BUDGETKV: PRE-ATTENTION KV RETENTION WITH RESIDENT INPUT EMBEDDINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context inference on memory-constrained devices must preserve nonlocal evidence while the key–value (KV) cache grows throughout prompt prefill. A request can therefore exceed available memory even when the model weights and intended final cache fit. Block-wise prefill can bound persistent state, but retention policies may still require contextual attention profiles, patched-prompt forwards, or auxiliary semantic encoders before deciding what matters. We introduce BudgetKV, a training-free method for requests whose task query is known before shared-context prefill. BudgetKV scores prompt tokens against the target model's resident input embeddings before contextual attention statistics are available, then refines a frozen lexical anchor through guarded equal-size block exchanges. This reuse of model parameters provides a query-conditioned pre-attention signal without another Transformer forward or learned encoder; under protected-block feasibility, persistent-KV token residency is at most B after selection and $B + p$ between selections. On a frozen 400-document LongBench study, resident-embedding refinement yields a statistically supported $+6.81$-point gain for Qwen3-1.7B at $B = 1024$, while the other tested model–budget cells remain inconclusive after adjustment. At 32K, the largest observed advantage is on needle retrieval rather than reasoning. On an 8GB Jetson, BudgetKV completes all six tested 8K requests at both budgets, whereas Full KV completes two; relative to successful Full-KV runs, persistent-KV footprint falls by 85.9–92.2%. These results support resident input embeddings as a useful pre-attention signal when contextual profiling is constrained by prefill memory, but benefits remain model- and task-dependent and scheduling adds overhead.

1 INTRODUCTION

Long-context inference on memory-constrained devices is not only a question of fitting model weights. Consider a repository assistant asked, "Which commit fixed the bug?" The relevant evidence may occur far from both the beginning of the repository context and the final query. Answering correctly therefore requires preserving nonlocal evidence while processing a prompt that may contain thousands of irrelevant tokens. The key–value (KV) cache grows with the processed context and competes with activations, allocator reservations, and other system processes (Kwon et al., 2023; Dao et al., 2022; Jiang et al., 2024). Evidence position and task structure further make simple recency-based retention unreliable (Bai et al., 2024; Hsieh et al., 2024; Liu et al., 2023a). On an edge device, a long prompt can thus fail during prefill even when the model weights and the intended final cache both fit in memory.

Existing KV-compression methods reduce the state retained for generation by selecting attention heavy hitters, prompt-dependent tokens, or layer-specific allocations (Zhang et al., 2023; Liu et al., 2023b; Li et al., 2024; Ge et al., 2024; Cai et al., 2024). Block-wise methods can already bound persistent KV residency during prefill (Kim et al., 2026), so the unresolved issue is not bounded execution alone. The sharper tension lies in the *selection signal*: attention-based policies may require an observation window, a patched-prompt forward, or substantial prefix computation before deciding what matters. A small final cache then says little about the state needed to compute that policy. Online sink-and-recent policies avoid full-prefix profiling (Xiao et al., 2024), but may discard the nonlocal evidence needed by the repository query. We therefore ask whether query-relevant blocks

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.

can be identified *before attention profiling*, using only representations already resident in the model, while retaining a verifiable persistent-KV contract.

We study this problem for applications in which the task query is known before shared-context prefill, including document/question, repository/question, and retrieval-augmented generation requests, even if the query is serialized after the context in the model prompt. This is application-side scheduling, not access to an unknown future token. We use *persistent KV residency* to denote KV state retained across prefill-chunk boundaries. We report it either as retained-token count or as its logical FP16 footprint; where ambiguity matters, we explicitly distinguish *persistent-KV token residency* from *persistent-KV footprint*. The setting suggests three design requirements: *pre-attention* relevance estimation, reuse of *resident representations*, and a bounded persistent-KV contract throughout chunked prefill.

BudgetKV implements these requirements by scoring the supplied query once against the model's input embedding table, without another Transformer forward, attention probe, or learned auxiliary encoder. A fixed lexical policy forms a conservative anchor; semantic scores then propose equal-size exchanges while sink and recent blocks remain protected. Selection after every prefill chunk lets distant evidence compete for a fixed budget. Under protected-block feasibility, the retained cache contains at most B tokens after selection and at most $B + p$ after appending the next chunk of at most p tokens. The contract covers persistent KV tokens, not transient workspaces, activations, allocator slack, or total process memory.

Our contributions are threefold:

1. We introduce a query-conditioned retention signal available before contextual attention statistics, obtained directly from the target language model's resident input embeddings. Unlike attention-profiling or auxiliary-encoder approaches, it requires neither another Transformer pass nor a separately learned semantic model.
2. We integrate that signal with a lexical anchor and chunk-wise selection. The resulting implementation has a conditional persistent-KV token-residency bound of B after selection and $B + p$ between selections.
3. We evaluate the representation signal, task dependence, and deployment capacity separately: a frozen 400-document study, retrieval and reasoning diagnostics through 32K, and isolated 8GB Jetson measurements.

Our results show a scoped capacity–quality trade-off. Resident-embedding refinement adds 6.81 points for Qwen3-1.7B at $B = 1024$, while the other tested model–budget cells remain inconclusive after adjustment. At 32K, the largest observed advantage is on retrieval rather than reasoning. On an 8GB Jetson, BudgetKV completes all six 8K requests at each tested budget, while Full KV completes two.

2 Related Work

KV eviction and token selection. H_2O combines attention-derived heavy hitters with recent tokens (Zhang et al., 2023); Scissorhands exploits persistence in attention-based token importance (Liu et al., 2023b). BudgetKV instead scores token identities and resident input embeddings, sharing positions across layers and KV heads. Our H2O-style comparator uses shared indices and is not the original per-layer algorithm; its prefill behavior is implementation-specific.

Attention- and query-aware compression. SnapKV selects context per head using an end-of-prompt observation window (Li et al., 2024); FastGen uses attention profiling to choose head-specific policies (Ge et al., 2024). PyramidKV and SqueezeAttention vary allocation across layers (Cai et al., 2024; Wang et al., 2024). Quest scores KV pages using the current attention query and key extrema (Tang et al., 2024); its sparse access is distinct from persistent eviction. Expected Attention analytically estimates future-query attention (Devoto et al., 2025), without necessarily requiring an extra probe. IntentKV aggregates attention assigned by inferred intention tokens, whereas BudgetKV uses a supplied task query and non-contextual input embeddings before attention profiling (Zhao et al., 2026). BudgetKV combines this parameter-resident signal with guarded equal-size exchanges and chunked persistent-KV enforcement. We do not claim query awareness as novel.

Table 1: Retention policies differ in their selection signal, signal availability, and primary resource objective. The table positions methods rather than asserting a single shared budget contract.

Method	Selection signal	Signal availability	Primary resource objective
StreamingLLM	Fixed sink and recent positions	Online, no relevance estimate	Bounded online window
SnapKV	Contextual attention in an observation window	After the window is processed	Decoding KV reduction
EpiCache	Patched-prompt attention and semantic organization	After each block-specific probe	Bounded persistent-KV residency
FastKV	Attention at a token-selective propagation layer	After full token propagation to that layer	Prefill propagation reduction
BudgetKV	Input embeddings plus a lexical anchor	Before attention, once Q is supplied	Conditional persistent-KV residency

Streaming and sink tokens. StreamingLLM preserves initial attention sinks alongside a recent window (Xiao et al., 2024). BudgetKV also protects sink and recent regions, but can retain nonlocal blocks using lexical and semantic scores. Its retention query is fixed before the shared-context prefill; the later stateful stress test does not establish robustness to evolving information needs.

