repair; gray band spans Random-K/Oldest-K.

Table 2. Real-repository diagnostic at K=192 on 48 callsite cases from 12 SWE-bench repositories. Accuracy = exact target-identifier recovery. Non-query = best of Random-K/Oldest-K/StaleQ-K/WrongQ-K. AnchorWindow and File-gated are label-assisted (others non-label).

Condition	Accuracy
Matched no-repair	18.8%
Best non-query control	16.7%
ToolFile-K	18.8%
RepairKV	72.9%
File-gated RepairKV	83.3%
AnchorWindow-K	89.6%

movement over pinned host-memory BF16 tensors with Qwen-shaped dimensions. At $K = 5000$, p95 repair latency is 50.0 ms at 32K offloaded candidates, 1.20 s at 1M, and 4.64 s at 4M (Figure 5). Although these numbers are hardware- and model-dependent, the key property is that cost scales linearly with the number of candidate KV rows, which keeps repair in the seconds range even at million-row candidate pools, unlike approaches that recompute or score over full context-query interactions with quadratic cost. At K=96 the full repair takes around 110 ms (p95) versus 2.13 s for full-prefix prefill on our system, even with FlashAttention-2/3 speedups (Dao, 2023; Shah et al., 2024); see Appendix C for selector ablations and time-budgeted query-aware baselines.

These costs are small relative to many agent and tool-use pauses: OS-level execution, including tool calls and setup, can account for 56–74% of task latency (Zheng et al., 2026), and tool calls range from sub-second file/shell operations to multi-second tests or code execution (Li et al., 2026). More broadly, the low repair cost suggests a runtime role beyond using otherwise idle GPU time: repair can be run opportunistically at turn boundaries, tool results, retrieval updates, or other state changes to trade a small amount of memory movement and time for substantially better resumed-cache quality.

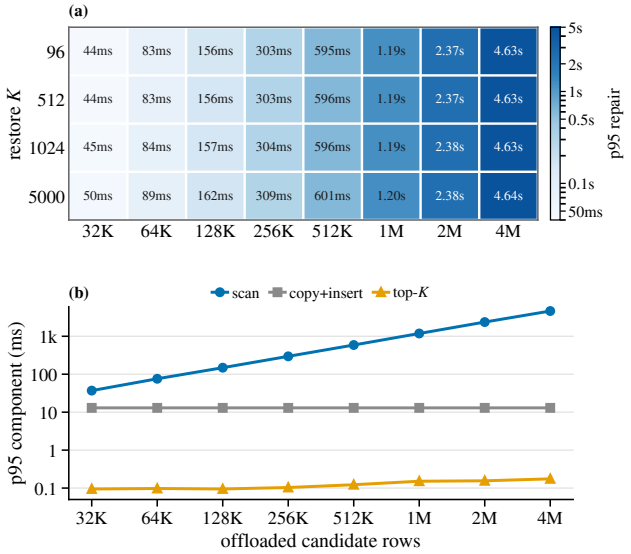

Figure 5. Runtime-capacity probe. (a) p95 of chunked scan + top-K select + KV move at query length 64, Qwen-shaped pinned host BF16 KV. (b) Stage-wise p95 at K=5000.

6. Conclusion

RepairKV makes the case for revisable KV compression. Existing compression policies decide which past tokens remain active using the information available at compression time. Our results show that this decision can be usefully revised when a later turn reveals a new relevance signal. Across MQ-NIAH-2Q/4Q/6Q/8Q, repeated relevance shifts, alternative eviction policies, and a small real-repository diagnostic, promoting a small set of relevant evicted rows substantially improves resumed-cache quality under the same active-cache budget.

The broader direction is multi-tier, mutable KV memory for resource-adaptive long-context inference. Repair suggests that runtimes need not treat compression as a one-way loss: they can maintain searchable evicted state, score it when new signals arrive, and use cheap promotion to trade a small amount of memory movement for better response quality. This opens several follow-up directions: stronger Q_2-aware selectors, page-level repair policies, scheduler-aware repair at arbitrary turn or tool events, and benchmarks with production-style multi-turn traces, hidden future relevance, and explicit accounting for active-cache, offloaded storage, transfer, and recompute cost.

