RELAXKV: RECOMPUTATION GUIDED BY THE QUERY WITH SPARSE CONTEXT ATTENTION FOR EFFICIENT KV CACHE REUSE

Anonymous authors
Paper under double-blind review

ABSTRACT

Cross-request KV caching reduces the prefill cost of Retrieval-Augmented Generation (RAG), but conventional prefix caching severely limits cache reuse across requests. Position-Independent Caching (PIC) removes this constraint by reusing independent chunks, but their KV states miss cross-chunk interactions. Existing methods selectively recompute token states to recover these missing interactions, but primarily allocate the recomputation budget to selecting which states to recompute, while fixing the recomputation context to the full causal prefix. We introduce **RelaxKV**, which formulates selective cache repair as a joint allocation problem over repair targets and recomputation context. Guided by the user query, RelaxKV identifies layer-specific repair targets and restricts their recomputation to a query-relevant context, reducing attention computation. Across four decoder models, RelaxKV at a 15% anchor ratio improves aggregate LongBench performance over ProphetKV on all models. On Qwen3-14B, RelaxKV provides a stronger quality–TTFT trade-off than ProphetKV across a 5%–30% anchor-ratio sweep, and achieves the best selective results on RULER-MV and LV-Eval at 16K and 32K context lengths. Controlled ablations further demonstrate the importance of recomputation context selection. Our code will be released upon acceptance.

1 INTRODUCTION

Retrieval-Augmented Generation (RAG) often retrieves overlapping document chunks across requests, making cross-request Key-Value (KV) cache reuse an effective way to reduce prefill cost (Lewis et al., 2020; Gao et al., 2023). Conventional prefix caching, however, only reuses KV states when requests share the same prefix, limiting reuse across requests. Position-Independent Caching (PIC) removes this restriction by precomputing chunks independently, but the resulting KV states miss cross-chunk interactions when the chunks are composed. Full recomputation restores these interactions but loses the efficiency benefit of cache reuse, while selective cache repair recomputes only part of the context to recover them at lower online cost.

Existing selective repair methods mainly differ in how they choose repair targets. EPIC repairs fixed boundary states, CacheBlend and KVShare select targets based on cache or hidden-state deviations, and ProphetKV uses query relevance (Hu et al., 2025; Yao et al., 2025; Yang et al., 2025b; Wang et al., 2026). Despite these different selection strategies, they share the same recomputation pattern: once a token is selected, it still attends to its full causal prefix during recomputation. Their recomputation budget therefore controls which token states are repaired, but not how much context each repair reads. As a result, even a small target set can still incur substantial attention computation, exposing a second design dimension: *recomputation context selection*.

We introduce RelaxKV, which formulates selective cache repair as a joint allocation problem over *repair targets* and *recomputation context*. RelaxKV first uses query attention to select repair anchors independently across layers. Deeper selections require the corresponding token states to be recomputed through the preceding layers, with each token remaining active only through its highest selected layer. This yields layer-specific active repair targets, while the union of all layer-wise anchors defines the KV positions available as attention context. At each layer, an active target attends only to the causally preceding positions in this union: active positions contribute newly recomputed

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.

KV states, while the remaining positions reuse their cached KV states. In this way, RelaxKV jointly controls which token states are recomputed and which KV entries they attend to, reducing both state and attention computation.

Experiments across four decoder models and multiple long-context benchmarks demonstrate the effectiveness of RelaxKV. At a 15% anchor ratio, RelaxKV improves aggregate LongBench performance over ProphetKV on all four models. On Qwen3-14B, sweeping the anchor ratio from 5% to 30% shows a stronger quality–TTFT trade-off than ProphetKV on RULER-MV. RelaxKV also achieves the best selective performance on both RULER-MV and LV-Eval at 16K and 32K context lengths. To isolate the role of recomputation context, we further evaluate RelaxKV^G, a matched-target control that retains ProphetKV's repair targets while changing only the recomputation context. Together with context-construction ablations, these results show that recomputation context selection is an important factor in the quality–latency trade-off.

Our main contributions are summarized as follows:

- We formulate selective cache repair as a joint allocation problem over two dimensions: *repair target selection*, governing state recomputation, and *recomputation context selection*, governing attention-context exposure.
- We introduce RelaxKV, which derives layer-specific repair targets and a shared query-relevant recomputation context from layer-wise query attention.
- Experiments demonstrate competitive quality–latency trade-offs across models, with particularly strong gains on long-context benchmarks.

2 Background and Motivation

2.1 Approximate Prefix KV Reuse

For a transformer with only a decoder (Vaswani et al., 2017) and L layers, a full prefill of a token sequence $x_{1:T}$ produces a KV cache indexed by layer, $\mathcal{K}^{\text{full}} = \{(K_l, V_l)\}_{l=0}^{L-1}$. In a reusable cache setting, the context is divided into chunks and each chunk is prefetched independently. Concatenating these chunk caches gives a cheap cache $\mathcal{K}^{\text{naive}}$, but generally $\mathcal{K}^{\text{naive}} \neq \mathcal{K}^{\text{full}}$ because the hidden state of a token was computed without all preceding chunks.

Positional consistency. Rotary position embedding (RoPE) rotates queries and keys according to their sequence positions, making their inner product depend on the relative displacement between tokens (Su et al., 2024). To separate this positional effect from contextual staleness across chunks, our evaluation prefetches each reusable chunk using its final global position IDs rather than positions that restart at the chunk boundary. If a cached chunk is later placed at a different offset, the RoPE rotation identity can realign its cached keys before repair (Chen et al., 2026). This positional operation is orthogonal to RelaxKV: even a positionally aligned cache remains approximate because each chunk was encoded without the hidden states of preceding chunks. RelaxKV addresses this missing context from preceding chunks rather than RoPE phase misalignment.

We use the following three cache states throughout the paper. *Full reuse* directly uses the precomputed chunk caches without repair. *Full recompute* performs a dense prefill over the complete context. *Selective repair* starts from the reusable cache, selectively recomputes token states, and updates their corresponding KV entries. RelaxKV operates within the selective repair setting; it does not modify cache construction or model weights.