Bounded and block-wise prefill. EpiCache evicts after each block under an $M + M_{\text{block}}$ cache bound and uses patched-prompt attention signals plus a lightweight encoder for semantic organization (Kim et al., 2026). BudgetKV does not claim block-wise bounded prefill as new; it instead studies whether the selection signal can come from the base model's input embeddings before attention profiling. FastKV reduces token propagation after a selective layer and decouples prefill reduction from the decoding KV budget (Jo et al., 2026); it still computes the full context through that layer, making its objective complementary to BudgetKV's persistent-residency contract.

Quantization and low-bit caches. KIVI uses asymmetric quantization of keys and values (Liu et al., 2024b); KVQuant develops low-precision representations and handling of outliers (Hooper et al., 2024). These methods reduce bytes per stored entry. BudgetKV instead reduces the number of entries and retains native floating-point KV. Cross-layer compression such as MiniCache reduces redundancy along depth rather than retaining a common subset of token positions (Liu et al., 2024a). These axes are not evaluated here; matching token counts would not constitute a memory-matched comparison against them.

Hardware-aware inference. PagedAttention addresses KV allocation and sharing (Kwon et al., 2023); FlashAttention reduces exact-attention memory traffic (Dao et al., 2022), while MInference accelerates prefill with sparse attention (Jiang et al., 2024). LLM in a flash targets parameter loading under limited DRAM (Alizadeh et al., 2023). These are distinct optimization axes. Our batch-one Jetson evaluation separates persistent KV residency from allocator peaks; eviction does not replace memory management or kernel optimization. Position-sensitive long-context behavior (Liu et al., 2023a) motivates task diagnostics, not universal retention-quality claims.

3 BUDGETKV

3.1 STATE, BLOCKS, AND RESOURCE QUANTITIES

Let $x_{0:L-1}$ be the tokenized prompt, Q the explicit task-query token IDs, and $E \in \mathbb{R}^{V \times d}$ the model's resident input embedding table. We assume Q is available to the application before shared-context prefill and remains fixed for the request; BudgetKV does not anticipate an unknown future query. At processed prefix length t, the cache contains ordered absolute positions $P_t \subseteq \{0, \dots, t-1\}$, with $N_t = |P_t|$. Define resident block $\mathcal{B}_{t,j} = \{i \in P_t : \lfloor i/b \rfloor = j\}$. Its size $n_j = |\mathcal{B}_{t,j}|$ can be smaller than block width b; positions need not be contiguous. All layers and KV heads use the same selection indices.

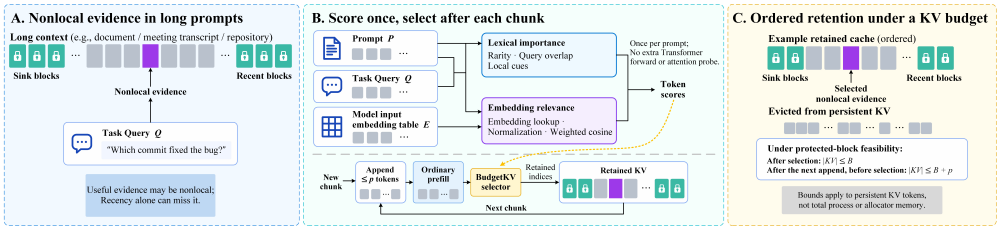

Figure 1: Overview of BudgetKV. Scores are computed once and selection follows each prefill chunk. Input embeddings support semantic scoring without an extra forward; protected blocks cannot be victims, and new exchanges require equal size, lexical safety, and higher utility. The conditional contract covers persistent KV residency, not whole-process memory.

We distinguish requested budget B, realized final-prefill cache size F, and sampled peak persistent-KV token residency $P_{\max}$. The first is a target, the second an observed count after final prefill selection, and the third the largest retained-token count immediately after a prefill forward or eviction. Neither F nor $P_{\max}$ measures all simultaneously live allocations. Peak CUDA-reserved memory additionally reflects weights, buffers, and allocator behavior, and still does not measure total system-wide unified-memory use.

At each selection, the implementation clamps the local budget to $B_t = \max(1, \min(B, N_t))$. Effective sink length s_t is at most B_t and t; recent length r_t is at most $B_t - s_t$ and $t - s_t$. It protects positions $i < s_t$ or $i \geq \max(s_t, t - r_t)$. A block is protected if *any* resident position is protected. This block closure determines budget feasibility; $s_t + r_t$ alone does not.

3.2 Frozen lexical anchor and resident-embedding scoring

The lexical anchor combines prompt-frequency rarity, query-token overlap, digit and uppercase cues, punctuation-only and short-token cues. It is a conservative engineering policy rather than the paper's representation contribution. A local max pool spreads importance to neighboring tokens; a small recency term breaks near ties. Denote the resulting token score by a_i. For a resident block j, its lexical importance is

$$\ell_j^{\text{raw}} = 0.7 \max_{i \in \mathcal{B}_{t,j}} a_i + 0.3\, n_j^{-1} \sum_{i \in \mathcal{B}_{t,j}} a_i. \tag{1}$$

The constants and exact token predicates are specified in Appendix B.2 and implemented in `src/budgetkv/selection.py` in the accompanying artifact. This score is a heuristic risk estimate, not a calibrated probability of answer relevance.

The semantic scorer deduplicates nonnegative query IDs and keeps at most $q = 32$, preferring IDs rarer in the prompt when truncation is necessary. Let $f(v)$ be the prompt frequency of ID v. Query weights and token scores are

$$w_v = \frac{1/\sqrt{\max(1, f(v))}}{\max_{u \in Q'} 1/\sqrt{\max(1, f(u))}}, \qquad z_i = \max\left(0, \max_{v \in Q'} w_v \langle \widehat{E}_{x_i}, \widehat{E}_v \rangle\right), \tag{2}$$

where Q' is the retained query set and $\widehat{E}_v$ is the row normalized in FP32 with norm floor 10^{-6}. Global min–max normalization gives semantic token scores e_i; a constant range gives zero scores. The same max/mean pooling yields semantic block importance. Lexical and semantic block values are separately normalized across the current blocks, then combined as $u_j = (1 - \alpha)\ell_j + \alpha e_j$ with $\alpha = 0.5$.

Equation 2 uses embedding lookups and matrix products, not contextual hidden states. It introduces no extra Transformer forward or learned parameters. Scoring runs once in 256-token batches and stores its output on the central processing unit (CPU). These representations are non-contextual and sensitive to tokenization. The primary semantic comparison holds the lexical policy fixed, isolating only this refinement signal.