Impact Statement

This paper studies inference-time KV-cache repair for long-context language models. Its intended impact is to reduce GPU memory pressure while improving response quality in long-context and agentic settings where relevance changes between turns. We do not foresee novel ethical risks beyond those already inherent to large language model use.

References

Abhyankar, R., He, Z., Srivatsa, V., Zhang, H., and Zhang, Y. InferCept: Efficient intercept support for augmented large language model inference. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., Dong, Y., Tang, J., and Li, J. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Chen, A., Geh, R., Grover, A., Van den Broeck, G., and Israel, D. The pitfalls of KV cache compression. *arXiv:2510.00231*, 2025.

Dao, T. FlashAttention-2: Faster attention with better parallelism and work partitioning. *arXiv:2307.08691*, 2023.

d-Matrix. Corsair: In-memory computing for AI inference. Hot Chips, 2025.

Dzikanyanga, G., Yang, W., Huang, H., Wu, D., Wang, S., Xia, W., and K C, S. TTKV: Temporal-tiered KV cache for long-context LLM inference. *arXiv:2604.19769*, 2026.

Gao, B., He, Z., Sharma, P., Kang, Q., Jevdjic, D., Zhao, J., and Zuo, P. Cost-efficient large language model serving for multi-turn conversations with CachedAttention. In *Proceedings of the 2024 USENIX Annual Technical Conference (USENIX ATC)*, 2024.

Hsieh, C.-Y., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., and Ginsburg, B. RULER: What's the real context window size of your LLM. In *Conference on Language Modeling (COLM)*, 2024.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. SWE-bench: Can language models resolve real-world GitHub issues. In *International Conference on Learning Representations (ICLR)*, 2024.

Kim, D., Lee, M., Kim, J., Kwon, H., Jeong, H., Park, S.-S., Yoon, M., Roh, S.-D., Kwon, Y., So, J., and Choi, J. Scalable processing-near-memory for 1M-token LLM inference: CXL-enabled KV-cache management beyond GPU limits. *arXiv:2511.00321*, 2025.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

Li, H., Mang, Q., He, R., Zhang, Q., Mao, H., Chen, X., Zhou, H., Cheung, A., Gonzalez, J., and Stoica, I. Continuum: Efficient and robust multi-turn LLM agent scheduling with KV cache time-to-live. *arXiv:2511.02230*, 2026.

Li, Y., Huang, Y., Yang, B., Venkitesh, B., Locatelli, A., Ye, H., Cai, T., Lewis, P., and Chen, D. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Li, Y., Jiang, H., Wu, Q., Luo, X., Ahn, S., Zhang, C., Abdi, A. H., Li, D., Gao, J., Yang, Y., and Qiu, L. SCBench: A KV cache-centric analysis of long-context methods. In *International Conference on Learning Representations (ICLR)*, 2025.

Li, Z., Yang, Y., Zhang, Y., and Liu, Z. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv:2406.02069*, 2024.

Liu, D. and Yu, Y. CXL-SpecKV: A disaggregated FPGA speculative KV-cache for datacenter LLM serving. *arXiv:2512.11920*, 2025.

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., Zhang, S., Deng, X., and others. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations (ICLR)*, 2024.

Liu, Y., Cheng, Y., Yao, J., An, Y., Chen, X., Feng, S., Huang, Y., Shen, S., Zhang, R., Du, K., and Jiang, J. LMCache: An efficient KV cache layer for enterprise-scale LLM inference. *arXiv:2510.09665*, 2025.

Liu, Z., Desai, A., Liao, F., Wang, W., Xie, V., Xu, Z., Kyrillidis, A., and Shrivastava, A. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

NVIDIA. NVIDIA BlueField-4 powers new class of AI-native storage infrastructure for the next frontier of AI. NVIDIA Newsroom, January 2026. Announces the Inference Context Memory Storage Platform (ICMSP) at CES 2026.

Qi, Y., Chen, X., Jiang, H., Wang, Q., Peng, B., and Palpanas, T. ParisKV: Fast and drift-robust KV-cache retrieval for long-context LLMs. *arXiv:2602.07721*, 2026.

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Hong, L., and others. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*, 2024.

Qwen Team. Qwen2.5 technical report. *arXiv:2412.15115*, 2024.

SambaNova Systems. SN40L reconfigurable dataflow unit, 2024.

