CONSTRUCT BEFORE YOU REPAIR: SURROGATE CONTEXTS FOR NON-PREFIX KV CACHE REUSE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model serving increasingly reuses previously computed KV caches to avoid repeatedly prefilling long or retrieved contexts. While prefix caching is exact, more flexible reuse requires cached text chunks to be composed under different future contexts, creating a mismatch between offline cache construction and online use. Existing approaches typically encode each chunk in isolation and repair the resulting cache through online selective recomputation, effectively treating unknown future context as empty. We revisit this default and introduce SurroKV, a training-free *surrogate-context cache construction* method that encodes each reusable chunk behind a fixed, request-independent surrogate context and stores only the resulting chunk KV caches. The surrogate is discarded after construction and never appears at inference time, preserving both cache size and the online execution path. Our analysis shows that independent construction introduces substantial attention-routing mismatch and that surrogate conditioning can reduce this mismatch without knowing the realized future context. Across RULER, LongBench, and HELMET on multiple models, SurroKV improves accuracy by 2.1%–12.7% relative to existing methods, most notably under limited recomputation budgets and longer contexts, with no additional online latency. These results highlight cache construction itself as an important design dimension for position-independent caching.

1 INTRODUCTION

Large language models (LLMs) integrated with retrieval-augmented generation (RAG) (Lewis et al., 2020) have become a widely adopted paradigm for grounding generation in external knowledge. Given a user query, a RAG system retrieves relevant document chunks from an external corpus and incorporates them into the model input. As modern applications retrieve more evidence and longer documents, their growing contexts increase the demands of long-context inference. The computational burden of long-context RAG is concentrated in the prefill stage, which processes the input prompt and constructs the KV cache before autoregressive decoding. Dense self-attention makes prefill cost grow quadratically with context length (Vaswani et al., 2017; Kwon et al., 2023), increasing time to first token (TTFT) as retrieved contexts grow. This computation is particularly redundant in RAG: the same document chunks are often retrieved across requests, yet their KV caches are recomputed whenever the surrounding context or retrieval order changes.

KV cache reuse provides a natural mechanism to avoid repeated computation. Prefix caching provides exact reuse for shared prefixes (Kwon et al., 2023; Zheng et al., 2024), but this condition is often not met in RAG, where retrieved chunks appear under different preceding contexts and in different orders. Position-independent caching (PIC) addresses this limitation by precomputing the KV cache of each chunk and assembling the caches at inference time (Gim et al., 2024). However, simply concatenating independently constructed KV caches results in severe accuracy degradation because of missed cross-chunk context interactions. To alleviate this, existing approaches repair the assembled cache via partial recomputation, selectively recomputing the KV cache for a subset of context tokens (Yao et al., 2025; Hu et al., 2024; Yang et al., 2025b; Wang et al., 2026).

Despite these efforts, we identify a common but limiting assumption: reusable chunks are constructed in isolation, effectively treating unknown future context as empty. This construction choice affects the KV cache of the entire chunk, whereas online partial recomputation can correct only a small fraction of important tokens under a limited computation budget. Consequently, many independently constructed caches remain unchanged during inference, leaving the construction-induced

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.

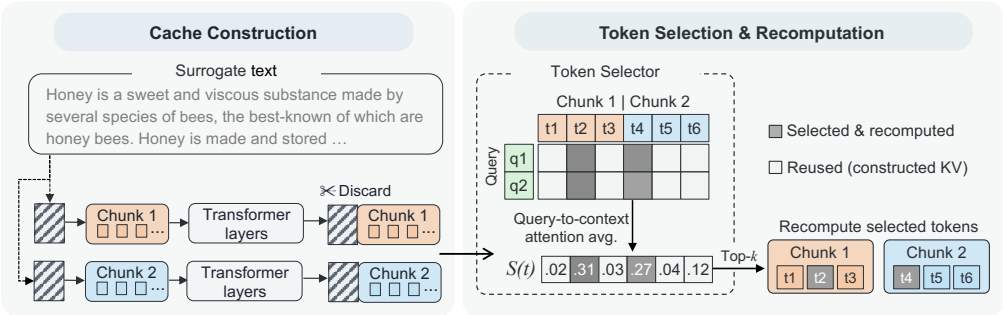

Figure 1: Framework overview. **Stage I: Offline Cache Construction.** Each chunk is encoded with a shared surrogate context, whose KV states are discarded after construction. **Stage II: Online Token Selection and Recomputation.** Query-to-context attention selects the top-k tokens for recomputation, while the remaining tokens reuse their surrogate-conditioned KV states.

mismatch uncorrected. Although the marginal impact of unrecomputed tokens may be limited, these residual errors can accumulate across the context and ultimately degrade performance.

Motivated by this, we revisit the cache-construction stage and ask: *how should a reusable cache be constructed when its future context is unknown?* To examine this, we compare independently constructed caches with full prefill and decompose their attention-output discrepancy into attention-routing and value-representation errors (defined in Eq. 3). Figure 2 shows that attention-routing error dominates the mismatch across layers under independent reuse. Since attention routing depends directly on the cached keys, this finding indicates that cache construction can affect how future queries attend to reused context. We therefore reconsider isolation as the default and examine a simple alternative: constructing each chunk under a fixed non-empty context. This change alone substantially reduces the routing mismatch without online recomputation, while leaving the value-representation error largely unchanged.

Building on these findings, we propose SurroKV, a simple yet effective *surrogate-context cache construction* strategy. Our approach comprises two stages, as shown in Figure 1. During offline cache construction, we prepend a fixed, request-independent surrogate context and retain only the resulting chunk KV cache, discarding the surrogate KV cache after construction. This allows the entire reusable chunk cache to be constructed under a non-empty context without knowing the future request. During online inference, the retrieved chunk caches are assembled at their realized positions, and a subset of context tokens is selected for recomputation using the actual query, while the remaining tokens retain their surrogate-conditioned KV entries. Our main contributions are:

- We identify offline cache construction as an underexplored source of error in position-independent KV cache reuse and illustrate that independent construction induces substantial attention-routing discrepancy relative to full prefill.
- We propose SurroKV, a training-free *surrogate-context cache construction* method that constructs reusable chunk caches under a fixed, request-independent context while preserving cache size and the existing online selective-recomputation pipeline. Our theoretical analysis further shows that a fixed surrogate can capture context effects shared across possible future contexts and reduce construction-induced attention-routing discrepancy.
- Extensive experiments are conducted on RULER, LongBench, and HELMET across multiple models and recomputation budgets. Specifically, our method outperforms existing cache-reuse baselines, achieving relative gains of 2.1%–7.8% on RULER, 3.8%–10.5% on LongBench, and 6.4%–12.7% on HELMET at a 10% recomputation budget. The improvements are particularly pronounced under limited recomputation budgets. Moreover, our cache construction benefits existing recomputation strategies without modifying their original procedures.