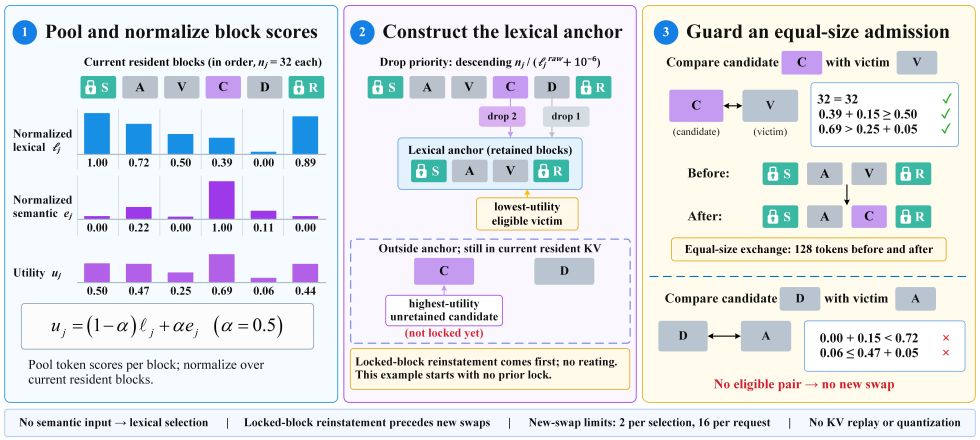

Figure 2: Guarded equal-size exchange refines the lexical anchor without increasing retained tokens. In this illustration, C replaces V after size, lexical, and utility checks; D fails. The star marks an admission lock, which can be reinstated without re-gating but cannot recover evicted KV.

3.3 Guarded equal-size replacement

For the current resident blocks, a lexical anchor is formed by dropping unprotected blocks in descending order of $n_j/(\ell_j^{\text{raw}} + 10^{-6})$ until the count is at most B_t or no eligible blocks remain. This internal anchor is not a second proposed method. It defines a conservative retention set for the semantic refinement.

Let c be an unretained candidate and v a retained, unprotected, unlocked victim. A new semantic admission requires

$$n_c = n_v, \qquad \ell_c + \varepsilon \geq \ell_v, \qquad u_c > u_v + \delta, \quad \varepsilon = 0.15,\ \delta = 0.05. \tag{3}$$

Candidates are visited in decreasing utility, victims in increasing utility, with absolute block IDs resolving ties. The implementation permits at most two new swaps per selection call and sixteen per request. Admitted block IDs are locked to reduce churn. On later anchor recomputations, an available locked block can be reinstated by replacing an equally sized, unlocked, unprotected block. *Reinstatement does not reapply the lexical or utility gate.* Thus a local utility improvement for new admissions is not a monotonicity guarantee for the entire online execution.

Algorithm 1 in the appendix gives the complete prefill loop.

Fallback and irreversibility. Empty caches, caches already within budget, or absent semantic-score input return the lexical selection without semantic refinement. A missing or empty query produces zero semantic scores in the embedding scorer; this is a zero-signal path, not permission to fabricate a query. The lexical scorer separately uses the last 128 prompt IDs only when its query argument is `None`. Missing coverage or non-finite semantic values raise an error, not a silent fallback. No eligible equal-size pair means no new swap. Evicted KV is never replayed or recovered: locks refer only to blocks still present in the current cache. The ordinary generation path continues from the retained states, so exact floating-point storage does not imply equivalence to Full-KV inference.

3.4 Budget properties and implementation cost

Let H_t be the number of resident tokens in the union of protected blocks. If $H_t \leq B_t$, dropping unprotected blocks can reach the local budget. If $H_t > B_t$, preserving all protected blocks and satisfying the budget are incompatible; the current implementation preserves protection and may exceed B_t. Budget enforcement is therefore conditional, not an unconditional hard cap. Rejecting infeasible requests or shrinking protection would require an explicit policy change; neither is claimed for these runs.

The following invariant is not presented as a new block-wise prefill principle; it specifies the resource contract used by BudgetKV and audited in the deployment experiments.

Table 2: Three strongest supported findings, each under its own protocol. The boundary column prevents cross-protocol generalization.

Evidence	Main observation	Boundary
Frozen LongBench	Qwen gains 6.81 points at $B = 1024$; adjusted 95% confidence interval (CI) $[2.37, 11.45]$	Other three adjusted intervals include zero
32K needle retrieval	BudgetKV scores 66.25/41.25 versus 10.00/2.50 without semantics (Qwen/Llama)	Full KV remains stronger
8GB Jetson, 8K	BudgetKV completes 6/6 at each budget versus 2/6 for Full KV; persistent-KV footprint is 70/126 versus 896 MiB	Full-KV mean uses two successes

Proposition 1 (Conditional persistent-KV token-residency invariant)**.** *Suppose protected-block feasibility holds at every selection call. Equal-size swaps preserve the lexical-anchor cardinality. If each prefill forward appends at most p tokens and is followed by selection, persistent-KV token residency is at most B after selection and $B + p$ between selections.*

The proof is a cardinality argument and induction, given in Appendix B. Whole-block eviction can undershoot the requested budget. Equal-size replacement does not prove that a separately executed lexical-only run has the same final count: the two runs may have different resident histories. Exact-token comparisons in the experiments are checked from run records, not inferred from this proposition. New gated admissions increase the current additive surrogate by more than δ per swap; no task-quality bound follows.

For D layers, h_{kv} KV heads, head width d_h, and β bytes per scalar, the logical cache for n retained tokens at batch size one occupies

$$M_{\mathrm{KV}}(n) = 2Dh_{\mathrm{kv}}d_h\beta n. \tag{4}$$

The main method uses native 16-bit floating-point (FP16) KV ($\beta = 2$), without quantization scales or low-bit reconstruction. Allocator reservations, transient gather copies, activations, and the model parameters are outside this formula.

Parameter reuse avoids an extra Transformer pass, not scheduler work: scoring costs $O(Lqd)$ with $q \leq 32$, while repeated pooling, selection and KV gathers add overhead. Appendix B.4 bounds and profiles the current implementation.

Prefill is not decoding. The proposition applies to chunked prefill, not automatically to generated tokens. Let D_g denote the resident cache count after processing g generated tokens. A growing-cache decoder can have $D_g > B$ even when its final prefill count satisfies $F \leq B$. In the desktop diagnostic, all compared methods use the same growing-cache decoder; in the Jetson study, each method uses its recorded native decode policy. Thus exact prefill matching and equal decode resource usage are different contracts. We report them separately rather than extending the prefill bound to either protocol without verification.

4 Experiments

4.1 Questions, protocols, and statistical units

We evaluate Qwen3-1.7B and Llama-3.2-1B-Instruct in FP16 at batch size one (Yang et al., 2025; Meta, 2024). An RTX 4090 D hosts controlled quality diagnostics; every edge-resource claim comes from the Jetson study. Evidence is organized around three claims and is never pooled across hardware or decode policies. Table 2 gives the shortest verification path.

The frozen confirmation set contains 400 documents from five LongBench tasks: 64 2WikiMQA, 74 HotpotQA, 123 MuSiQue, 58 PassageRetrieval-en, and 81 TriviaQA examples (Bai et al., 2024). Its 3200 primary requests compare BudgetKV with the same lexical policy but no semantic refinement at $B = 512, 1024$ on both models. Comparator and no-gate runs reuse these documents and are secondary evidence.