Shah, J., Bikshandi, G., Zhang, Y., Thakkar, V., Ramani, P., and Dao, T. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Sun, H., Chang, L.-W., Bao, W., Zheng, S., Zheng, N., Liu, X., Dong, H., Chi, Y., and Chen, B. ShadowKV: KV cache in shadows for high-throughput long-context LLM inference. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Tang, J., Zhao, Y., Zhu, K., Xiao, G., Kasikci, B., and Han, S. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Wang, D., Liu, Z., Wang, S., Ren, Y., Deng, J., Hu, J., Chen, T., and Yang, H. FIER: Fine-grained and efficient KV cache retrieval for long-context LLM inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

Wu, W., Wang, Y., Xiao, G., Peng, H., and Fu, Y. Retrieval heads mechanistically explain long-context factuality. *arXiv:2404.15574*, 2024.

Xiao, G., Tang, Y., Zuo, J., Guo, J., Yang, S., Tang, H., Fu, Y., and Han, S. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations (ICLR)*, 2024.

Yoon, D., Min, Y., Kim, H., Noh, S. H., and Kim, J. TraCT: Disaggregated LLM serving with CXL shared memory KV cache at rack-scale. *arXiv:2512.18194*, 2025.

Zhang, H., Zhang, H., Ma, X., Zhang, J., and Guo, S. LazyEviction: Lagged KV eviction with attention pattern observation for efficient long reasoning. *arXiv:2506.15969*, 2025.

Zhang, X., Chen, Y., Hu, S., Xu, Z., Chen, J., Hao, M. K., Han, X., Thai, Z. L., Wang, S., Liu, Z., and Sun, M. ∞Bench: Extending long context evaluation beyond 100K tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Zhang, Y., Yan, Y., Yang, N., and Yuan, D. AgentServe: Algorithm-system co-design for efficient agentic AI serving on a consumer-grade GPU. *arXiv:2603.10342*, 2026.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., Wang, Z., and Chen, B. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zheng, Y., Fan, J., Fu, Q., Yang, Y., Zhang, W., and Quinn, A. AgentCgroup: Understanding and controlling OS resources of AI agents. *arXiv:2602.09345*, 2026.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., and Neubig, G. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2024.

A. Outlook and open directions

We share this prototype as preliminary evidence that compressed KV state can be usefully revised at a turn boundary. The patterns we observe across matched active-cache budgets, eviction-policy variations, repeated relevance shifts, and a small real-repository diagnostic suggest a mechanism worth deeper study. We propose several extensions as natural next steps.

Scale and coverage. With more compute, we plan to run the matched-budget protocol on additional instruct-tuned families (Llama, Mistral), at larger model scale (70B-class), and at longer contexts (64K, 128K, 256K), where eviction errors compound and the room for repair grows.

Stronger benchmarks. SCBench and full SWE-bench resolution are potential short-term next steps. The ideal benchmark would be closer to production multi-turn agent traces with real tool calls.

Stronger comparisons. Running query-aware retrieval methods (e.g., QUEST, ShadowKV) as repair operators under the matched active-cache-budget protocol would isolate selector quality from architectural commitments, alongside end-to-end serving evaluations under realistic deployment baselines.

The mechanism also fits naturally into the multi-tier KV systems that have become a frontier of LLM serving. NVIDIA's ICMSP makes NVMe-resident KV part of the context address space (NVIDIA, 2026), CXL-based KV systems push the warm tier off-package with coherent host-side access (Kim et al., 2025; Yoon et al., 2025; Liu and Yu, 2025), and inference accelerators are exposing tiered memory architectures (d-Matrix, 2025; SambaNova, 2024). The open runtime question across these stacks, namely which slow-tier rows to promote back into the active tier under a budget and on what signal, is the question this mechanism is designed to address. We see revisable compressed KV as a research primitive worth broader investigation.

B. Additional Evaluation Details

Protocol details. Concise split prompts request values only, comma-separated, with a short answer prefix and fixed decode budgets of 24 tokens for MQ-NIAH-4Q, 48 tokens for MQ-NIAH-6Q, and 64 tokens for MQ-NIAH-8Q. Generation is deterministic greedy decoding with no sampling. The main examples use dataset seed offset 0. Random-K uses a deterministic example- and K-dependent seed. Bootstrap intervals are percentile intervals over paired example-partition rows with fixed seeds. Main Qwen runs use BF16 weights on one RTX PRO 6000 Blackwell GPU with Python 3.12.3, PyTorch 2.10.0+cu128, and Transformers 5.2.0. The exact-Q_2 selector described in Section 4 is distinct from decode-time answer-token attention and from the final answer metric.