2.2 Query-Aware Cache Repair

Let q denote the user query appended after the cached context. Running q against $\mathcal{K}^{\text{naive}}$ produces query-to-context attention at every layer. As illustrated in Figure 1, highly attended token positions can vary substantially across layers. We therefore pool the attention weights at each layer into a score $a_{l,i}$ for context position i, and use these layer-specific scores to guide repair. More generally, selective repair involves two decisions: which token states to recompute, and which cached KV states those tokens can attend to during recomputation. Repair target selection decides which states are recomputed, whereas recomputation context selection decides which KV states are used during

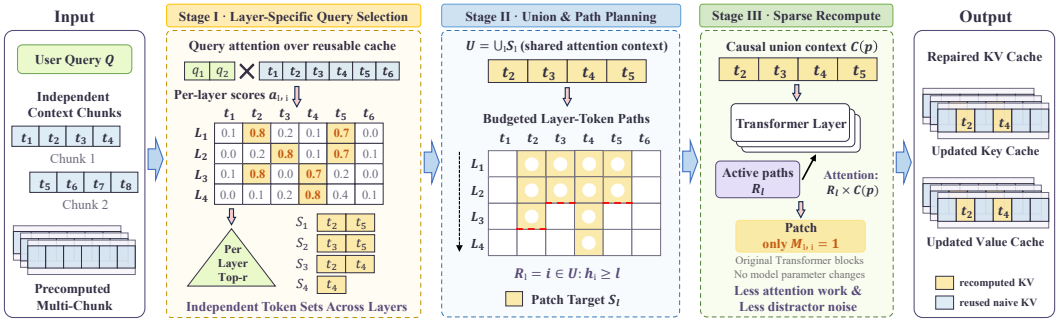

Figure 2: **Overview of RelaxKV.** Layer-wise query attention selects repair anchors, from which RelaxKV derives layer-specific active repair targets and a shared recomputation context. At each layer, active targets attend only to the causally valid positions in this context and update their corresponding KV entries.

their recomputation. Existing methods mainly optimize repair target selection while treating the full causal prefix as a fixed recomputation context.

This distinction motivates the design of RelaxKV. A token selected at a particular layer is a *repair anchor*. If the same token is selected at a deeper layer, its state must be recomputed through the preceding layers required to reach that layer, yielding the layer-specific *active repair targets*. Separately, *recomputation context tokens* contribute the keys and values available to these active targets during attention. Distinguishing repair anchors, active repair targets, and recomputation context tokens makes explicit the two dimensions of selective repair: which attention rows are recomputed and which cached KV columns they can access.

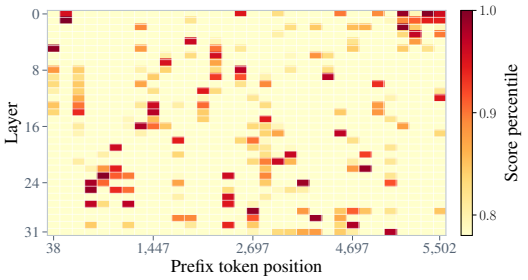

Figure 1: Layer-wise query-attention scores over prefix tokens. High-attention score tokens positions shift across layers.

3 METHOD

3.1 OVERVIEW

RelaxKV repairs a reusable chunk KV cache for a given query by jointly selecting repair targets and their recomputation context. Given the naive cache $\mathcal{K}^{\text{naive}}$, a query q, and an anchor ratio r, RelaxKV first uses layer-wise query attention to select repair anchors independently at each layer. A token selected at a deeper layer is recomputed through the preceding layers required to reach its highest selected layer, producing layer-specific active repair targets. The union of the layer-wise anchors forms a shared recomputation context.

At each layer, only the active repair targets execute the Transformer block. Each target attends to the causally valid subset of the shared context and updates its corresponding KV entry. Thus, repair target selection determines which token states are recomputed, while recomputation context selection determines which cached KV states they can attend to. Figure 2 summarizes the workflow.

3.2 LAYER-WISE QUERY ATTENTION AND REPAIR TARGETS

We run the query with $\mathcal{K}^{\text{naive}}$ and collect its attention to the cached context. For layer l and context position i, we average over heads and query positions and normalize within each layer:

$$a_{l,i} = \text{MinMax}\left(\frac{1}{HQ}\sum_{h=1}^{H}\sum_{t=1}^{Q} A_l[h,t,i]\right).$$

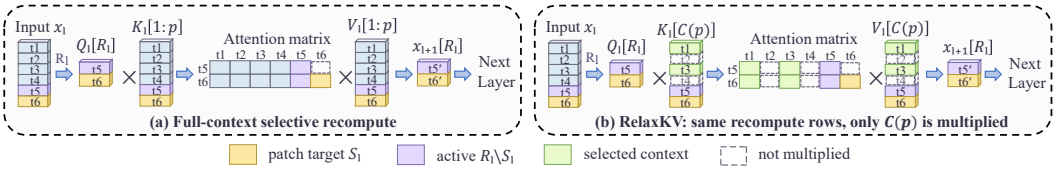

Figure 3: **Full-context versus RelaxKV recomputation.** Both execute the same active repair targets R_l. For a target at position p, full-context recomputation attends to the complete causal prefix, whereas RelaxKV attends only to $C_l(p) = U \cap \{j : j \le p\}$. Dashed entries are excluded from the recomputation context.

The score $a_{l,i}$ is used for selection; recomputation still executes the original Transformer blocks.

At each layer, RelaxKV independently selects the $k = \max(1, \mathrm{round}(rT_c))$ context positions with the highest scores, where T_c is the number of context tokens and r is the anchor ratio. We denote the selected set at layer l by S_l and refer to its elements as *repair anchors*. Independent layer-wise selection allows the repair targets to adapt to changes in token relevance across model depth rather than forcing all layers to share a single global target set.

A token selected at a deeper layer must be recomputed through the preceding layers required to reach that layer. For each token i appearing in at least one anchor set, we define its highest selected layer as

$$h_i = \max\{l : i \in S_l\},$$
$$U = \bigcup_l S_l,$$
$$R_l = \{i \in U : h_i \ge l\}.$$

Here, U is the union of all layer-wise anchors, and R_l is the set of active repair targets at layer l. A token remains active through its highest selected layer h_i and becomes inactive thereafter. Thus, $\{R_l\}_{l=0}^{L-1}$ closes the layer-wise anchor selections under the dependencies required for recomputation. The nominal anchor ratio r controls the number of selected anchors, while the resulting number of layer-token executions can be larger because deeper selections require recomputation through preceding layers.

3.3 Recomputation Context Selection

RelaxKV constructs the recomputation context from the shared anchor union U. At layer l, only active repair targets in R_l execute the Transformer block. For an active target at its original sequence position p, we define the causal recomputation context as

$$C_l(p) = U \cap \{j : j \le p\}.$$

Each target therefore attends only to the causally valid positions in U, rather than its full causal prefix. We gather the cached keys and values at these positions and replace entries in $R_l \cap C_l(p)$ with freshly recomputed KV states before computing attention. The resulting hidden states then pass through the original output projection, residual connections, normalization, and MLP. Tokens with $h_i = l$ complete their repair at this layer and leave the active set.

All active repair targets retain their original global position IDs when applying RoPE. Restricting attention to $C_l(p)$ therefore changes only which KV positions are available during recomputation, without altering the positional coordinates or relative offsets of the retained token pairs.

Figure 3 contrasts full-context recomputation with RelaxKV under the same active repair targets R_l. The two settings differ only in the KV positions that each target can attend to: full-context recomputation uses the complete causal prefix, whereas RelaxKV uses only $C_l(p)$.