Scores are task-macro means on a 0–100 scale using the archived scorer, not a claim of official leaderboard equivalence. Confirmation-set differences pair identical documents and inputs. We

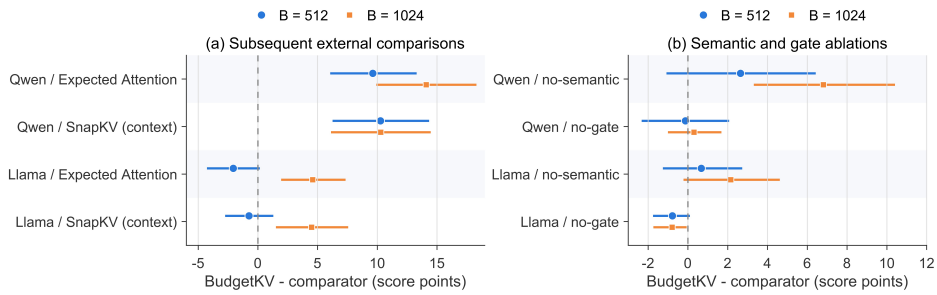

Figure 3: The family-adjusted interval excludes zero only for Qwen at $B = 1024$. Circles/squares denote 512/1024-token budgets; all 400-document pairs match final-prefill counts. Whiskers are nominal paired 95% intervals. External and no-gate contrasts are secondary, and SnapKV is context-only.

recompute 20,000 task-stratified paired document bootstrap draws (seed 20260913); figures show nominal 95% intervals, and the appendix also gives Bonferroni intervals for four semantic, four gate, and eight external contrasts. Budgets are not treated as independent documents. Long-context depth cells share source seeds and are descriptive, not independent significance tests. Appendix C specifies scoring and provenance.

4.2 Does the resident-embedding signal improve fixed-budget retention?

At 1024 tokens, Qwen's confirmation-set task-macro score increases from 30.17 without semantic refinement to 36.98 with BudgetKV: $+6.81$ points, nominal 95% CI $[3.31, 10.41]$, and family-adjusted CI $[2.37, 11.45]$. At 512 tokens, its $+2.64$ interval includes zero. Llama gains 0.67 and 2.15 points at the two budgets, with both intervals including zero. Thus the confirmation supports one model–budget operating point, not a backbone-independent effect (Figure 3). All 1600 semantic pairs have identical realized final-prefill token counts.

The no-gate contrasts do not establish a quality benefit of the lexical guard after family adjustment. For example, Llama at 1024 has a nominal negative difference of -0.79 points, but its adjusted interval includes zero. We retain the gate as an explicit constraint on new semantic admissions, not as a proven accuracy improvement. Removing it is a secondary ablation, not a basis for retuning the frozen method.

Selection mechanism. Every primary BudgetKV request records at least one semantic admission. Mean new admissions are 4.62/4.75 for Qwen and 4.48/4.79 for Llama at 512/1024, below the cap of 16. These records establish that semantic refinement changes selection, but without oracle relevance labels they do not establish evidence recall or explain every quality change.

4.3 Where does the signal help: retrieval or reasoning?

The length study uses RULER-derived generators (Hsieh et al., 2024): multi-value needle retrieval, HotpotQA-based question answering, and four-hop variable tracking at exact 8K/16K/32K inputs. This is an adapted diagnostic, not the official full RULER protocol. Each task has 20 items per length and model. Needle items cross four source seeds with five target sentence-depth percentages (10,30,50,70,90); question answering and tracking do not have controlled depth bins. Full KV, BudgetKV, no-semantic, and StreamingLLM run at all lengths; query-visible SnapKV runs only at 8K. Longer SnapKV cells were excluded after a recorded 16K implementation-level allocation failure, not scored as failures or replaced with another algorithm.

At 32K and $B = 1024$, Qwen needle retrieval is 66.25 for BudgetKV, 10.00 for no-semantic, 0 for StreamingLLM, and 80.00 for Full KV. Llama scores 41.25, 2.50, 0, and 76.25. The advantage over lexical-only retention is consistent with retaining nonlocal evidence; the remaining Full-KV gap shows substantial information loss. The result does not extend uniformly to reasoning: at 32K, BudgetKV/no-semantic question-answering scores are 30/25 for Qwen and 25/30 for Llama, while tracking scores are identical (6/6 and 0/0). We therefore make a retrieval-specific claim.

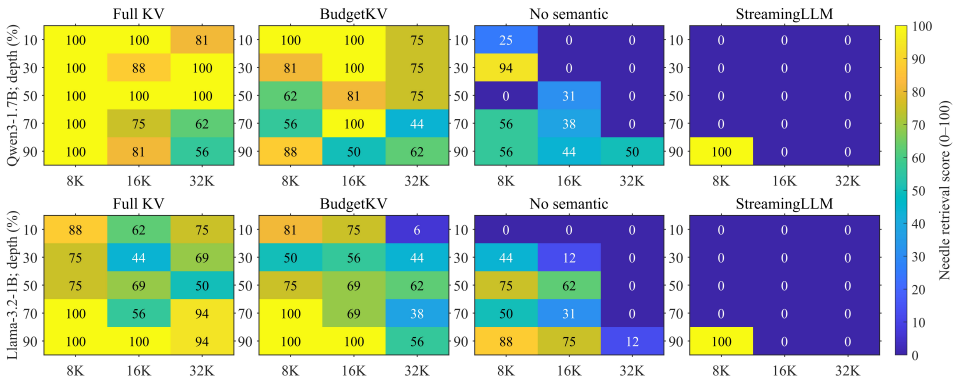

Figure 4: Semantic refinement preserves substantially more nonlocal needle evidence than lexical-only retention at $B = 1024$. Each cell averages four repeated source seeds; sentence depth is not exact token depth. This adapted diagnostic is not a full long-context benchmark; both budgets and all tasks appear in Appendix C.

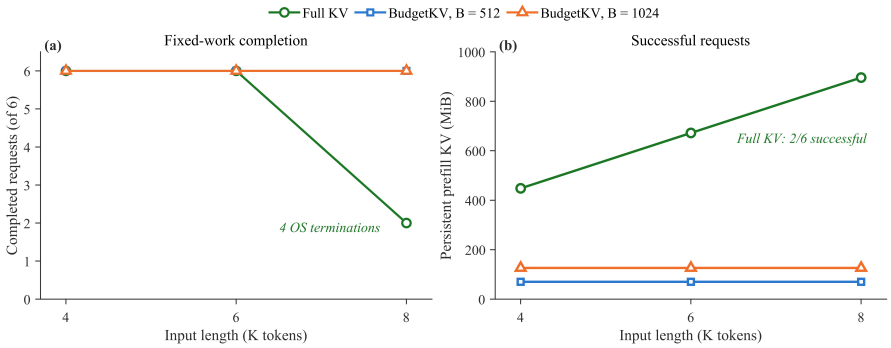

Figure 5: BudgetKV completes all tested 8K fixed-workload requests, while Full KV completes two of six. Each point covers the same six documents and 64 forced output tokens. Panel (b) averages successful requests only; the Full-KV 8K point therefore has two observations.

4.4 Does bounded residency translate into edge capacity?

The capacity study uses six hash-selected documents, exact 4K/6K/8K inputs, isolated processes, and a 64-token forced decode. Full KV and BudgetKV complete all 4K and 6K requests. At 8K, BudgetKV completes $6/6$ at both $B = 512$ and 1024; Full KV completes $2/6$, while four workers receive SIGKILL (signal 9) after model loading. Among the two successful Full-KV requests, persistent-KV footprint is 896 MiB, versus 70/126 MiB for BudgetKV, reductions of 92.2/85.9%. Peak CUDA-reserved memory is 5024 versus 3456/3516 MiB, reductions of 31.2/30.0%. This observed device-capacity result does not establish a deterministic Full-KV out-of-memory boundary. Appendix E reports every cell.