Span-group diagnostic. $\mathrm{SpanRef}\text{-}K$ starts from the same B state and offloaded evicted-KV store as REPAIRKV. It enumerates the small family of annotated Q_2 answer-span-group subsets, one key/value span group per requested needle, evaluates each subset once under actual Q_2 generation, and chooses the highest-scoring subset with total cost at most K. It is exact over this benchmark-specific span-group family, but not over all arbitrary token subsets or possible final generations. Because the selected span-group subset may use fewer than K rows, a repair selector can occasionally score higher at low budgets by spending the full budget on useful local neighborhoods around future-relevant spans. Thus $\mathrm{SpanRef}\text{-}K$ is a diagnostic span-group reference, not a ceiling.

Partition choices. For MQ-NIAH-4Q, the pooled panel uses $14\rightarrow23$, $24\rightarrow13$, and $34\rightarrow12$. These are the balanced $2|2$ partitions whose Q_2 side excludes the tail-anchored fourth needle. For MQ-NIAH-6Q, the pooled panel uses $156\rightarrow234$, $256\rightarrow134$, $356\rightarrow124$, and $456\rightarrow123$, so the second turn excludes the two latest needles. For MQ-NIAH-8Q, the pooled panel uses $5678\rightarrow1234$, $1678\rightarrow2345$, $2678\rightarrow1345$, $3678\rightarrow1245$, and $4678\rightarrow1235$, again excluding the two latest needles from the second turn. Within-turn order is not treated as a separate condition because the prompt asks for comma-separated values only and scoring checks whether each requested value is recovered.

Scissorhands-style persistence stress test. Table 3 repeats the MQ-NIAH-6Q matched-budget protocol with the initial SnapKV-style eviction replaced by a one-shot persistence policy inspired by Scissorhands, testing whether repair still works when the compressed cache is chosen by attention persistence rather than the primary eviction policy. Canonical online per-head Scissorhands reproduction remains future work.

Table 3. Scissorhands-style persistence stress test on MQ-NIAH-6Q (B_{base}=18,432, n=24). Values are exact scores; full-cache reference is 0.990 for all K. Best ctrl. is the stronger of Random-K and Oldest-K; CI is the paired bootstrap 95% interval for REPAIRKV$-B_{\mathrm{match}}$.

K	B_{match}	Ctrl.	REPAIRKV	CI
48	0.365	0.382	0.712	[0.292, 0.403]
64	0.365	0.382	0.795	[0.378, 0.490]
80	0.378	0.385	0.934	[0.503, 0.604]
96	0.375	0.382	0.993	[0.566, 0.670]
128	0.368	0.378	0.990	[0.573, 0.674]

C. Time-Matched Selector Ablations

Time-budgeted baselines. Two additional baselines extend the protocol of Section 4, both subject to a wall-clock budget matched to REPAIRKV's measured per-pause repair time. **Refresh-K-budgeted** is the Refresh-buffered baseline of Section 4 with this wall-clock budget applied: when the budget runs out, it stops scoring further evicted positions. **PageSummary-Quest-inspired** pre-computes per-chunk summary keys (max over each chunk) at compression time, ranks chunks by a cheap inner product against Q_2, and within the same wall-clock budget visits top-ranked chunks using the same per-position scorer. Both methods choose from the same evicted store as REPAIRKV and are matched on resumed active-cache rows ($B_{\text{base}}+K$); only their off-budget service costs (scoring time, transfer, projection) differ. All numbers in this supplement come from a smaller paired n=36 MQ-NIAH-4Q slice, run separately to keep the wall-clock-budgeted baselines tractable; small differences from the n=300 headline numbers in Section 5 (e.g., B_{match}=0.208 here vs. 0.245 there) reflect this resampled slice rather than a protocol change.