2 Related Work

Reusable KV caching raises two complementary questions: how a chunk cache is constructed offline, and how it is corrected after multiple cached chunks are assembled at inference time. Existing work on non-prefix reuse has focused primarily on the latter, whereas our work revisits the former.

KV cache reuse. Prefix caching enables exact reuse for shared prefixes (Kwon et al., 2023; Zheng et al., 2024; Yu et al., 2025). Position-independent caching relaxes this constraint by precomputing reusable chunk caches that can be relocated at inference time (Gim et al., 2024). However, independently constructed caches miss contextual interactions, motivating online selective recomputation to reduce reuse error. Existing methods select tokens based on cache deviation (Yao et al., 2025; Yang et al., 2025b), chunk boundaries (Hu et al., 2024), query-aware attention or sparse-query signals (Wang et al., 2026; Yan et al., 2026; Zhang et al., 2026), or learned predictors (Yang et al., 2025a). Other approaches combine attention sparsification with local recomputation (Cao et al., 2026). In contrast, our work focuses on cache construction which is complementary to existing methods.

Context-aware and parallel chunk encoding. A related line of work studies how separately encoded segments differ from sequential processing. Parallel Context Windows (Ratner et al., 2023), APE (Yang et al., 2025c), and Block-Attention (Ma et al., 2025) modify how independently encoded segments are combined or consumed, while prior analyses have highlighted artifacts caused by missing left context, attention sinks, and positional mismatch (Xiao et al., 2024; Gu et al., 2025; Barbero et al., 2025). CacheClip (Yang et al., 2025a) is particularly related in that it also introduces shared context during offline chunk encoding. Our setting differs in that the surrogate context is used only to shape the reusable chunk cache and is discarded after construction; the stored cache has the same size as conventional chunk caching, and no additional context or training is required online.

Orthogonal efficiency techniques. Other work reduces the cost of long-context inference through KV-cache eviction (Zhang et al., 2023; Li et al., 2024; Cai et al., 2024), quantization (Liu et al., 2024), or sparse prefill attention (Jiang et al., 2024). These approaches optimize the amount of cache or attention computation after the context is available, rather than how reusable chunk caches are constructed. They are therefore largely orthogonal to our method and can be combined with it.

3 Background and Motivation

3.1 KV Cache and Reuse in RAG

Transformers process each token through query (Q), key (K), and value (V) representations. Queries interact with the keys of preceding tokens to determine attention weights, while the corresponding values are aggregated to form the attention output. During autoregressive inference, the computed keys and values are stored in the KV cache and reused for subsequent decoding.

Here, we consider a context-before-query RAG layout, where retrieved chunks $x_1, \ldots, x_M$ appear before the user query, possibly following an instruction. To avoid repeated prefill computation, reusable chunk caches are typically constructed independently and placed at their realized positions during inference. However, under full prefill, the cache of a chunk depends on its preceding context. For a reusable chunk x, we denote this realized preceding context by p, which may include the instruction and previously retrieved chunks.

Let $\mathrm{Cache}(x \mid c)$ denote the KV states of x obtained by prefilling $[c, x]$ and retaining only the states associated with x. Full prefill and independent cache construction therefore produce

$$(K_x^{\mathrm{full}}, V_x^{\mathrm{full}}) = \mathrm{Cache}(x \mid p), \qquad (K_x^{\mathrm{ind}}, V_x^{\mathrm{ind}}) = \mathrm{Cache}(x \mid \emptyset). \tag{1}$$

Thus, independent caching makes a particular construction choice: although the realized preceding context is unknown at construction, the reusable chunk is constructed as if that context were empty.

3.2 Mismatch under Independent Cache Construction

We next examine how this construction choice changes the behavior of the reused cache relative to full prefill. For the analysis, we consider a single attention head and omit layer and head indices for clarity. Given a query $q \in \mathbb{R}^d$ and assembled key and value matrices $K, V \in \mathbb{R}^{n \times d}$, attention is

$$z = \frac{Kq}{\sqrt{d}}, \qquad a = \mathrm{softmax}(z), \qquad o = a^\top V. \tag{2}$$

To isolate the direct cache-side effect of construction, we compare independent reuse and full prefill at the same online positions using the same full-prefill query. Let $o^{\mathrm{ind}} = (a^{\mathrm{ind}})^\top V^{\mathrm{ind}}$ and $o^{\mathrm{full}} = (a^{\mathrm{full}})^\top V^{\mathrm{full}}$. Their difference admits the exact decomposition

$$o^{\mathrm{ind}} - o^{\mathrm{full}} = \underbrace{(a^{\mathrm{ind}} - a^{\mathrm{full}})^\top V^{\mathrm{ind}}}_{\text{Attention routing error}} + \underbrace{(a^{\mathrm{full}})^\top (V^{\mathrm{ind}} - V^{\mathrm{full}})}_{\text{Value representation error}}. \tag{3}$$

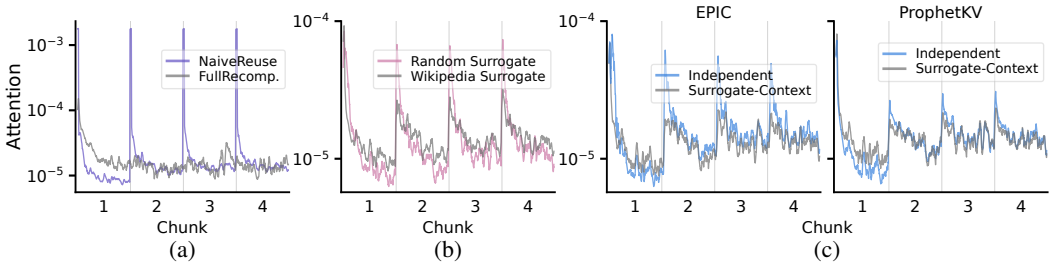

Figure 3: **Query-to-context attention on LongBench HotpotQA** (Llama-3.1-8B-Instruct, 8K). (a) independent reuse and full prefill. (b) surrogate-conditioned caches with random and Wikipedia surrogates, without recomputation. (c) independent and surrogate-conditioned caches with 5% recomputation across two existing token selectors.

The two terms isolate mismatch from attention routing and cached value representations, respectively. We quantify each component by its squared norm, normalized by the squared norm of the full-prefill attention output, and aggregate over query tokens, attention heads, and examples. We report this normalized mean squared error (NMSE) at each Transformer layer. Since the two components are not generally orthogonal, their errors need not sum exactly to the overall error.