Measured resource cost. The broader systems study contains 576 isolated processes over 12 documents, exact 4K/8K inputs, two budgets, four methods, and three repetitions. At $B = 1024$, BudgetKV reduces median peak CUDA-reserved memory by 488 MiB versus Expected Attention and 520 MiB versus SnapKV, but adds 1.51 and 2.05 seconds, respectively; paired 95% intervals exclude zero. Against StreamingLLM, BudgetKV uses 2 MiB more and adds 1.75 seconds. Appendix E reports the resource contract and measured overhead. These native decode paths are not an equal-cache speed comparison.

4.5 Comparator and stateful boundaries

In the secondary matched comparison, Qwen exceeds archived context-only SnapKV and Expected Attention configurations at both budgets; Llama's positive paired intervals appear at 1024, not 512

(Figure 3). Because those implementations use different decode policies, we also test 80 exposed documents with a common growing-cache decoder. Query-visible SnapKV is stronger in all four model–budget cells (Appendix Figure 9); query visibility and observation-window length change together, so this compares procedures rather than isolating one factor. The result prevents a universal quality claim.

Stateful boundary. SharedContextBench (SCBench) is a boundary test, not primary efficacy evidence (Li et al., 2025). In a five-turn Qwen run, BudgetKV scores 5.83/5.86 versus 8.66 for Full KV, while averaging 56/112 MiB live KV versus 1.83 GiB. These low absolute scores support bounded execution, not robust multi-turn quality; Appendix F reports the result.

4.6 Deployment regime and evidence synthesis

The experiments support three linked but distinct claims. First, resident embeddings can alter retention usefully without attention profiling: the fixed-policy semantic contrast is positive for Qwen at $B = 1024$, and its largest repeatable effect is distant needle retrieval. This evidence is about a representation signal, not a universal accuracy gain. The inconclusive Llama contrast, unchanged tracking cells, and query-visible SnapKV result locate the boundary: contextual attention remains more informative when its observation pass is affordable.

Second, the selection loop converts that signal into an executable resource contract. The proof establishes conditional token cardinality, and the measured prefill peaks agree with the $B + p$ invariant. This claim is stronger than reporting a small cache after prefill, but narrower than bounding total device memory. In particular, the implementation may transiently hold gather copies and allocator reservations that the token proof does not count.

Third, the Jetson study tests whether this narrower contract changes deployment outcomes. It does on the measured 8K workload: every BudgetKV request completes, whereas four Full-KV workers are terminated. The paired systems data also show that capacity is not free; pooling, selection, and gathers add latency. Thus the device result should be read as evidence of a feasible operating point, not as a kernel-speed claim or a deterministic memory threshold.

These distinctions suggest a practical decision rule. BudgetKV is most appropriate when the task query is available before shared-context prefill, nonlocal retrieval matters, and computing an attention-derived policy is itself difficult under the device budget. An attention-based selector is preferable when full profiling is feasible and contextual discrimination dominates. For evolving multi-turn queries, the method must re-prefill the original context with the new query or use another online policy; the old cache cannot recover evicted evidence.

5 Limitations and Discussion

BudgetKV has three main boundaries. *Representation:* input embeddings are non-contextual, shared indices cannot express layer/head-specific importance, and the lexical guard has no confirmed accuracy benefit. *Workload:* the query must be known before prefill and remain fixed; query-visible SnapKV is stronger on the exposed diagnostic, while reasoning and stateful quality are weak. *Systems:* the bound covers persistent KV, not total memory, and scheduler/gather work adds latency. Evidence spans two small models and one 8GB device; scale transfer, combined compression, and sensitivity remain open. The SIGKILL events are deployment outcomes, not a deterministic Full-KV limit.

6 Conclusion

We have shown that resident input embeddings can guide pre-attention retention under tight prefill budgets. Chunk-wise selection provides a conditional persistent-KV contract, while the experiments identify retrieval-focused quality and edge-capacity gains. Our results motivate evaluating KV-compression methods not only by final cache size, but also by the representations and intermediate states required to compute the retention policy. BudgetKV provides one such operating point: it trades contextual discrimination for a pre-attention signal derived from representations already resident in the model.

AI USE STATEMENT

Generative AI tools were used to assist with language polishing and literature retrieval during manuscript preparation. AI-assisted literature discovery was used only to identify potentially relevant references, and all cited works were independently checked by the authors. All research ideas, methodological design, mathematical derivations, experimental implementation, data analysis, results, and conclusions were produced and verified by the authors. The authors take full responsibility for the content of the paper.

REPRODUCIBILITY STATEMENT

The supplementary ZIP provides the BudgetKV implementation, experiment configurations and frozen sample manifests, prepared-input hashes, compact request-level records and plotting inputs, analysis and validation scripts, and MATLAB sources for the reported figures. Model weights are not redistributed; benchmark text is limited to the prepared Jetson payload needed to audit the device protocol, with upstream sources and checksums documented. The 400-document confirmation matrix was fixed before manuscript revision, and later exposed diagnostics were not used to retune the method. Appendix G maps each headline claim to the corresponding artifact files.

REFERENCES

Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, Karen Khatamifard, Minsik Cho, Carlo C. Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. LLM in a flash: Efficient large language model inference with limited memory. *arXiv preprint arXiv:2312.11514*, 2023.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://arxiv.org/abs/2308.14508.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

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.

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.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. *arXiv preprint arXiv:2401.18079*, 2024.

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.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. *arXiv preprint arXiv:2407.02490*, 2024.

Dongwon Jo, Jiwon Song, Yulhwa Kim, and Jae-Joon Kim. FastKV: Decoupling of context reduction and KV cache compression for prefill-decoding acceleration. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 32167–32186. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1610. URL https://aclanthology.org/2026.findings-acl.1610/.

Minsoo Kim, Arnav Kundu, Han-Byul Kim, Richa Dixit, and Minsik Cho. EpiCache: Episodic KV cache management for long-term conversation on resource-constrained environments. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2509.17396.

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 *ACM 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 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. *arXiv preprint arXiv:2404.14469*, 2024.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. MiniCache: KV cache compression in depth dimension for large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024a. URL https://proceedings.nips.cc/paper_files/paper/2024/hash/fd0705710bf01b88a60a3d479ea341d9-Abstract-Conference.html.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023a.

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*, volume 36, 2023b.

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

Meta. Llama 3.2 model card, 2024. URL https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47901–47911, 2024. URL https://proceedings.mlr.press/v235/tang24l.html.

Zihao Wang, Bin Cui, and Shaoduo Gan. SqueezeAttention: 2D management of KV-cache in LLM inference via layer-wise optimal budget. *arXiv preprint arXiv:2404.04793*, 2024.

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

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*, volume 36, 2023.

Liang Zhao, Xiaocheng Feng, Weihong Zhong, Lei Huang, Kun Zhu, Baoxin Wang, Dayong Wu, Guoping Hu, Ting Liu, and Bing Qin. Question tells you where the answer is: Intention-aware long-context KV cache compression. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27153–27169. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1250. URL https://aclanthology.org/2026.acl-long.1250/.