3.4 CACHE UPDATE SEMANTICS

Recomputing the active targets produces fresh KV states for every $i \in R_l$. These states overwrite the corresponding entries in the reusable cache, while all other entries retain their naive values. The update set is therefore determined directly by R_l and requires no additional selection step.

During recomputation, however, the attention context can contain both fresh and cached KV states. For each position $j \in C_l(p)$ at layer l, RelaxKV uses

$$(\widetilde{K}_{l,j}, \widetilde{V}_{l,j}) = \begin{cases} (K^{\text{new}}_{l,j}, V^{\text{new}}_{l,j}), & j \in R_l \cap C_l(p), \\ (K^{\text{naive}}_{l,j}, V^{\text{naive}}_{l,j}), & j \in C_l(p) \setminus R_l. \end{cases}$$

Thus, an active target attends to freshly recomputed KV states for positions that are also active at the current layer, while reusing cached KV states for the remaining positions in its recomputation context. Once a token reaches its highest selected layer and leaves the active target set, its KV states at deeper layers are not recomputed and retain their naive values. The token may nevertheless remain in the shared context U and contribute these cached KV states during later recomputation. This mixed fresh–cached KV context is part of RelaxKV's approximation and is used consistently in all comparisons.

4 EVALUATION

Our evaluation addresses four questions. (1) Does RelaxKV preserve task quality across models and long-context workloads? (2) Does its effectiveness persist as context length increases? (3) When repair targets are fixed, how does recomputation context selection affect quality and cost? (4) Does jointly allocating repair targets and recomputation context improve the end-to-end quality–TTFT trade-off across repair budgets?

4.1 EXPERIMENTAL SETUP

Models. We evaluate four decoder models spanning 3B to 14B parameters: Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen3-14B (Yang et al., 2025a), Phi-4-14B (Abdin et al., 2024), and Llama-3.2-3B-Instruct (Meta AI, 2024). Llama and Phi use instruction-tuned checkpoints, while Qwen3 is evaluated with its default thinking mode enabled.

Tasks and metrics. We evaluate controlled long-context retrieval with RULER-MV (Hsieh et al., 2024) and natural long-context tasks with LongBench (Bai et al., 2024) and LV-Eval (Yuan et al., 2024), using the standard metric for each benchmark. For LongBench, we select seven QA, multi-hop QA, and passage-retrieval tasks that naturally match the retrieval-based reusable-cache setting, including MuSiQue (Trivedi et al., 2022), while the length stress tests evaluate contexts from 4K to 32K tokens.

Implementation and hardware. Experiments are conducted on NVIDIA H100 80GB HBM3 GPUs using PyTorch 2.8.0, CUDA 12.9, and Transformers 4.57.1, with BF16 model weights and greedy decoding. All selective repair methods run on a single GPU. Only the 16K and 32K full-recomputation reference runs use two GPUs with model parallelism and SDPA.

Baselines and protocol. We evaluate cache repair after retrieval, with the same retrieved chunks shared across all methods and retrieval latency excluded from the measured request time. Within each comparison, directly comparable methods use identical prompts, tokenization, model weights, 512-token chunks, and evaluation examples. We compare RelaxKV with full recomputation, full reuse by directly concatenating the precomputed chunk caches, and four selective repair methods: CacheBlend (Yao et al., 2025), EPIC (Hu et al., 2025), KVShare (Yang et al., 2025b), and ProphetKV (Wang et al., 2026). For Llama-3.2-3B, we additionally report KVLink using its publicly released checkpoint and blocked-attention inference (Yang et al., 2025c).

Control with Fixed ProphetKV Targets. Let P denote ProphetKV's global repair target set, which is shared across layers. We use RelaxKV^{G} as a matched-target diagnostic control: it retains ProphetKV's repair targets P and cache update mask, but replaces the full causal prefix with

Table 1: **LongBench results.** Full Recompute serves as the dense reference. Directly comparable selective baselines and RelaxKV^{G} use a nominal 20% repair ratio, while RelaxKV uses a fixed 15% anchor ratio. Bold and underline denote the best and second-best selective results, respectively.

Qwen3-14B

Method	WQA	TQA	HQA	NQA	MQue	PR-en	PR-zh	LB Avg
Full Recompute	8.50	88.41	54.95	24.15	25.57	68.00	100.00	52.80
Full Reuse	7.22	66.60	28.01	12.45	5.47	24.00	72.14	30.84
CacheBlend	8.46	85.11	42.57	21.57	20.57	66.00	77.00	45.90
EPIC	8.52	**87.23**	47.63	23.31	21.11	53.00	88.00	46.97
KVShare	7.46	86.47	33.46	13.64	7.03	36.50	69.04	36.23
ProphetKV	**8.64**	86.78	52.24	24.20	25.92	**74.00**	**100.00**	53.11
RelaxKV$^{\mathrm{G}}_{.20}$	8.55	86.70	50.94	**24.61**	23.89	**74.00**	**100.00**	52.67
RelaxKV$_{.15}$	8.32	86.84	**52.41**	23.88	**26.52**	**74.00**	**100.00**	**53.14**

Llama-3.1-8B

Method	WQA	TQA	HQA	NQA	MQue	PR-en	PR-zh	LB Avg
Full Recompute	47.23	88.01	49.10	24.73	25.86	69.00	85.71	55.66
Full Reuse	27.28	83.68	33.28	15.85	9.65	11.00	11.25	27.43
CacheBlend	37.24	84.93	45.68	21.03	20.15	33.00	37.88	39.99
EPIC	35.43	87.91	**51.76**	22.34	19.31	44.00	61.92	46.10
KVShare	29.35	88.73	36.01	21.47	18.56	19.50	24.00	33.95
ProphetKV	45.21	90.30	49.04	23.13	25.88	73.50	93.33	57.20
RelaxKV$^{\mathrm{G}}_{.20}$	45.86	**90.80**	48.62	**25.30**	25.40	74.50	**94.43**	**57.84**
RelaxKV$_{.15}$	**48.24**	90.46	48.32	22.96	**25.93**	**76.00**	91.60	57.64

Phi-4-14B

Method	WQA	TQA	HQA	NQA	MQue	PR-en	PR-zh	LB Avg
Full Recompute	22.02	87.74	38.41	13.83	9.11	65.25	97.33	47.67
Full Reuse	12.51	81.13	19.88	7.88	4.91	18.00	23.00	23.90
CacheBlend	15.47	82.99	28.84	8.23	4.72	50.00	37.10	32.48
EPIC	18.81	85.34	31.21	8.12	5.44	49.00	55.00	36.13
KVShare	14.79	84.82	32.01	8.58	6.13	42.50	47.08	33.70
ProphetKV	**20.48**	87.01	36.19	**8.90**	8.38	67.83	89.90	45.53
RelaxKV$^{\mathrm{G}}_{.20}$	18.97	86.19	33.39	8.69	8.14	66.64	90.24	44.61
RelaxKV$_{.15}$	19.86	**87.59**	**38.40**	8.21	**8.51**	**69.06**	**92.28**	**46.27**