Figure 2 shows that independent cache construction introduces substantial attention-routing mismatch relative to full prefill, with routing error dominating across layers. One possible explanation is the absence of preceding context: when encoded in isolation, chunk tokens can attend only within the chunk, potentially concentrating attention on its initial tokens. Prepending a fixed, non-empty context provides additional tokens to attend to during offline construction, altering the attention outputs of chunk tokens and, consequently, their KV representations in subsequent layers. This offers a simple way to introduce preceding-context conditioning without knowing the actual future request.

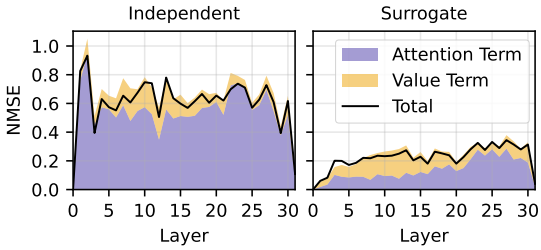

Figure 2: **Layer-wise decomposition of attention output mismatch.** NMSE of the routing and value errors in Eq. 3 on LongBench (Llama-3.1-8B-Instruct, 8K). Black curve represents the overall error, including the cross term.

To examine this intuition, we construct chunk caches with a fixed prefix and evaluate them without online recomputation. As shown in the right panel of Figure 2, this change substantially reduces attention-routing error while leaving value-representation error largely unchanged. These findings indicate that a significant portion of the routing mismatch can be mitigated during offline construction, motivating our surrogate-context construction strategy.

3.3 Understanding Attention Behavior under Cache Reuse

To better understand routing mismatch, we examine query-to-context attention with and without online recomputation in Figure 3. Without recomputation, we compare independent reuse, full prefill, and two surrogate constructions, one using a Wikipedia passage and one using randomly sampled tokens, to isolate the effect of offline construction. At a 5% recomputation budget, we further compare independent and surrogate-conditioned caches across two token selectors, EPIC (Hu et al., 2024) and ProphetKV (Wang et al., 2026). Since full prefill captures the actual cross-chunk interactions, we use its attention profile as a reference: profiles closer to full prefill suggest reduced routing distortion.

Takeaways. Figure 3 highlights three key observations: **1) Independent reuse introduces boundary artifacts.** Independently constructed caches exhibit sharp attention peaks at chunk boundaries, consistent with artificial attention sinks. **2) Surrogate construction mitigates these artifacts.** Surrogate conditioning suppresses boundary peaks and produces attention profiles closer to full prefill, even without recomputation. **3) Surrogate construction complements online recomputation.** Under the same 5% recomputation budget, this benefit persists across existing token selectors, demonstrating its compatibility with different token-selection strategies.

4 SURROGATE-CONDITIONED KV CACHE REUSE FRAMEWORK

4.1 OVERVIEW

The preceding analysis motivates improving reusable KV caches at construction time rather than relying solely on online recomputation. We introduce SurroKV, which integrates offline surrogate conditioning with existing online selective recomputation. Figure 1 presents the two-stage framework. **Stage I: Offline Cache Construction.** Each reusable chunk is encoded once with a fixed surrogate context placed before it. Only the resulting KV states of the chunk are retained, while the surrogate states are discarded, keeping the stored cache query-independent and the same size as a conventional reusable cache. **Stage II: Online Token Selection and Recomputation.** Retrieved caches are assembled at their realized positions, after which a small subset of context tokens is selected and recomputed under the actual request. The remaining tokens retain their surrogate-conditioned KV states. Thus, our method improves offline construction without changing the existing online recomputation procedure.

4.2 SURROGATE-CONDITIONED CACHE CONSTRUCTION

For each reusable chunk x, we prepend a fixed surrogate context s and prefill the sequence $[s, x]$. Following the notation defined in Sec. 3, the resulting chunk cache is

$$(K_x^{\mathrm{sur}}, V_x^{\mathrm{sur}}) = \mathrm{Cache}(x \mid s). \tag{4}$$

The surrogate is shared across chunks and is independent of future queries, retrieved neighbors, and chunk ordering. Only the KV states of x are retained, while those of s are discarded. This allows each chunk to be constructed once offline and reused across requests.

To support position-independent reuse, we store the cached keys in canonical chunk-local coordinates by removing the positional offset introduced by the surrogate. During online inference, the keys are relocated to their realized positions using the standard RoPE transformation. Thus, surrogate conditioning requires neither additional stored KV states nor surrogate processing online.

4.3 QUERY-GUIDED TOKEN SELECTION

While surrogate conditioning reduces construction-induced KV-cache mismatch, its fixed, request-independent context cannot fully capture interactions with the realized query and the specific combination of retrieved chunks. To address this residual mismatch, we adopt selective recomputation (Yao et al., 2025) to refine query-relevant context tokens under the actual request without recomputing the entire context. Specifically, inspired by prior work (Wang et al., 2026; Zhang et al., 2024), we leverage query-to-context attention to further estimate token importance and select relevant context tokens for online recomputation.

Let $\mathcal{X}$ and $\mathcal{Q}$ denote the reusable context tokens and user-query tokens, respectively. After assembling the retrieved caches, we process the query and collect its attention to the cached context. Let $a_{q,t}^{\ell}$ denote the attention weight from query token $q \in \mathcal{Q}$ to context token $t \in \mathcal{X}$ at layer ℓ, averaged over attention heads. We define the token importance score as

$$S(t) = \frac{1}{L|\mathcal{Q}|} \sum_{\ell=1}^{L} \sum_{q \in \mathcal{Q}} a_{q,t}^{\ell}. \tag{5}$$

Given a recomputation ratio r, we select the $k = \lfloor r|\mathcal{X}| \rfloor$ highest-scoring context tokens:

$$\mathcal{R} = \mathbf{TopK}_{t \in \mathcal{X}}\big(S(t), k\big), \tag{6}$$

where $\mathcal{R}$ denotes the recomputation set. We use the same selected set across all Transformer layers to maintain a consistent recomputation path. Token scoring requires only processing the query against the assembled context cache, without fully recomputing the retrieved context.

4.4 ONLINE TOKEN RECOMPUTATION

Given the selected set $\mathcal{R}$, we recompute these context tokens across all Transformer layers using the assembled cache, while directly reusing the surrogate-conditioned KV states of unselected tokens. At each layer, the selected tokens incorporate information from their realized preceding context and produce updated key and value states, which replace the corresponding cached entries:

$$(K_t^{\ell}, V_t^{\ell}) = \begin{cases} (K_t^{\ell,\mathrm{rec}}, V_t^{\ell,\mathrm{rec}}), & t \in \mathcal{R}, \\ (K_t^{\ell,\mathrm{sur}}, V_t^{\ell,\mathrm{sur}}), & t \notin \mathcal{R}. \end{cases} \tag{7}$$

The updated representations of selected tokens are propagated through subsequent layers, progressively refining their KV states without recomputing the entire context.