Appendix

TABLE OF CONTENTS

A Supplement Scope and Evidence Guide **13**

B Method Contract and Proofs **13**
B.1 Query availability and scope 13
B.2 Exact lexical anchor 13
B.3 Feasibility and proof of Proposition 1 14
B.4 Prefill algorithm and cost 14

C Frozen Quality Confirmation **15**

D Long-Context Task Diagnostic **16**

E Edge Deployment Evidence **17**
E.1 Common-decode quality diagnostic 17
E.2 Fixed-workload capacity 17

F Boundary Diagnostics **18**

G Reproducibility Map **19**

A Supplement Scope and Evidence Guide

The supplement follows the paper's three claims. Section B specifies the query contract, persistent-KV invariant, algorithm, and fixed configuration. Section C gives the complete controlled semantic contrast; Section D separates retrieval from reasoning across length; and Section E reports edge quality, capacity, resource, and profiling measurements. Section F records the strongest comparator and stateful limitations, and Section G maps each headline claim to its archived evidence. Historical development tables and superseded recovery narratives are omitted because they do not change the submitted claims.

Table 3: Evidence inventory. Counts exclude smoke tests and failed attempts; repeated process runs are not treated as independent documents.

Study	Documents/items	Runs	Role
Frozen semantic confirmation	400	3200	Primary controlled contrast
Secondary matched comparisons	Same 400	4800	External and no-gate context
Length/task diagnostic	60 per length	2760	8K, 16K, 32K behavior
Query-visible diagnostic	80	960	Common-decode boundary
SCBench follow-up	24 conversations/cases	1104	Stateful and matched boundary
Jetson systems	12	576	Native deployment cost
Jetson fair diagnostic	12	384	Common-decode device check
Jetson capacity	6	54	Fixed-workload completion
Jetson profiling	2	72	Component overhead

Table 4: Claim calibration. Each result answers one question and leaves a stronger extrapolation unsupported.

Observation	Supported interpretation	Unsupported extrapolation
Qwen $+6.81$ at $B = 1024$	Resident embeddings add useful information to the fixed lexical policy	Backbone-independent accuracy improvement
32K needle advantage	The signal helps retain distant, query-linked evidence	General reasoning or state-tracking improvement
Jetson 8K completion	Bounded residency enables the workload in the measured setup	Deterministic Full-KV OOM threshold or faster execution
Query-visible SnapKV lead	Contextual attention can be stronger when its observation pass is feasible	BudgetKV is dominated under the prefill constraint it targets

B Method Contract and Proofs

B.1 Query availability and scope

BudgetKV assumes that the application receives a shared context and a task query before model prefill. The query may be serialized after the context in the model input, but its token IDs are available to the selector before the first context chunk. This contract covers document/question, repository/question, and retrieval-augmented generation requests. It does not anticipate a future user turn or recompute semantic scores when the information need changes.

The explicit query is separately tokenized without special tokens. A missing or empty query produces zero embedding scores and therefore no semantic refinement. The lexical anchor then supplies the fallback retention policy; the system does not synthesize a query from future tokens. All layers and KV heads use the same retained absolute positions.

B.2 Exact lexical anchor

For prompt token ID x_i, let $f(x_i)$ be its prompt frequency and let $\mathcal{Q}$ be the set of explicit lexical-query token IDs. The pre-pooling token score is the sum of all applicable increments in Table 5; predicates that do not hold contribute zero.

Table 5: Frozen token predicates and additive increments in the lexical anchor.

Token predicate	Increment
$x_i \in \mathcal{Q}$	$2.4/\sqrt{\max(1, f(x_i))}$
Decoded token contains the literal `KV`	5.0
Decoded token contains a digit	3.0
Decoded token contains an uppercase ASCII letter	1.4
Stripped token is nonempty and contains no ASCII alphanumeric	0.2
Stripped token length is one or two characters	0.1

The implementation then applies one-dimensional max pooling with radius $\lfloor b/2 \rfloor$ (kernel width at most the sequence length) and adds a linear recency ramp from 0 to 0.01. Section 3.2 pools these values within each block using the stated 0.7 max / 0.3 mean rule. If the lexical scorer receives `None` rather than an explicit query argument, it uses the last 128 prompt IDs as its query set; an explicitly empty query does not trigger this fallback. These predicates and coefficients are fixed in every controlled semantic comparison. The archived implementation is `src/budgetkv/selection.py`.

B.3 Feasibility and proof of Proposition 1

Let the resident cache be partitioned into disjoint blocks $\mathcal{J}$, with $1 \le n_j \le b$. Write $\mathcal{H} \subseteq \mathcal{J}$ for protected blocks and $H = \sum_{j \in \mathcal{H}} n_j$. If $H \le B_t$, removing every unprotected block leaves at most B_t tokens, so the lexical eviction loop can terminate within budget. If $H > B_t$, preserving all protected blocks and meeting the budget are incompatible. Because a boundary token protects its whole resident block, clamping only sink and recent token-region lengths does not eliminate this infeasible case.

Proof. Let S be a retained set and let candidate $c \notin S$ and victim $v \in S$ satisfy $n_c = n_v$. Their exchange gives $|S'| = |S| - n_v + n_c = |S|$. Every new admission and lock reinstatement preserves the lexical-anchor count, while protected blocks are never victims. Under protected-block feasibility, the anchor has at most $B_t \le B$ tokens. Initially the persistent cache is empty. If the preceding selection leaves at most B tokens, an ordinary forward appending $h \le p$ tokens returns at most $B + h \le B + p$ tokens; the next feasible selection restores a count of at most B. Induction over chunks proves the proposition, including a partial final chunk. □

The proof counts persistent KV tokens. It excludes temporary old/new caches during concatenation or gather, attention activations, scorer workspaces, allocator slack, parameter memory, and whole-process or system memory. A block removal may also undershoot the requested budget by fewer than b tokens.

B.4 Prefill algorithm and cost

Algorithm 1 BudgetKV prefill (high-level form)

Require: Prompt x, application-supplied query Q, budget B, block size b, chunk size p
1: Compute frozen lexical scores and resident-embedding scores once
2: $P \leftarrow \emptyset$, locked blocks $\mathcal{K} \leftarrow \emptyset$, admissions $A \leftarrow 0$
3: **for** each prompt chunk of size $h \le p$ **do**
4: Append ordinary forward KV and update absolute positions P
5: Build blocks and sink/recent protection; form lexical anchor under B_t
6: **if** $|P| > B$ and semantic scores are nonempty **then**
7: Pool block scores; restore eligible locks by size-matched exchange
8: Admit at most two new candidates satisfying Eq. 3, while $A < 16$
9: **end if**
10: Gather exact FP16 KV at retained indices in original positional order
11: **end for**
12: Apply final selection if required and return retained KV

Embedding scoring costs $O(Lqd)$ with $q \leq 32$. With R selection calls, at most N resident tokens and M resident blocks per call, a conservative bound for the current Python scheduler is

$$O\big(Lqd + R\left[L + N\log N + M^2 + KM\log M\right]\big), \qquad K \leq 16. \tag{5}$$

KV gathering additionally copies $O(Dh_{\mathrm{kv}}d_hN)$ scalars per selection. Reusing parameters therefore removes an auxiliary model or Transformer pass, not pooling, scheduling, and movement overhead.