Llama-3.2-3B

Method	WQA	TQA	HQA	NQA	MQue	PR-en	PR-zh	LB Avg
Full Recompute	29.49	82.47	48.06	22.10	18.57	67.00	40.00	43.96
Full Reuse	14.84	57.00	19.66	5.15	5.91	4.00	7.18	16.25
CacheBlend	18.84	57.89	20.72	5.95	5.07	3.00	7.45	16.99
EPIC	21.94	80.97	40.22	18.79	14.55	22.00	9.00	29.64
KVShare	20.63	77.59	25.52	12.99	13.37	5.50	5.53	23.02
KVLink†	17.69	**86.32**	44.24	19.32	6.92	26.00	5.66	29.45
ProphetKV	29.13	85.33	43.82	19.90	16.66	**72.50**	24.00	41.62
RelaxKV$^{\mathrm{G}}_{.20}$	28.74	84.83	45.82	20.07	17.17	68.50	30.00	42.16
RelaxKV$_{.15}$	**30.17**	84.41	**46.71**	**20.22**	**17.78**	70.00	**39.00**	**44.04**

† KVLink uses the publicly released Llama3B-KVLink5 checkpoint.

RelaxKV's recomputation context. Specifically, a target at position p attends to $U \cap \{j : j \leq p\}$, where U is constructed from the layer-wise query-attention anchors at the same nominal ratio. Thus, RelaxKV^{G} keeps repair target selection and cache updates fixed while changing only the recomputation context. We report it as a diagnostic control rather than a separate method.

In the main comparisons, selective baselines and RelaxKV^{G} use a 20% repair ratio, whereas RelaxKV uses an anchor ratio of $r = 0.15$. These nominal ratios are not directly equivalent to the realized computation of RelaxKV: selecting anchors at deeper layers requires recomputation through preceding layers, while their union determines the size of the recomputation context. We therefore separately report the realized context-set and active-target ratios in Sec. 4.5 and use measured TTFT to characterize actual online cost.

4.2 Accuracy on Standard Tasks

Table 1 reports task-level and aggregate LongBench results. Directly comparable selective baselines and RelaxKV^{G} use a nominal 20% repair ratio, while RelaxKV uses a 15% anchor ratio. Under these settings, RelaxKV improves the aggregate score over ProphetKV on all four models while maintaining performance close to the full-recomputation reference. Under matched ProphetKV targets, RelaxKV^{G} achieves aggregate scores within a small margin of ProphetKV across all models, showing that changing the recomputation context alone can preserve comparable task quality. At the task level, no single repair strategy dominates every dataset, but RelaxKV remains consistently competitive across multihop QA and passage-retrieval tasks.

4.3 Context Length Scaling

Table 2 evaluates Qwen3-14B on RULER-MV from 4K to 32K context lengths and on LV-Eval at 16K and 32K. Selective baselines and RelaxKV^{G} use a 20% repair ratio, while RelaxKV uses its standard 15% anchor ratio. On RULER-MV, Full Reuse degrades rapidly as the context grows, while selective repair remains substantially more robust. At 32K, RelaxKV achieves 95.00 accuracy, compared with 91.17 for ProphetKV and 92.00 for RelaxKV^{G}, and remains close to the Full Recompute reference.

LV-Eval provides a complementary setting with multiple documents and natural distractors. RelaxKV achieves the best selective performance at both 16K and 32K, reaching 28.71 and 25.40 token F1, compared with 24.78 and 19.32 for ProphetKV. Together, these results reveal an additional benefit of recomputation context selection beyond controlling computation: as the context becomes longer and contains more irrelevant information, restricting repair to query-relevant con-

Table 2: **Long-context results on Qwen3-14B.** RULER-MV reports accuracy and LV-Eval reports token F1.

Method	RULER-MV ↑				LV-Eval ↑	
	4K	8K	16K	32K	16K	32K
Full Recompute	100.00	100.00	96.67	95.83	32.37	23.27
Full Reuse	52.00	41.17	38.67	13.83	0.64	0.29
CacheBlend	69.33	55.50	41.50	20.33	1.66	0.74
EPIC	57.17	49.00	52.17	71.33	14.37	10.11
KVShare	53.67	47.17	44.33	24.00	0.98	0.38
ProphetKV	96.67	**98.83**	92.83	91.17	24.78	19.32
RelaxKV$^{\mathrm{G}}_{.20}$	96.17	97.83	93.67	92.00	22.13	17.30
RelaxKV$_{.15}$	**98.67**	98.67	**95.33**	**95.00**	**28.71**	**25.40**

Table 3: Effect of recomputation context construction on Qwen3-14B at a 15% anchor ratio. All variants share the same repair anchors, dependency-closed active repair targets, and pre-truncation context-set size $|U|$; only context construction varies. Panel (a) defines the variants, and panel (b) reports task quality / mean recomputation latency (s).

Variant	Context Source	Scope	Set size
Global Sparse	Across-Layer Mean	Shared	$\|U\|$
Layerwise Sparse	Per-Layer Score	At Each Layer	$\|U\|$
RelaxKV$_{.15}$	Anchor union U	Shared	$\|U\|$

(a) Context Set Components

Variant	MuSiQue	RULER 32K	LV-Eval 16K	LV-Eval 32K
Global Sparse	34.81 / 5.97	**95.00** / 31.87	27.32 / 25.44	25.35 / 78.99
Layerwise Sparse	31.78 / 5.89	85.00 / 31.86	18.18 / 25.48	21.76 / 79.02
RelaxKV$_{.15}$	**38.08** / 5.84	**95.00** / 31.72	**28.71** / 25.50	**25.40** / 78.39

(b) Quality / recomputation latency (s)

text can better preserve quality under a limited repair budget. This pattern is consistent with an implicit filtering, or denoising, effect during recomputation.

4.4 Ablation Study of Recomputation Context Construction

Table 3 isolates the effect of recomputation context construction while fixing the repair schedule. All variants use the same 15% layer-wise anchor budget, identical anchors S_l, and the same active repair targets R_l. They also use the same nominal context-set size before causal truncation. Let U_l denote the context set used at layer l; for a repair target at position p, the resulting causal context is $C_l(p) = U_l \cap \{j : j \leq p\}$. The variants differ only in how U_l is constructed. *Global Sparse* selects a single context set shared across layers using scores averaged over depth, *Layerwise Sparse* selects an independent context set at each layer. RelaxKV uses the union of the layer-wise repair anchors.

Matching the context-set size controls the nominal number of retained KV positions, but does not guarantee identical attention work. Because causal truncation depends on the target position p, the effective edge count $\mathcal{E}_{\mathrm{unmasked}} = \sum_l \sum_{p \in R_l} |C_l(p)|$ can differ across constructions even when their pre-truncation context sizes match. We therefore treat this experiment as a controlled ablation of context construction rather than a strict equal-work comparison, and report measured recomputation latency alongside quality.