5 Understanding Surrogate-Context Cache Construction

In this section, we further explain why a fixed surrogate can improve reusable KV states even when the realized future context is unknown. Our analysis addresses three main questions: **a)** what construction effects can be shared across possible future contexts? [Theorem 5.1], **b)** how offline cache changes affect attention routing at reuse time? [Theorem 5.2], and **c)** when a particular surrogate improves upon independent construction? [Theorem 5.3]. Proofs are provided in Appendix B.

5.1 Construction Effects and Shared Future Context

We first examine how a preceding context changes the representation of a reusable chunk. Under standard RoPE attention, prepending a context preserves the relative positions and first-layer attention logits among chunk tokens, while introducing additional tokens that they can attend to. We characterize the first-layer construction effect by separating the attention allocated to the preceding context from the representation change it introduces. The resulting attention output and its change relative to independent construction can be expressed as

$$\begin{aligned} &\textbf{Attention Output:} \quad o_i(c) = (1-\rho_i(c))o_i(\emptyset) + \rho_i(c)\mu_i(c), \\ &\textbf{Construction Effect:} \quad g_i(c) \triangleq o_i(c) - o_i(\emptyset) = \underbrace{\rho_i(c)}_{\text{attention mass}} \underbrace{\big(\mu_i(c) - o_i(\emptyset)\big)}_{\text{representation shift}}. \end{aligned} \tag{8}$$

where $\rho_i(c)$ denotes the attention mass assigned by token i to context c, and $\mu_i(c)$ its normalized attention-weighted value representation. The remaining attention mass is distributed over chunk tokens in the same proportions as under independent construction. Equation (8) shows that the construction effect depends on both the attention mass assigned to the preceding context and the representation shift it introduces. Extending a fixed prefix on the left increases its first-layer attention mass, but a larger induced change is not necessarily more accurate. An effective surrogate must produce a correction with an appropriate direction and magnitude; excessive conditioning may instead increase the mismatch.

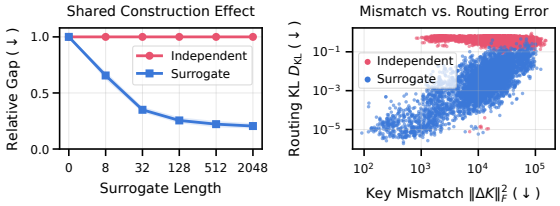

Figure 4: **Empirical analysis of surrogate construction** on LongBench HotpotQA (Llama-3.1-8B-Instruct, 8K). **Left:** Normalized gap to the mean construction effect over eight preceding contexts (independent construction=1). **Right:** Routing KL versus key-cache mismatch for independent and surrogate-conditioned (1024-token) caches, with 5,000 sampled points per method.

Since the realized preceding context is unknown during offline construction, we next characterize and analyze the shared effect that a fixed surrogate should approximate via the following theorem. Appendix B.3 provides further analysis of the preferred conditioning strength and surrogate length.

Theorem 5.1 (**Future-context effect decomposition**). *Let p denote the unknown future preceding context that will be realized at reuse time and $\bar{g}_i = \mathbb{E}_p[g_i(p)]$. For any fixed surrogate s,*

$$\mathbb{E}_p\|g_i(s) - g_i(p)\|_2^2 = \underbrace{\mathbb{E}_p\|g_i(p) - \bar{g}_i\|_2^2}_{\textit{future-specific variation}} + \underbrace{\|g_i(s) - \bar{g}_i\|_2^2}_{\textit{surrogate gap}}. \tag{9}$$

The first term captures variation across future contexts that no fixed surrogate can eliminate, while the second measures how closely the surrogate approximates their mean construction effect. Since independent construction induces no additional effect, $g_i(\emptyset) = 0$, surrogate conditioning reduces the expected construction-effect mismatch whenever $\|g_i(s) - \bar{g}_i\|_2^2 < \|\bar{g}_i\|_2^2$. Thus, a fixed surrogate can reduce expected mismatch without predicting the realized preceding context, provided its induced effect is closer to the mean than independent construction. Figure 4 (left) shows that surrogate construction reduces the gap to the mean construction effect, with larger surrogates generally yielding smaller gaps.

5.2 From Cache Construction to Routing

The first-layer construction effect in Equation (8) changes the attention outputs of chunk tokens and, consequently, their hidden representations. Under standard RoPE attention, the first-layer chunk keys and values remain unchanged in canonical coordinates, but these representation changes affect the KV states computed at subsequent layers. After accounting for positional differences, let

$G^\ell(p) = K^\ell(p) - K^\ell(\emptyset)$. The same decomposition as Theorem 5.1 applies in key space, separating a shared cache correction from future-specific variation.

We next establish how these key-cache differences affect attention routing at reuse time. After placing all caches at their realized positions, let $z^\bullet = K^\bullet q/\sqrt{d}$ and $a^\bullet = \mathrm{softmax}(z^\bullet)$ for $\bullet \in \{\mathrm{ind}, \mathrm{sur}, \mathrm{full}\}$, where q is fixed to its full-prefill value.

Theorem 5.2 (**Cache-to-routing bound**). *Define routing discrepancy as* $\mathcal{L}(z) = D_{\mathrm{KL}}(a^{\mathrm{full}} \| \mathrm{softmax}(z))$. *With identical attention masks and realized positions, and non-reusable keys matching full prefill, the error bound is* $\mathcal{L}(z^{\mathrm{sur}}) \le \frac{\|q\|_2^2}{4d} \|K^{\mathrm{sur}} - K^{\mathrm{full}}\|_F^2$.

The bound connects offline cache construction to online routing: reducing key-cache mismatch tightens the upper bound on routing discrepancy. Under RoPE, this cache distance is preserved when corresponding keys undergo the same positional relocation. Figure 4 (right) shows that surrogate-conditioned caches exhibit lower key-cache mismatch and routing discrepancy than independently constructed caches. Whether a particular surrogate actually reduces routing error is examined below.

5.3 When Does Surrogate Construction Improve Routing?

The previous bound connects key-cache mismatch to routing discrepancy, but does not establish whether a particular surrogate actually improves over independent construction. To characterize when such an improvement occurs, we derive the exact change in routing discrepancy and identify the condition under which surrogate conditioning reduces routing error.