Table 6: Frozen BudgetKV configuration used in the reported experiments.

Parameter	Value
Block size b / prefill chunk p	32 / 128 tokens
Embedding chunk / query cap q	256 / 32 tokens
Semantic weight α	0.50
Semantic / lexical margins (δ, ε)	(0.05, 0.15)
New swaps per call / request	2 / 16
Sink / recent at $B = 512$	32 / 128
Sink / recent at $B = 1024$	64 / 256
KV representation / precision	shared-index exact KV / FP16
Batch size / attention backend	1 / SDPA

The lexical coefficients, margins, and admission cap remained fixed in the confirmation study. Their individual sensitivity and transfer to other model scales are not established.

C Frozen Quality Confirmation

The confirmation uses 400 documents across five LongBench tasks, with every task receiving equal weight in the macro score. Each BudgetKV/no-semantic pair has the same effective input and realized final-prefill count. Differences use 20,000 task-stratified paired document-bootstrap draws. The adjusted intervals below use Bonferroni correction within the four semantic contrasts. The no-semantic condition keeps the same frozen lexical policy and disables only resident-embedding refinement.

Table 7: Confirmation task-macro means (400 documents per entry).

Model	Method	$B = 512$	$B = 1024$
Qwen3-1.7B	BudgetKV	30.28	36.98
	No semantic	27.63	30.17
	Query-hidden SnapKV	20.01	26.69
	Expected Attention	20.65	22.88
	Full KV reference	46.76	
Llama-3.2-1B-Instruct	BudgetKV	12.11	19.77
	No semantic	11.44	17.62
	Query-hidden SnapKV	12.86	15.27
	Expected Attention	14.17	15.18
	Full KV reference	21.02	

Table 8: BudgetKV minus no-semantic: paired score differences.

Model	B	Difference	95% CI	Family-adjusted CI
Qwen3-1.7B	512	2.64	$[-1.08, 6.42]$	$[-2.13, 7.52]$
Qwen3-1.7B	1024	6.81	[3.31, 10.41]	[2.37, 11.45]
Llama-3.2-1B	512	0.67	$[-1.26, 2.73]$	$[-1.74, 3.32]$
Llama-3.2-1B	1024	2.15	$[-0.22, 4.62]$	$[-0.91, 5.28]$

The Qwen $B = 1024$ cell supports the incremental semantic signal. The other three adjusted intervals include zero, so the evidence does not establish a model- or budget-independent improvement. The Full-KV rows are unmatched desktop references.

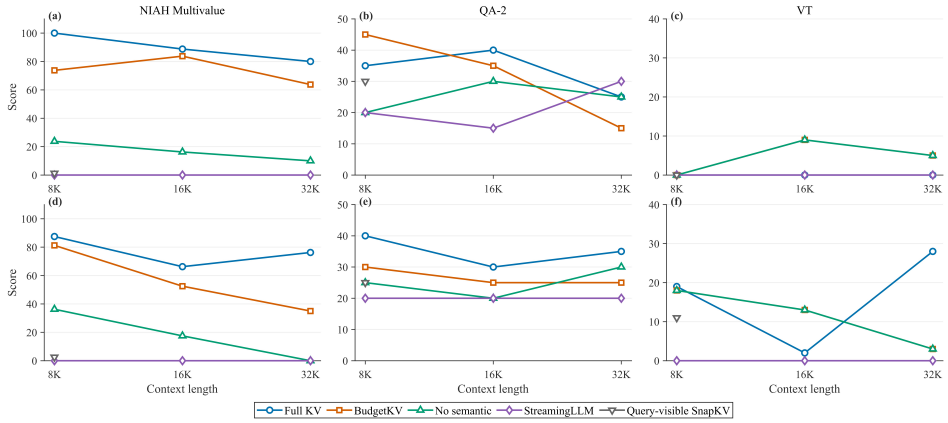

Figure 6: Complete three-task length diagnostic at $B = 512$. Query-visible SnapKV is observed only at 8K; no values are imputed at 16K/32K. Full KV is an uncompressed reference.

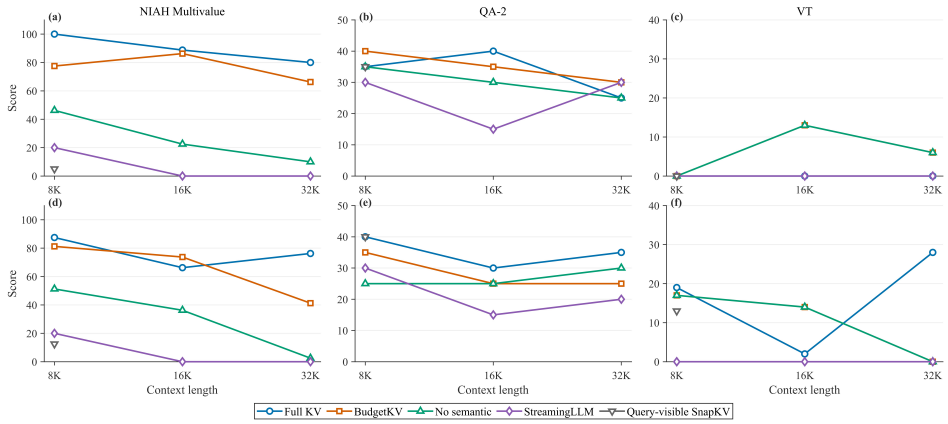

Figure 7: Same diagnostic at $B = 1024$. Resident-embedding refinement most consistently helps needle retrieval; QA and tracking do not show a uniform benefit.

D Long-Context Task Diagnostic

The adapted RULER diagnostic uses multi-value needle retrieval, HotpotQA-based question answering, and four-hop variable tracking at exact 8K, 16K, and 32K lengths. It has 20 items per task/model/length; repeated source clusters across lengths are not independent samples. Full KV, BudgetKV, no-semantic, and StreamingLLM run at every length. Query-visible SnapKV is shown only at 8K because its measured implementation failed allocation at 16K; omitted cells are not scored as zero. This is not an official RULER score.

Table 9: Representative 32K results at $B = 1024$ (score, 0–100).

Model	Task	BudgetKV	No semantic	StreamingLLM	Full KV
Qwen3-1.7B	Needle retrieval	66.25	10.00	0.00	80.00
	Question answering	30.00	25.00	30.00	25.00
	Variable tracking	6.00	6.00	0.00	0.00
Llama-3.2-1B	Needle retrieval	41.25	2.50	0.00	76.25
	Question answering	25.00	30.00	20.00	35.00
	Variable tracking	0.00	0.00	0.00	28.00

E EDGE DEPLOYMENT EVIDENCE

The Jetson studies use Qwen3-1.7B in FP16 at batch size one on an 8GB Jetson Orin Nano. The recorded configuration uses MAXN_SUPER, default dynamic voltage and frequency scaling, active file swap and zram, and unlocked clocks. These are deployment conditions, not locked-clock kernel benchmarking conditions.

E.1 COMMON-DECODE QUALITY DIAGNOSTIC

The exposed 12-document diagnostic uses identical realized prefill counts and a common growing-cache decoder. It supports on-device feasibility and quality under one controlled path, but its size and exposed selection do not make it an independent confirmation set.

Table 10: Jetson common-decode diagnostic scores (0–100; 12 exposed documents).