The results show that context construction matters even when the repair targets and nominal context budget are fixed. The three variants have similar recomputation latency, yet their task quality can differ substantially, indicating that recomputation context selection is not determined by sparsity alone: which KV positions are retained is itself an important design choice.

4.5 Budget Accounting and Online Serving Latency

Figure 4 relates RelaxKV's nominal anchor budget to its realized computation and end-to-end quality–TTFT trade-off on Qwen3-14B. Panel (a) reports the shared-context union ratio and the dependency-closed active-target ratio as the layer-wise anchor ratio varies. Because anchors selected at deeper layers require recomputation through preceding layers, the nominal anchor ratio does not directly equal the fraction of executed layer-token states. At $r = 0.15$, for Qwen3-14B on the 8K RULER-MV workload with 512-token chunks, the shared anchor union covers 63.05% of context positions, while the dependency-closed active targets account for 42.85% of layer-token

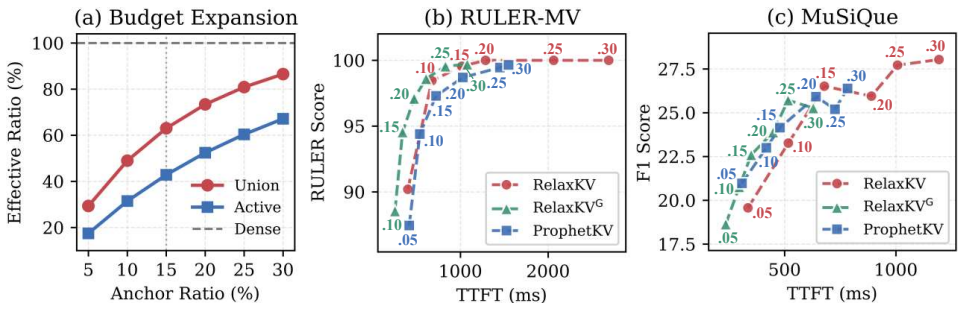

Figure 4: Qwen3-14B budget accounting and quality–TTFT trade-offs. (a) A layer-wise anchor ratio induces a shared-context union ratio and a dependency-closed active-target ratio. (b–c) Quality–latency trade-off on Qwen3-14B. We sweep nominal recomputation ratios from 5% to 30%; labels denote the ratios. RelaxKV reaches higher RULER-MV accuracy at lower TTFT.

states, leaving 57.15% unrecomputed. Panels (b–c) report the corresponding quality and TTFT on RULER-MV and MuSiQue. As the anchor ratio increases, RelaxKV trades additional recomputation for higher task quality, producing a favorable quality–TTFT trade-off across the evaluated budgets. RelaxKV^{G} provides the complementary matched-target control, isolating the effect of recomputation context under ProphetKV's repair targets. Together, these results show that RelaxKV can make a relatively broad query-relevant KV context available while recomputing only a subset of token states. We use $r = 0.15$ for the cross-model comparisons in the main experiments.

5 Related Work

5.1 KV Cache Reuse and Selective Recomputation

Position-Independent Caching (PIC) enables KV states of independently encoded chunks to be reused across requests with different preceding contexts. Because independently encoded chunks omit cross-chunk interactions, directly composing their KV caches introduces approximation error, motivating online selective repair. CacheBlend, EPIC, KVShare, and ProphetKV address this reusable-cache repair problem with different repair-target selection strategies: EPIC repairs fixed boundary states, CacheBlend and KVShare identify targets using cache or hidden-state deviations, and ProphetKV selects targets according to query relevance (Yao et al., 2025; Hu et al., 2025; Yang et al., 2025b; Wang et al., 2026). These methods primarily determine which token states to recompute, while selected targets still attend to their full causal prefixes. RelaxKV operates on the same PIC-based reusable cache without modifying the base model or cache construction. In contrast to prior methods, which select repair targets but recompute them against the full causal prefix, RelaxKV explicitly selects the KV context available during recomputation. This extends selective cache repair from repair-target selection alone to joint allocation over repair targets and recomputation context.

Several related works approach reusable KV caching from broader system or model-design perspectives. Cache-Craft integrates chunk-level KV reuse with cache-variant management, hierarchical storage, and selective recomputation in an end-to-end RAG serving system (Agarwal et al., 2025). Its scope therefore extends beyond the selective cache-repair problem considered here. Other methods modify the cache construction or model behavior to improve cross-context reuse. KVLink combines positional adjustment with trainable special tokens to recover cross-document interactions (Yang et al., 2025c), while KV Packet introduces reusable soft-token boundary adapters learned offline to reduce online recomputation (Chen et al., 2026). In contrast, RelaxKV leaves the base model and cache construction unchanged and focuses on allocating online recomputation over repair targets and recomputation context.

5.2 Query-Aware Selection and Sparse Attention

Several methods use query-related signals to identify important KV states or context positions. SnapKV uses query-side attention to identify salient context positions and compress an already con-

textualized KV cache (Li et al., 2024). KVzip instead estimates KV importance by reconstructing the original context, producing a compressed cache that is independent of subsequent queries (Kim et al., 2025). These methods reduce the size of an existing contextualized cache, but do not address the missing cross-chunk interactions that arise when independently encoded caches are composed.

Query guidance has also been used to reduce computation during decoding, prefill, and cache repair. ProphetKV, the closest baseline to RelaxKV, uses the user query to select repair targets in a reusable-cache setting (Wang et al., 2026). Quest uses the current query vector to select KV pages during decoding, while QUOKA selects representative queries and relevant keys to accelerate chunked prefill (Tang et al., 2024; Jones et al., 2026). REFORM combines retrieval and recomputation for long contexts by constructing compressed representations across layers, gathering query-relevant input segments, and forwarding the selected inputs to build a compact cache (Song et al., 2025). These methods demonstrate the value of query-guided selection and sparse computation, but they optimize different stages or cache representations from the selective repair problem considered here.

RelaxKV starts from an existing reusable cache, retains KV entries outside the repair target set, and updates selected entries through online recomputation. More importantly, it separates two decisions that are coupled in prior selective-repair methods: which token states are recomputed and which cached KV states are available during that recomputation.

5.3 Long-Context Evaluation

LongBench provides a diverse suite of natural long-context tasks, including question answering, multihop reasoning, summarization, and passage retrieval (Bai et al., 2024). RULER complements these tasks with controlled synthetic evaluations that probe retrieval and reasoning as context length increases (Hsieh et al., 2024). LV-Eval further evaluates long-context understanding in settings with multiple documents and natural distractors (Yuan et al., 2024). Together, these benchmarks cover both natural task diversity and controlled length scaling. We use LongBench to evaluate overall task quality, and RULER-MV and LV-Eval to examine whether selective cache repair remains effective as context length increases and irrelevant context becomes more prominent.