Theorem 5.3 (**Exact routing effect**). *Let* $\Delta z^{\mathrm{sur}} = z^{\mathrm{sur}} - z^{\mathrm{ind}}$ *and routing discrepancy as* $\mathcal{L}(z) = D_{\mathrm{KL}}(a^{\mathrm{full}} \| \mathrm{softmax}(z))$. *Then*

$$\mathcal{L}(z^{\mathrm{sur}}) - \mathcal{L}(z^{\mathrm{ind}}) = \underbrace{-\left\langle a^{\mathrm{full}} - a^{\mathrm{ind}}, \Delta z^{\mathrm{sur}} \right\rangle}_{A^{\mathrm{sur}}:\ \textit{alignment benefit}} + \underbrace{D_{\mathrm{KL}}(a^{\mathrm{ind}} \| a^{\mathrm{sur}})}_{C^{\mathrm{sur}}:\ \textit{routing shift}}. \tag{10}$$

Hence, surrogate construction improves routing if and only if $A^{\mathrm{sur}} > C^{\mathrm{sur}}$.

The result identifies two competing effects: the surrogate-induced change must align with full-prefill behavior while outweighing the routing shift it introduces. To interpret this benefit, we express it in terms of the full-prefill logit change and the geometry of softmax attention.

Proposition 5.1 (**Interpreting the alignment benefit**). *Let* $\Delta z^{\mathrm{full}} = z^{\mathrm{full}} - z^{\mathrm{ind}}$ *and* $\overline{H}^{\mathrm{full}} = \int_0^1 H(z^{\mathrm{ind}} + t\Delta z^{\mathrm{full}})\, dt$, *where* $H(z) = \mathrm{diag}(a) - aa^\top$ *for* $a = \mathrm{softmax}(z)$. *Then*

$$A^{\mathrm{sur}} = (\Delta z^{\mathrm{sur}})^\top \overline{H}^{\mathrm{full}} \Delta z^{\mathrm{full}}. \tag{11}$$

Since $\overline{H}^{\mathrm{full}} \succeq 0$, this expression measures alignment in an attention-relevant geometry. Its sign nevertheless depends on the surrogate-induced direction: an arbitrary surrogate need not yield a positive alignment benefit. Importantly, exact recovery of the full-prefill change via surrogate is unnecessary in practice. A partial correction along the full-prefill direction is sufficient to reduce routing discrepancy.

Corollary 5.3.1 (**Partial full-prefill correction is sufficient**). *If* $\Delta z^{\mathrm{sur}} = \alpha \Delta z^{\mathrm{full}}$ *with* $0 < \alpha \le 1$, *then* $\mathcal{L}(z^{\mathrm{sur}}) \le (1-\alpha)\mathcal{L}(z^{\mathrm{ind}})$. *Thus, any nonzero partial correction along the full-prefill direction reduces the routing discrepancy whenever the initial discrepancy is nonzero.*

Multi-chunk RAG. The construction analysis applies separately to each chunk, while online attention routing is coupled across chunks through a shared softmax. Our routing results therefore can extend to the assembled context, as detailed in Appendix B.9.

6 Experiments

6.1 Experimental Setup

Base models and benchmarks. In this paper, we select three representative LLMs as our base models, including Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct, and Qwen2.5-14B-Instruct (Grattafiori et al., 2024; Yang et al., 2024). We evaluate on RULER (Hsieh et al., 2024) and LongBench (Bai et al., 2024), and on the RAG tasks of HELMET (Yen et al., 2025) (Appendix A.5). RULER is a synthetic benchmark generated at specified context lengths. We use 8K for the main results and report 16K results, additional tasks, and detailed configurations in Appendix A.

Table 1: Performance comparison of different models under a 10% recomputation budget. SurroKV (n) denotes our method with an n-token surrogate prefix. The best and second-best results among cache-reuse methods are highlighted in **boldface** and underline, respectively.

	RULER							LongBench							
Methods	**MK1**	**MQ**	**MV**	**QA1**	**QA2**	**VT**	**Avg.**	**WQA**	**TQA**	**HQA**	**MQue**	**QMSum**	**PR-en**	**PR-zh**	**Avg.**
Base model: Llama3.2-3B-Inst.															
FullRecomp.	98.6	98.7	98.1	68.4	51.0	83.7	83.1	39.7	32.8	53.6	25.1	24.4	98.0	21.5	42.2
NaiveReuse	33.4	32.2	23.9	15.0	14.8	21.9	23.5	18.2	35.6	10.1	4.1	16.8	3.0	3.9	13.1
EPIC	66.4	70.8	45.3	55.8	40.6	58.8	56.3	35.6	35.7	41.3	17.4	21.8	14.5	6.0	24.6
CacheBlend	71.4	76.4	61.8	58.2	41.4	71.5	63.5	34.7	33.9	44.5	18.3	21.3	33.0	8.0	27.7
KVShare	70.4	78.3	65.1	58.2	38.2	60.2	61.7	36.6	**36.0**	44.5	19.7	21.8	25.5	11.5	27.9
ProphetKV	84.8	95.5	88.2	57.4	43.2	79.2	74.7	37.7	23.9	46.3	18.2	21.8	84.0	15.0	35.3
SurroKV (512)	94.2	**99.0**	93.0	63.0	47.4	82.1	79.8	38.2	24.7	47.7	21.7	22.2	90.5	**20.0**	37.9
SurroKV (1024)	**94.8**	98.9	**93.8**	**64.2**	**48.6**	**82.7**	**80.5**	**40.4**	24.7	**49.1**	**22.7**	**22.4**	**94.0**	**20.0**	**39.0**
Base model: Llama3.1-8B-Inst.															
FullRecomp.	99.8	99.8	99.6	81.4	59.0	99.9	89.9	47.7	90.8	57.8	30.4	25.4	100.0	97.0	64.1
NaiveReuse	51.0	37.3	31.8	47.4	31.6	66.8	44.3	32.9	77.5	36.5	17.1	20.9	8.2	9.2	28.9
EPIC	76.2	66.2	44.7	77.0	49.6	62.2	62.7	40.0	89.8	47.1	22.6	23.2	43.0	41.2	43.8
CacheBlend	78.2	73.9	51.1	76.6	48.4	68.9	66.2	42.7	90.7	46.3	24.7	23.4	49.5	64.0	48.8
KVShare	79.0	74.2	52.4	75.4	46.6	82.4	68.3	40.0	90.4	46.4	25.6	23.4	49.5	70.2	49.3
ProphetKV	96.8	96.8	92.3	78.8	53.0	**86.3**	84.0	45.2	90.0	49.6	**28.4**	23.2	87.5	94.7	59.8
SurroKV (512)	98.2	**98.2**	**97.1**	**80.6**	**54.4**	**86.3**	**85.8**	46.0	89.4	52.1	26.3	23.2	96.5	**95.5**	61.3
SurroKV (1024)	**98.6**	**98.2**	97.0	80.2	54.2	85.7	85.7	**46.1**	**91.4**	**52.9**	27.7	**23.9**	**97.5**	**95.5**	**62.1**
Base model: Qwen2.5-14B-Inst.															
FullRecomp.	99.8	99.9	95.2	78.8	61.4	100.0	89.2	60.0	83.4	63.1	38.4	24.1	97.4	91.2	65.4
NaiveReuse	68.2	56.1	30.4	59.4	31.0	40.6	47.6	18.5	33.2	13.9	3.8	19.3	13.0	18.4	17.2
EPIC	76.4	83.9	38.9	73.8	50.4	21.4	57.5	43.1	71.0	53.5	30.6	22.0	23.8	17.9	37.4
CacheBlend	89.4	88.4	48.9	73.8	52.6	22.2	62.6	43.9	73.5	55.0	31.8	21.9	47.2	25.2	42.6
KVShare	88.2	87.4	48.5	74.4	50.2	51.2	66.7	38.0	68.1	49.2	29.2	21.6	52.5	29.3	41.1
ProphetKV	96.6	92.2	77.9	76.4	55.2	**99.7**	83.0	48.3	81.0	54.3	36.1	22.2	94.0	49.2	55.0
SurroKV (512)	99.2	97.3	83.5	**78.4**	59.0	99.4	86.1	47.9	**85.2**	**60.2**	**38.0**	**23.4**	94.3	**59.2**	**58.3**
SurroKV (1024)	**99.8**	**97.6**	**85.2**	**78.4**	**59.6**	99.6	**86.7**	**48.9**	84.1	59.6	37.8	23.1	**95.8**	55.9	57.9