Method	$B = 512$	$B = 1024$
BudgetKV	26.88	43.34
No semantic	1.81	26.84
StreamingLLM	2.04	2.81
Query-visible SnapKV	9.26	25.57

E.2 FIXED-WORKLOAD CAPACITY

Table 11: Fixed-workload capacity cells. Each entry reports successes/6; persistent-KV footprint / peak CUDA-reserved memory in MiB. The Full-KV 8K memory mean uses only its two completed requests.

Input	Full KV	BudgetKV, $B = 512$	BudgetKV, $B = 1024$
4K	6/6; 448 / 4034	6/6; 70 / 3456	6/6; 126 / 3516
6K	6/6; 672 / 4504	6/6; 70 / 3456	6/6; 126 / 3516
8K	2/6; 896 / 5024	6/6; 70 / 3456	6/6; 126 / 3516

Every request uses the same six hash-selected documents and forces 64 output tokens. The four failed 8K Full-KV workers exit with signal 9 after model loading. This is an observed completion outcome with active swap, not a controlled CUDA out-of-memory event or proof that every Full-KV 8K request fails.

Table 12: BudgetKV-minus-comparator effects at 8K over 12 document-level repetition medians; brackets are paired 95% bootstrap intervals.

B	Comparator	Reserved-memory difference (MiB)	Request-time difference (s)
512	Expected Attention	-530 $[-567, -530]$	-0.30 $[-0.64, 0.20]$
512	Query-hidden SnapKV	-466 $[-471, -466]$	0.63 $[0.52, 0.89]$
512	StreamingLLM	2 $[2, 2]$	0.96 $[0.50, 1.22]$
1024	Expected Attention	-488 $[-488, -488]$	1.51 $[1.30, 1.67]$
1024	Query-hidden SnapKV	-520 $[-520, -519]$	2.05 $[1.84, 2.28]$
1024	StreamingLLM	2 $[2, 2]$	1.75 $[1.42, 1.96]$

The native decode paths differ, so these values characterize the implemented systems rather than an equal-cache speed contest. Reserved CUDA memory includes allocator behavior and is not total unified-memory use.

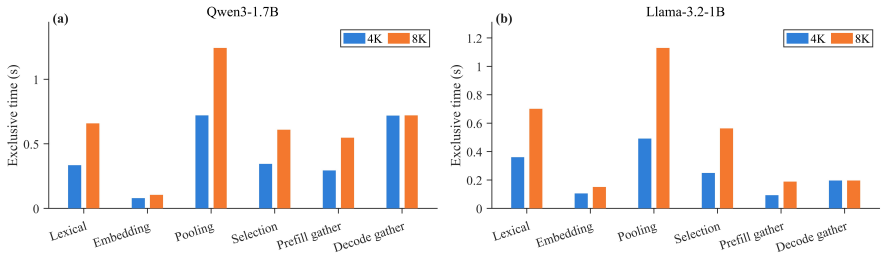

Figure 8: Median exclusive BudgetKV component times in the instrumented Jetson profile. Pooling, selection, and KV movement dominate the embedding lookup; instrumentation perturbs execution and does not measure production kernel speed.

For Qwen at 8K, embedding scoring takes 0.10 seconds, versus 1.24 for pooling, 0.61 for selection, 0.55 for prefill gathering, and 0.72 for decode gathering. This profile identifies optimization targets and explains why avoiding an extra Transformer forward does not imply a latency improvement.

F Boundary Diagnostics

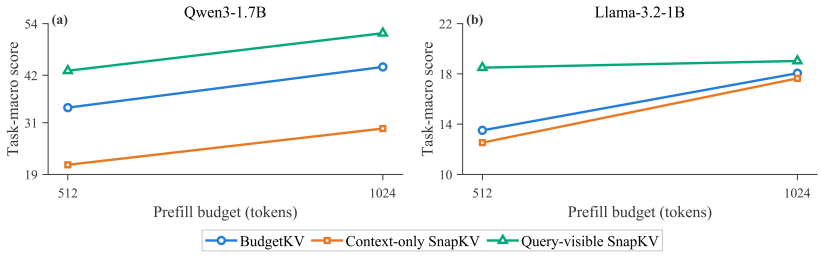

Figure 9: Query-visible SnapKV is stronger in all four exposed LongBench cells under a common growing decoder and matched prefill KV bytes. Visibility and observation-window length change together; this is a procedural comparison, not a single-factor ablation.

The stateful diagnostic uses 24 SCBench conversations and five sequential turns after one shared-context prefill. Its low scores make it a boundary test rather than primary efficacy evidence.

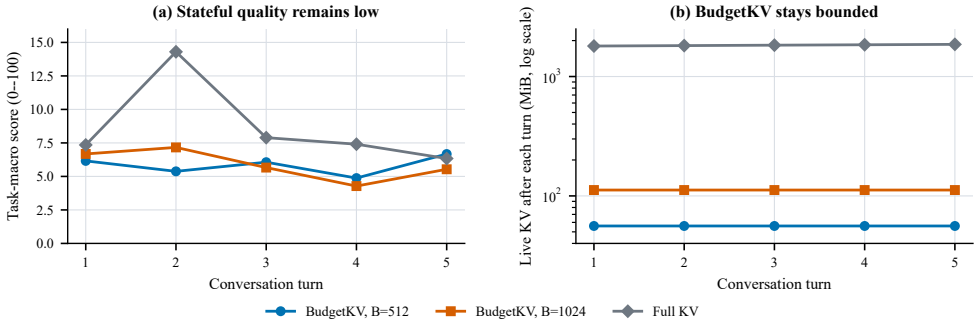

Figure 10: Five-turn stateful execution bounds live KV but does not establish strong multi-turn quality. Each curve averages six conversations per task.

Table 13: Five-turn Qwen stateful summary. Score averages four task means and five turns.

Method / B	Score	Live KV (MiB)	Decode (s)
BudgetKV / 512	5.83	56.0	2.09
BudgetKV / 1024	5.86	112.0	2.07
No semantic / 512	5.55	56.0	2.11
No semantic / 1024	6.68	112.0	2.14
Query-visible SnapKV / 512	6.32	94.3	2.07
Query-visible SnapKV / 1024	5.82	150.3	2.07
StreamingLLM / 512	5.91	56.0	2.05
StreamingLLM / 1024	6.07	112.0	2.11
Full KV	8.66	1830.3	4.17

BudgetKV does not recompute its query signal across turns, so evolving information needs remain outside the verified operating contract.

G Reproducibility Map

Table 14: Minimum reviewer path from claim to archived evidence.

Claim	Required records	Verification
Semantic increment	Frozen manifest, per-request outputs, paired effects	Match input/final-token hashes; recompute task-macro pairs and intervals
32K retrieval behavior	Generator manifest, source evidence checks, raw predictions	Verify exact lengths and score every measured method; leave unrun cells missing
Persistent-KV contract	Source snapshot and per-chunk token traces	Check append/select ordering, protected-block feasibility, and $B + p$ peaks
Jetson capacity	Per-process command, exit, and measurement receipts	Count completions before averaging successful-run memory
Figures	Plot tables and MATLAB sources	Regenerate plots without model inference

All primary numerical claims in the paper correspond to completed records. Failed or unsupported cells are retained as failures or missing values rather than converted to zero.