6 Conclusion

We introduced RelaxKV, which extends selective cache repair from repair-target selection alone to joint allocation over repair targets and recomputation context. Guided by layer-wise query attention, RelaxKV derives layer-specific repair targets and a shared query-relevant recomputation context. Across four decoder models and multiple long-context benchmarks, RelaxKV achieves competitive quality–latency trade-offs across models and particularly strong performance in long-context settings. Its advantage remains strong as context length increases, with particularly competitive performance on the 16K and 32K RULER-MV and LV-Eval evaluations. Matched-target and context-construction controls further validate the effectiveness of recomputation context selection.

AI use statement

Generative AI tools were used solely for language polishing, including improvements to grammar, wording, clarity, and readability. All AI-assisted edits were reviewed by the authors, who take responsibility for the final content of the manuscript.

References

Marah Abdin et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Shubham Agarwal, Sai Sundaresan, Subrata Mitra, Debabrata Mahapatra, Archit Gupta, Rounak Sharma, Nirmal Joshua Kapu, Tong Yu, and Shiv Saini. Cache-craft: Managing chunk-caches for efficient retrieval-augmented generation. *Proceedings of the ACM on Management of Data*, 3(3), June 2025. doi: 10.1145/3725273.

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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Chuangtao Chen, Grace Li Zhang, Xunzhao Yin, Cheng Zhuo, Bing Li, and Ulf Schlichtmann. KV Packet: Recomputation-free context-independent KV caching for LLMs. *arXiv preprint arXiv:2604.13226*, 2026.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

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

Cheng-Ping Hsieh et al. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Junhao Hu, Wenrui Huang, Weidong Wang, Haoyi Wang, Tiancheng Hu, Zhang Qin, Hao Feng, Xusheng Chen, Yizhou Shan, and Tao Xie. EPIC: Efficient position-independent caching for serving large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24391–24402, 2025. URL https://proceedings.mlr.press/v267/hu25j.html.

Dalton Jones, Junyoung Park, Matthew Morse, Mingu Lee, Chris Lott, and Harper Langston. QUOKA: Query-oriented KV selection for efficient LLM prefill. In *International Conference on Learning Representations*, 2026.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-agnostic KV cache compression with context reconstruction. In *Advances in Neural Information Processing Systems*, 2025.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 2020.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: Llm knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, 2024.

Meta AI. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. Official model release and model card, 2024. URL https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/.

Woomin Song, Sai Muralidhar Jayanthi, Srikanth Ronanki, Kanthashree Mysore Sathyendra, Jinwoo Shin, Aram Galstyan, Shubham Katiyar, and Sravan Babu Bodapati. Compress, gather, and recompute: REFORMing long-context processing in transformers. In *Advances in Neural Information Processing Systems*, 2025.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

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*, 2024.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Shihao Wang, Jiahao Chen, Yanqi Pan, Hao Huang, Yichen Hao, Xiangyu Zou, Wen Xia, Wentao Zhang, Chongyang Qiu, and Pengfei Wang. ProphetKV: User-query-driven selective recomputation for efficient KV cache reuse in retrieval-augmented generation. *arXiv preprint arXiv:2602.02579*, 2026. doi: 10.48550/arXiv.2602.02579. URL https://arxiv.org/abs/2602.02579.

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

Huan Yang, Renji Zhang, Mingzhe Huang, Weijun Wang, Yin Tang, Yuanchun Li, Yunxin Liu, and Deyu Zhang. KVShare: An LLM service system with efficient and effective multi-tenant KV cache reuse. *arXiv preprint arXiv:2503.16525*, 2025b.

Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. KVLink: Accelerating large language models via efficient KV cache reuse. *arXiv preprint arXiv:2502.16002*, 2025c. doi: 10.48550/arXiv.2502.16002. URL https://arxiv.org/abs/2502.16002.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. CacheBlend: Fast large language model serving for RAG with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 94–109, 2025. doi: 10.1145/3689031.3696098. URL https://doi.org/10.1145/3689031.3696098.

Tao Yuan, Xuefei Ning, Dong Zhou, Zhijie Yang, Shiyao Li, Minghui Zhuang, Zheyue Tan, Zhuyu Yao, Dahua Lin, Boxun Li, Guohao Dai, Shengen Yan, and Yu Wang. LV-Eval: A balanced long-context benchmark with 5 length levels up to 256k. *arXiv preprint arXiv:2402.05136*, 2024.

A Exact Sparse-Execution Semantics

A.1 Repair Schedule

Let T_c denote the number of reusable context tokens and L the number of decoder layers. Query attention produces a layer-specific score $a_{l,i}$ for context token i at layer l. Given an anchor ratio r, RelaxKV selects

$$k = \max(1, \mathrm{round}(rT_c)) \tag{1}$$

repair anchors independently at each layer:

$$S_l = \mathrm{TopK}_i(a_{l,i}, k). \tag{2}$$

For each token selected at least once, we define its highest selected layer, the shared anchor union, and the active repair targets as

$$h_i = \max\{l : i \in S_l\}, \qquad U = \bigcup_l S_l, \tag{3}$$

$$R_l = \{i \in U : h_i \geq l\}. \tag{4}$$

A token selected at a deeper layer is recomputed through the preceding layers required to reach its highest selected layer h_i, and becomes inactive thereafter. Thus, R_l contains exactly the repair targets that execute layer l.

For an active repair target at original sequence position p, the causal recomputation context is

$$C_l(p) = U \cap \{j : j \leq p\}. \tag{5}$$

The same union U is shared across layers, while causal masking determines the subset available to each target. Original global position IDs are retained when applying RoPE, so restricting attention to $C_l(p)$ changes only which KV positions are available during recomputation without renumbering the retained tokens or altering their positional offsets.

Algorithm 1 RelaxKV cache repair

Require: Naive cache $\mathcal{K}^{\mathrm{naive}}$, context tokens x, query q, anchor ratio r
Run q against $\mathcal{K}^{\mathrm{naive}}$ and obtain layer-wise scores $a_{l,i}$
$k \leftarrow \max(1, \mathrm{round}(rT_c))$
for $l = 0, \ldots, L-1$ **do**
  $S_l \leftarrow \mathrm{TopK}_i(a_{l,i}, k)$
end for
$U \leftarrow \bigcup_l S_l$
for each $i \in U$ **do**
  $h_i \leftarrow \max\{l : i \in S_l\}$
end for
for $l = 0, \ldots, L-1$ **do**
  $R_l \leftarrow \{i \in U : h_i \geq l\}$
end for
Initialize the output cache from $\mathcal{K}^{\mathrm{naive}}$
Initialize hidden states for positions in U from their input representations
for $l = 0, \ldots, L-1$ **do**
  Compute Q/K/V for the active repair targets in R_l
  Gather cached K/V states at positions in U
  Replace gathered entries in R_l with freshly recomputed K/V states
  Apply causal attention so that each target $p \in R_l$ attends only to $C_l(p)$
  Execute the unchanged output projection, residual connections, normalization, and MLP
  Update the cache entries corresponding to all targets in R_l
end for
return Repaired cache

A.2 Recomputation Context and Cache Update

RelaxKV constructs a compact recomputation context indexed by the shared anchor union U. For an active repair target at position p and a context position $j \in C_l(p)$, the KV state used at layer l is

$$(\widetilde{K}_{l,j}, \widetilde{V}_{l,j}) = \begin{cases} (K^{\mathrm{new}}_{l,j}, V^{\mathrm{new}}_{l,j}), & j \in R_l \cap C_l(p), \\ (K^{\mathrm{naive}}_{l,j}, V^{\mathrm{naive}}_{l,j}), & j \in C_l(p) \setminus R_l. \end{cases} \tag{6}$$

Table 4: Controlled configurations. Every active repair target updates its corresponding KV entry; configurations vary in active repair targets or recomputation context columns. Global repair targets P are shared by all layers.