Baselines. Several state-of-the-art methods are adopted as our baselines for performance comparisons: 1) FULL RECOMPUTATION prefills the entire prompt, recomputing all context tokens; 2) NAIVE REUSE does not recompute any context token; 3) EPIC (Hu et al., 2024) recomputes the first tokens of every chunk; 4) CACHEBLEND (Yao et al., 2025) recomputes the tokens with the largest value deviation at the second layer; 5) KVSHARE (Yang et al., 2025b) recomputes the tokens with the largest attention-weighted value deviation; 6) PROPHETKV (Wang et al., 2026) selects tokens by the attention they receive from the user query, averaged over layers and heads.

Implementation. We implement all methods in HuggingFace Transformers (Wolf et al., 2019) using bfloat16 and PyTorch SDPA (Paszke et al., 2019), without method-specific kernels. All experiments run on NVIDIA L40S GPUs (48 GB). We use a fixed, task-independent Wikipedia passage shared across chunks as the surrogate context, evaluating SurroKV (512) and SurroKV (1024) with corresponding prefix lengths. Unless otherwise specified, we use a 10% recomputation budget, report official task metrics on a 0–100 scale, and measure latency as time to first token (TTFT).

6.2 MAIN RESULTS

Our method consistently outperforms existing baselines. Table 1 compares all methods on three models under a 10% recomputation budget. As shown in Table 1, our proposed method outperforms all baselines, with gains of 2.1%–7.8% on RULER and 3.8%–10.5% on LongBench. These results support surrogate-context construction over independent construction.

Efficiency. Table 2 reports the average TTFT (s) over examples on Llama-3.1-8B-Instruct, using a 10% recomputation budget and batch size of 1. We evaluate RULER at 8K, 16K, and 32K to examine latency across context lengths, alongside LongBench. Offline-constructed caches reside in GPU memory, and TTFT includes token selection and recomputation. Our method matches ProphetKV in TTFT and is faster than several baselines, with negligible online overhead. Moreover, offline construction costs are detailed in Appendix A.11.

6.3 ABLATION STUDY

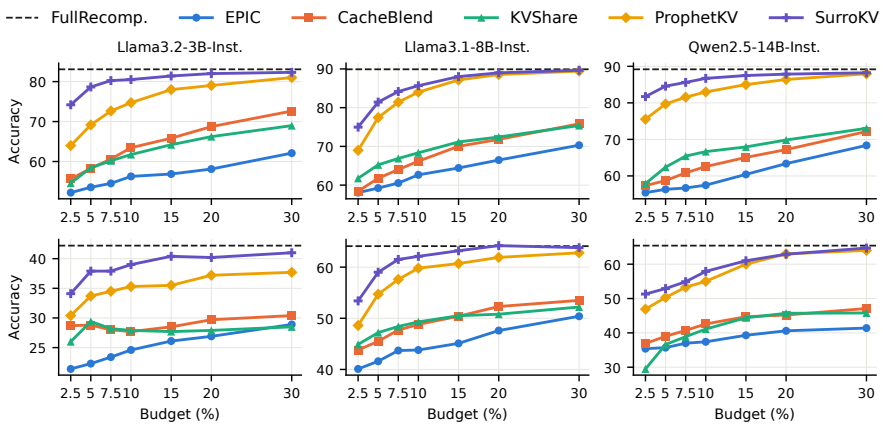

Figure 5: Average performance comparison across different models and recomputation budgets (**Top**: RULER; **Bottom**: LongBench). By default, the surrogate context length is 1024.

Table 3: Performance comparison without / with surrogate-based offline cache construction (1024-token prefix). We evaluate three budgets: 5%, 10%, and 15%. Results are presented as (without surrogate / with surrogate). Subscripts indicate score changes (blue: gains; red: losses).

	RULER			LongBench		
Selector	**5%**	**10%**	**15%**	**5%**	**10%**	**15%**
EPIC	59.2 / 64.1 $_{+4.9}$	62.7 / 66.4 $_{+3.7}$	64.4 / 67.9 $_{+3.5}$	41.6 / 42.7 $_{+1.1}$	43.8 / 44.5 $_{+0.7}$	45.1 / 45.6 $_{+0.5}$
CacheBlend	61.8 / 68.5 $_{+6.7}$	66.2 / 70.1 $_{+3.9}$	70.0 / 71.0 $_{+1.0}$	45.5 / 46.2 $_{+0.7}$	48.8 / 48.1 $_{-0.7}$	50.4 / 49.9 $_{-0.5}$
KVShare	65.2 / 70.1 $_{+4.9}$	68.3 / 72.0 $_{+3.7}$	71.1 / 73.4 $_{+2.3}$	47.2 / 50.7 $_{+3.5}$	49.3 / 51.5 $_{+2.2}$	50.5 / 52.3 $_{+1.8}$
ProphetKV	77.4 / 81.4 $_{+4.0}$	84.0 / 85.7 $_{+1.7}$	87.1 / 88.0 $_{+0.9}$	54.7 / 59.0 $_{+4.3}$	59.8 / 62.1 $_{+2.3}$	60.7 / 63.2 $_{+2.5}$

Performance under different recomputation budgets. Figure 5 compares all methods across recomputation budgets from 2.5% to 30% on three base models. Our approach achieves strong performance on both RULER and LongBench, with particularly pronounced gains at low recomputation budgets. These results highlight the benefit of better initial KV caches when online recomputation is limited. More detailed results are provided in Appendix A.7.