Wall-clock-matched K-sweep. On MQ-NIAH-4Q at a 150 ms absolute scoring budget across $K \in \{32, 64, 96, 128\}$ (Qwen2.5-7B-Instruct, B_{base}=16,384, n=36 paired observations per K), the budget runs out before Refresh-K-budgeted finishes scoring on all 36/36 examples, scoring approximately 4,000 of 32,768 evicted positions per call. REPAIRKV dominates at $K \geq 64$, reaching 0.917 at K=96 against 0.472 for Refresh-K-budgeted and 0.194 for PageSummary-Quest-inspired (matched no-repair B_{match}=0.208). At K=128 under the same budget, REPAIRKV saturates at 1.000 while the two time-budgeted baselines remain near 0.5 and 0.2.

Per-position scoring vs. burst-expansion contributions. The gains decompose into two contributions at K=96 on Qwen. Promoting only the top-scoring evicted positions, without the burst-expansion step of Section 3, reaches 0.653 and exceeds PageSummary-Quest-inspired by Δ_{score}=+0.459. This isolates the lift from per-position scoring relative to the cheaper per-chunk-summary scorer. Adding burst expansion lifts REPAIRKV to 0.917, an additional Δ_{burst}=+0.264. Per-position scoring contributes the larger share.

D. Additional Experiments

The supplementary figures probe external validity, query-count and operating-regime breadth, partition consistency, runtime capacity, and appendix selector diagnostics.

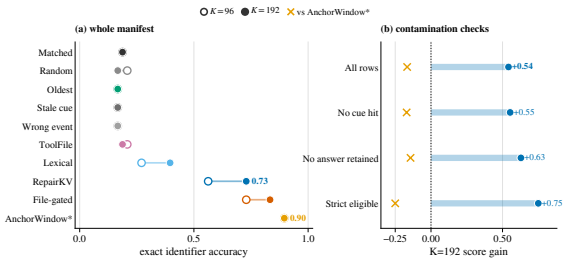

Figure 6. Controlled real-repository relevance-shift diagnostic over 48 callsite examples from 12 repositories drawn from the SWE-bench repository pool at pinned commits. Repair sees only the event cue. All conditions decode the same event-plus-question prompt. Left: exact identifier accuracy at K = 96 and K = 192. Right: K = 192 REPAIRKV gain over matched no-repair after removing cue-only or answer-retention rows. ToolFile-K is filename-assisted with oldest-row backfill. File-gated REPAIRKV restricts candidates to the event-named file before applying the event-only scorer. Lexical is an answer-sanitized diagnostic outside the deployable-selector comparison. AnchorWindow-K is a label-assisted locality reference outside the deployable-baseline and end-to-end issue-resolution comparisons.

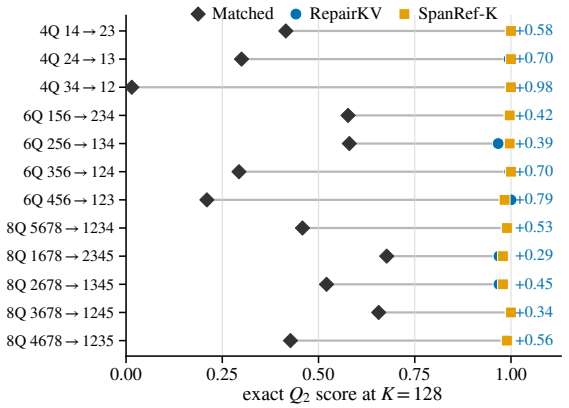

Figure 7. Per-partition endpoint scores at K = 128 on the final exact evaluations. Gray segments show the change from matched no-repair to REPAIRKV, orange points show the benchmark-metadata SpanRef-K diagnostic, and labels report the REPAIRKV difference over matched no-repair.

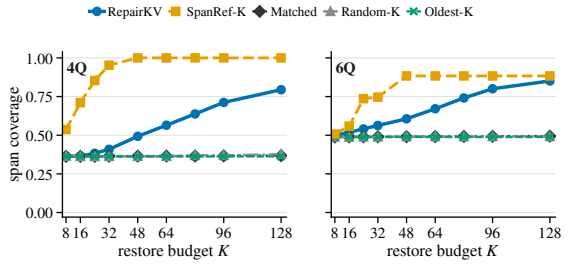

Figure 8. Mechanism diagnostic for the final exact MQ-NIAH evaluations. Curves show overlap between promoted tokens and annotated Q_2 answer spans. Content-agnostic promotions stay near baseline while REPAIRKV increasingly covers future-relevant spans.