Configuration	Active Repair Targets	Recomputation Context	Role
RelaxKV	R_l	$C_l(p)$	Main method
RelaxKV^{G}	Global P	$C_l(p)$	Matched-target diagnostic control
Full-Prefix Control	R_l	Full causal prefix	Full-prefix quality–cost control
Global Sparse	R_l	Global Top-$\lvert U\rvert$	Global context construction
Layerwise Sparse	R_l	Layerwise Top-$\lvert U\rvert$	Layerwise context construction
Global Repair Target Sweep	Global P_{r_g}	$C_l(p)$	Global target capacity

Thus, active positions contribute freshly recomputed KV states, while the remaining positions in the recomputation context reuse their cached values. Positions outside U are not directly available during recomputation. After executing the Transformer block, the newly recomputed KV states overwrite the cache entries corresponding to all targets in R_l; no additional selection step is required for cache update.

For compact execution, the global positions in U are stored in sorted order and mapped to compact indices once per request. At each layer, cached KV states at these positions are gathered, and entries corresponding to active targets are replaced with freshly recomputed states. Causal masking then restricts each target at position p to

$$C_l(p) = U \cap \{j : j \leq p\},$$

while preserving the original global position IDs used by RoPE.

The two selection dimensions affect different parts of the computation. We characterize attention-context exposure by the number of causal attention edges,

$$\mathcal{E}_{\mathrm{unmasked}}(R, C) = \sum_l \sum_{p \in R_l} |C_l(p)|, \tag{7}$$

while projection and MLP work scale with the number of executed layer-token states,

$$\mathrm{Cost}_{\mathrm{state}}(R) \propto \sum_l |R_l|. \tag{8}$$

The edge count $\mathcal{E}_{\mathrm{unmasked}}$ is a structural measure rather than an exact FLOP count for our eager grouped-query attention (GQA) implementation. KV heads are expanded to the query-head layout before rectangular attention is executed, and causally masked entries may therefore still incur kernel work. We consequently use measured recomputation latency and TTFT as the primary efficiency metrics. The nominal anchor ratio, realized context-set ratio, and active-target ratio describe different quantities.

A.3 Controlled Configurations

Table 4 summarizes the controlled configurations used to isolate the effects of context restriction, cross-layer context sharing, and repair-target construction. Unless otherwise noted, context-construction variants use the same layer-wise anchors S_l and active repair targets R_l, so that the primary design variable is the recomputation context.

A.4 Full-Prefix Quality–Cost Control

The context-construction ablation in the main paper matches the nominal context-set size before causal truncation, but does not assume equal kernel work. The realized budgets can differ substantially from the nominal anchor ratio. In the Llama-3.1 sweep, a 20% anchor ratio yields mean context-set ratios of 65.1% on RULER-MV and 70.4% on MuSiQue. For Qwen3-14B at a 15% anchor ratio, the mean context-set and active-target ratios are 64.1%/44.8% on MuSiQue, 78.1%/55.2% on RULER-MV 32K, and 80.3%/57.8% on LV-Eval. These measurements illustrate

Table 5: Full-prefix quality–cost control on Qwen3-14B with a 20% anchor ratio. Both configurations share repair anchors, dependency-closed active targets, global position IDs, and cache updates. Recomp. is mean recomputation latency per example.

		MuSiQue		RULER-MV 32K		LV-Eval 16K		LV-Eval 32K	
Variant	**Recomputation Context**	**F1** ↑	**Recomp. (s)** ↓	**Score** ↑	**Recomp. (s)** ↓	**F1** ↑	**Recomp. (s)** ↓	**F1** ↑	**Recomp. (s)** ↓
Full-Prefix Control	Full causal prefix	39.63	10.11	96.00	55.05	29.32	38.78	23.32	115.97
RelaxKV$_{.20}$	$C_l(p)$	39.01	8.51	95.00	42.30	28.87	32.88	23.86	104.45

why the nominal anchor ratio should not be interpreted as either the realized context budget or the fraction of executed layer-token states.

Table 5 isolates the effect of recomputation context restriction by fixing the active repair targets and varying only the context available to them. The full-prefix control allows each target to attend to its complete causal prefix, whereas RelaxKV uses its restricted recomputation context. Across the four settings, RelaxKV reduces mean recomputation latency by 9.9–23.2%, while the absolute quality difference remains within 0.45–1.00 points. This experiment therefore characterizes the direct quality–cost trade-off introduced by context restriction and is complementary to the equal-size context-construction ablation in the main paper.

A.5 Cross-Model Quality–TTFT Trade-Off Curves

Figure 5 extends the Qwen3-14B quality–TTFT analysis in the main paper to all four evaluated models. The top row reports RULER-MV accuracy and the bottom row reports MuSiQue F1, with columns corresponding to Qwen3-14B, Llama-3.1-8B, Phi-4-14B, and Llama-3.2-3B. All panels use the same cache-ready TTFT protocol, and each curve traces the quality–latency trade-off as the nominal repair or anchor ratio varies.

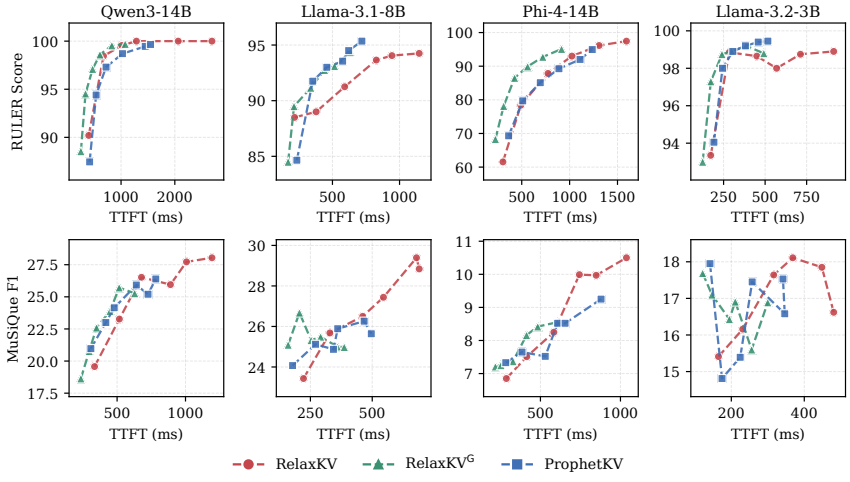

Figure 5: Cross-model quality–TTFT trade-off curves.