Table 2: Latency (TTFT (s)) Comparison.

Method	**RULER**			**Long Bench**
	8K	**16K**	**32K**	
FullRecomp.	0.79	1.85	4.45	1.26
NaiveReuse	0.04	0.06	0.10	0.07
EPIC	0.12	0.31	0.92	0.23
CacheBlend	0.18	0.43	1.21	0.31
KVShare	0.28	0.89	3.14	0.59
ProphetKV	0.17	0.38	1.07	0.40
SurroKV	0.17	0.38	1.07	0.40

Surrogate cache construction benefits recomputation methods. Here, we integrate our surrogate construction with four baselines to highlight the broad applicability of our approach. Table 3 shows that surrogate construction yields substantial performance improvements in most evaluated settings. These gains are achieved without modifying their online selection or recomputation procedures, demonstrating the complementary nature of our offline construction. Besides, additional ablations on surrogate prefix length and chunk size are provided in Appendices A.6 and A.8.

7 Conclusion

In this work, we revisit how reusable KV caches should be constructed when their future context is unknown. Our analysis shows that independent construction introduces substantial mismatch from full prefill, particularly in attention routing. We therefore propose *surrogate-context cache construction*, which constructs each reusable chunk behind a fixed, request-independent context while preserving cache size and the existing online selective-recomputation pipeline. Both our analysis and experiments show that this simple offline modification can substantially improve cache reuse, especially under limited recomputation budgets and longer contexts, without additional online latency. Improvements across different token selectors further demonstrate the compatibility of our construction with existing online methods. These results establish cache construction as an important design dimension for position-independent KV reuse.

AI USE STATEMENT

In this work, we used generative AI tools to assist with code implementation, translation and drafting of parts of the manuscript, editing for readability, and figure creation. All AI-assisted code, text, and figures were reviewed by the authors. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. It studies the efficiency of KV cache reuse in LLM inference and uses only publicly available benchmarks and open-weight models. No human subjects, personally identifiable information, or private data are involved.

REPRODUCIBILITY STATEMENT

Section 6.1 and Appendix A.1 describe the models, benchmarks, recomputation budgets, and evaluation protocol. The construction of the surrogate context is specified in Section 4.2. Proofs of all theoretical results are given in Appendix B.

REFERENCES

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3119–3137, 2024.

Federico Barbero, Alvaro Arroyo, Xiangming Gu, Christos Perivolaropoulos, Michael Bronstein, Petar Veličković, and Razvan Pascanu. Why do llms attend to the first token? *arXiv preprint arXiv:2504.02732*, 2025.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Ziyi Cao, Qingyi Si, Jingbin Zhang, and Bingquan Liu. Sparse attention across multiple-context kv cache. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30165–30173, 2026.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. *Proceedings of Machine Learning and Systems*, 6:325–338, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. In *International Conference on Learning Representations*, volume 2025, pp. 97114–97144, 2025.

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? *arXiv preprint arXiv:2404.06654*, 2024.

Junhao Hu, Wenrui Huang, Weidong Wang, Haoyi Wang, Tiancheng Hu, Qin Zhang, Hao Feng, Xusheng Chen, Yizhou Shan, and Tao Xie. Epic: Efficient position-independent caching for serving large language models. *arXiv preprint arXiv:2410.15332*, 2024.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 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. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

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*, 2024.

Dongyang Ma, Yan Wang, and Tian Lan. Block-attention for efficient prefilling. In *International Conference on Learning Representations*, volume 2025, pp. 63774–63788, 2025.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. Parallel context windows for large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6383–6402, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

Jianxin Yan, Wangze Ni, Zhenxin Li, Jiabao Jin, Zhitao Shen, Haoyang Li, Jia Zhu, Peng Cheng, Xuemin Lin, Lei Chen, et al. Qcfuse: Query-aware cache fusion via compressed view for efficient rag serving. *arXiv preprint arXiv:2606.05875*, 2026.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Bin Yang, Qiuyu Leng, Jun Zeng, and Zhenhua Wu. Cacheclip: Accelerating rag with effective kv cache reuse. *arXiv preprint arXiv:2510.10129*, 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.

Xinyu Yang, Tianqi Chen, and Beidi Chen. Ape: Faster and longer context-augmented generation via adaptive parallel encoding. In *International Conference on Learning Representations*, volume 2025, pp. 82887–82907, 2025c.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018.

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.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. Helmet: How to evaluate long-context models effectively and thoroughly. In *International Conference on Learning Representations*, volume 2025, pp. 98914–98965, 2025.

Lingfan Yu, Jinkun Lin, and Jinyang Li. Stateful large language model serving with pensieve. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 144–158, 2025.

Quqing Zhang, Kai Chen, Ning Liao, Zehao Lin, Bo Tang, Feiyu Xiong, Zhiyu Li, and Xiaoxing Wang. Sparsex: Efficient segment-level kv cache sharing for interleaved llm serving. *arXiv preprint arXiv:2606.01751*, 2026.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*, 2024.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in neural information processing systems*, 36:34661–34710, 2023.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

APPENDIX

ORGANIZATION OF THE APPENDIX

The appendix is organized as follows.