A.6 Chunk Size Sensitivity

We vary the reusable chunk size while keeping the model, prompts, tokenization, evaluation examples, and nominal repair ratios fixed. Table 6 reports Qwen3-14B results on MuSiQue without truncation for the chunk sizes evaluated by all methods. Across the evaluated range, all reuse-based methods improve as the chunk size increases, consistent with larger chunks preserving more within-chunk contextual interactions during independent encoding.

Table 6: Sensitivity to reusable chunk size on Qwen3-14B MuSiQue. Selective baselines and RelaxKV^{G} use a nominal 20% repair target ratio; RelaxKV uses a 15% anchor ratio. Full Recompute is independent of reusable chunk size and is repeated as a common dense reference. Bold and underline mark the best and second-best selective results.

Chunk size	Recompute	Reuse	CacheBlend	EPIC	KVShare	ProphetKV	RelaxKV$^{\mathrm{G}}_{.20}$	**RelaxKV**$_{.15}$
64	41.89	4.45	5.54	22.54	6.40	<u>26.77</u>	24.94	**32.66**
128	41.89	5.27	7.33	25.49	7.81	<u>31.75</u>	30.08	**34.54**
256	41.89	7.81	9.64	30.49	10.46	<u>33.97</u>	31.62	**36.39**
512	41.89	12.62	13.64	30.19	13.41	35.57	<u>36.28</u>	**38.08**
1024	41.89	18.46	22.47	34.05	19.00	**39.27**	37.62	<u>38.93</u>

Table 7: Global repair target capacity sweep on Llama-3.1-8B-Instruct. The shared context is fixed as the union of layerwise Top-20% anchors. TTFT is in milliseconds.

Repair Target Ratio	RULER ↓	MuSiQue ↓	LB Avg ↑
0.2	361.58	267.86	57.84
0.3	486.23	332.68	57.24
0.4	609.55	396.31	57.57
0.5	753.35	458.32	57.37
0.6	871.97	539.33	58.06
0.7	941.26	608.62	58.10
Target-Union	914.47	606.73	58.10

B Global Repair Target Capacity Analysis

B.1 Controlled Setup

This diagnostic tests whether a larger global repair target set shared across layers can substitute for RelaxKV's layer-specific active repair targets. We fix the recomputation context U as the union of layer-wise Top-20% anchors and define the global score and target set as

$$g_i = \frac{1}{L}\sum_{l=0}^{L-1} a_{l,i}, \qquad P_{r_g} = \mathrm{TopK}_i(g_i, \min(\lfloor r_g T_c \rfloor, |U|)) \,. \tag{9}$$

The same target set P_{r_g} is recomputed and updated at every layer, while each target attends only to the causal subset of U. Targets are ranked over the full context, so the cardinality cap does not require $P_{r_g} \subseteq U$. The *Target-Union* configuration matches only the target count, $|P| = |U|$, rather than setting $P = U$.

Increasing the global target ratio substantially raises TTFT while providing limited aggregate quality gains. From 20% targets to the Target-Union configuration, RULER-MV TTFT increases from 361.58 to 914.47 ms and MuSiQue TTFT from 267.86 to 606.73 ms, corresponding to 2.53× and 2.27× increases. Over the same range, the seven-task LongBench average changes only from 57.84 to 58.10 and varies non-monotonically at intermediate ratios. Thus, under a fixed recomputation context, simply increasing the number of globally shared repair targets yields rapidly increasing execution cost with limited aggregate benefit.

Requested target ratios may saturate at the per-example context-set cardinality. Table 8 therefore reports the realized target and context-set ratios together with the per-task scores used in the aggregate. Some examples are already capped at a requested ratio of 0.7, causing the realized ratios to differ across datasets.

This diagnostic shows that target count alone does not explain the benefit of a repair schedule that varies across layers. It does not imply that a particular target selector is universally optimal, nor that equal nominal ratios correspond to equal computation. End-to-end quality–latency comparisons remain those reported in the main paper.

Table 8: Effective repair-target/context-set ratios and complete LongBench scores for the global target sweep. Requested target ratios are capped per example at $|U|$.

Repair Target	RULER Ratio	MuSiQue Ratio	WQA	TQA	HQA	NQA	MQue	PR-en	PR-zh	Avg
0.2	.200/.651	.200/.704	45.86	90.80	48.62	25.30	25.40	74.50	94.43	57.84
0.3	.300/.651	.300/.704	44.96	90.30	47.61	24.90	25.02	75.50	92.38	57.24
0.4	.400/.651	.400/.704	45.14	90.30	48.57	25.37	25.71	76.00	91.88	57.57
0.5	.500/.651	.500/.704	45.56	91.13	48.48	24.51	25.48	76.00	90.46	57.37
0.6	.600/.651	.600/.704	46.40	91.46	49.29	25.86	25.99	75.50	91.90	58.06
0.7	.651/.651	.694/.704	45.70	91.46	48.07	26.19	28.13	76.00	91.13	58.10
Target-Union	.651/.651	.704/.704	45.70	91.46	48.07	26.19	28.13	76.00	91.13	58.10

C Reproducibility Details

C.1 Model, Cache, and Precision

We evaluate Llama-3.1-8B-Instruct, Qwen3-14B, Phi-4-14B, and Llama-3.2-3B-Instruct in BF16 with batch size one, greedy decoding, and random seed 42. Experiments use PyTorch 2.8.0, CUDA 12.9, and Transformers 4.57.1 on NVIDIA H100 80GB GPUs. All selective methods run on one GPU; only the 16K and 32K full-recomputation references use two-GPU model parallelism.

Reusable contexts are partitioned into 512-token chunks and encoded independently with their final global position IDs. Their KV caches are concatenated to form the naive reusable cache used by all selective methods.

C.2 Task Definitions and Evaluation Splits

RULER-MV uses a nominal 8K context in the standard comparison and 4K–32K in the length stress test. LongBench follows the standard task metrics with an 8,192-token input limit and head–tail truncation. The recomputation-context ablation uses untruncated MuSiQue inputs, while LV-Eval is evaluated at 16K and 32K context lengths.

C.3 Accuracy Protocol

Within each comparison, all methods use identical prompts, tokenization, model weights, and evaluation examples. The global target-capacity analysis uses Llama-3.1-8B-Instruct. The reported LongBench aggregate is the arithmetic mean of the seven selected task scores.

C.4 Cache-Ready TTFT Protocol

TTFT is measured after one untimed warm-up with three timed repeats and one generated token. The reusable naive cache is resident on GPU before timing. For selective methods, timing includes query-attention scoring, target and context selection, KV gathering, recomputation, cache update, query prefill, and first-token decoding. Reusable-cache construction is excluded.