- **Section A:** Provides experimental details, additional results, and ablation studies.
  - **Section A.1:** Describes benchmark protocols, task selection, and metrics.
  - **Section A.2:** Gives the surrogate context used in all experiments.
  - **Section A.3:** Evaluates RULER performance at 16K.
  - **Section A.4:** Reports results on additional LongBench tasks.
  - **Section A.5:** Reports results on the RAG tasks of HELMET.
  - **Section A.6:** Studies the effect of surrogate prefix length.
  - **Section A.7:** Provides detailed results across recomputation budgets.
  - **Section A.8:** Examines the effect of chunk size on cache reuse.
  - **Section A.9:** Analyzes attention profiles across benchmark tasks.
  - **Section A.10:** Evaluates surrogate construction with other token selectors.
  - **Section A.11:** Measures offline construction time and GPU memory.
- **Section B:** Provides detailed derivations, proofs, and extensions of the theoretical analysis.
  - **Section B.1:** Derives the first-layer effect of surrogate conditioning.
  - **Section B.2:** Proves the future-context effect decomposition theorem.
  - **Section B.3:** Analyzes surrogate direction, strength, and prefix length.
  - **Section B.4:** Extends construction effects to deeper layers and reusable KV states.
  - **Section B.5:** Proves the bound relating cache mismatch to routing error.
  - **Section B.6:** Proves the exact criterion for improved attention routing.
  - **Section B.7:** Characterizes alignment with the full-prefill routing change.
  - **Section B.8:** Proves that partial full-prefill correction reduces routing error.
  - **Section B.9:** Extends the theoretical analysis to multi-chunk RAG.

A EXPERIMENTAL DETAILS AND ADDITIONAL RESULTS

A.1 BENCHMARK DETAILS

We follow the official prompts, generation limits, and evaluation metrics of RULER (Hsieh et al., 2024) and LongBench (Bai et al., 2024). Averages over RULER and LongBench in the figures and tables use the tasks of Table 1.

RULER. We generate 500 examples per task at 8K and 16K using each model's tokenizer, with fixed seeds and templates. Table 1 reports a selected set of six tasks: MK1, MQ, MV, QA1, QA2, and VT. Table 5 reports the same six tasks at 16K. The Avg. column is computed over the task columns reported in each table. We use the official string-matching metrics: the fraction of reference strings recovered for the retrieval tasks, and whether any reference string is recovered for QA1 and QA2.

LongBench. Table 1 reports seven selected tasks: 2WikiMQA, TriviaQA, HotpotQA, MuSiQue, QMSum, and PassageRetrieval in English and Chinese. Additional LongBench tasks are reported in Table 6. We use the first 200 examples per task, except MultiFieldQA-en, which contains 150 examples. TREC and LSHT are excluded. We use F1 for WQA, TQA, HQA, MQue, MFQA-en, MFQA-zh, Qasper, and NQA; ROUGE-L for QMSum, DuReader, SAMSum, MultiNews, VCSUM, and GovReport; accuracy (EM) for PR-en, PR-zh, and PassCnt; and edit similarity for LCC and RepoBench.

HELMET. We follow the official HELMET (Yen et al., 2025) setup for its four RAG tasks: NQ, TriviaQA, HotpotQA, and PopQA. We sample 100 questions per task at 16K and 32K. Each NQ, TriviaQA, and PopQA question is evaluated with the gold passage at six depths, and each HotpotQA question with three passage orderings. This gives 600 examples for NQ, TriviaQA, and PopQA, and 300 for HotpotQA. We report substring exact match (SubEM).

A.2 Implementation Details

Surrogate context. The surrogate is a fixed passage of Wikipedia paragraphs from the HotpotQA distractor validation set (Yang et al., 2018); none of it overlaps with the evaluated examples. Table 4 lists the full text.

A.3 Performance at Extended Context Length

To evaluate our approach in a longer-context setting, we extend the RULER evaluation to 16K using Llama-3.1-8B-Instruct and Qwen2.5-14B-Instruct. As shown in Table 5, SurroKV achieves the highest average accuracy among cache reuse methods on both models at a 10% recomputation budget.

A.4 Performance on Additional LongBench Tasks

We extend our evaluation to additional LongBench tasks on Llama-3.2-3B-Instruct and Llama-3.1-8B-Instruct. Table 6 reports results at a 10% recomputation budget. SurroKV outperforms all evaluated cache reuse baselines on Llama-3.2-3B-Instruct and achieves competitive average performance on Llama-3.1-8B-Instruct.

A.5 Performance on RAG Benchmarks

We further evaluate our approach on the four RAG tasks of HELMET (Yen et al., 2025) at 16K and 32K. As shown in Tables 7 and 8, SurroKV achieves the highest average accuracy among cache reuse methods on all models and context lengths at both 10% and 20% recomputation budgets.

A.6 Effect of Surrogate Prefix Length

We evaluate surrogate prefix lengths ranging from 8 to 2048 tokens across different recomputation budgets on Llama-3.1-8B-Instruct at 8K. As shown in Figure 6, all evaluated prefix lengths improve upon independent chunk construction. On RULER, even a short prefix captures a substantial portion of the improvement, while LongBench generally benefits from longer prefixes, with gains extending to approximately 1024 tokens. Table 9 provides the corresponding numerical results. A prefix length of zero denotes independent chunk construction without a surrogate context.

A.7 Detailed Results Across Recomputation Budgets

Table 10 provides the detailed numerical results underlying Figure 5, reporting average performance on RULER and LongBench across three models and different recomputation budgets. SurroKV (n) uses an n-token surrogate prefix. Full recomputation and naive reuse are included as budget-independent reference points.

A.8 Effect of Chunk Size

We evaluate chunk sizes of 256, 512, and 1024 tokens on Llama-3.1-8B-Instruct at 8K. As shown in Table 11, performance generally improves with larger chunks, as independent construction preserves more context within each chunk. Surrogate-based construction provides larger gains for smaller chunks, where the absence of preceding context causes greater cache mismatch. For example, at a 10% recomputation budget, SurroKV (1024) improves the RULER average over ProphetKV by 5.7, 1.7, and 1.0 points for chunk sizes of 256, 512, and 1024, respectively. With 1024-token chunks and a 20% recomputation budget, both methods approach full recomputation accuracy.

Table 4: The surrogate context used by SurroKV (1024). SurroKV (512) uses its last 512 tokens.

Surrogate context (1,024 tokens)

Elliott Lester is an English film and television director, best known for directing the film "Blitz". He made his directing debut in 2006 with "Love Is the Drug", and his latest film, "Aftermath", was released on April 4, 2017. Anthony C. Ferrante is an American film director, producer, and writer, known for directing the "Sharknado" series, the 2017 thriller Forgotten Evil and the 2005 ghost story "Boo", which was his feature film writing and directing debut. Sayo Yamamoto (山本 沙代 "Yamamoto Sayo"; born April 13, 1977) is a Japanese anime director. She is known for directing the anime series "Michiko & Hatchin" and "Yuri on Ice" and the critically acclaimed anime series "". After graduating from the College of Art and Design in Tokyo, she began work at Studio Madhouse, where she had her directing debut at age 25. Who's the Man? is a 1993 thriller comedy film, directed by Ted Demme, in his feature film directing debut. The film stars "Yo! MTV Raps" hosts Doctor Dré and Ed Lover as its two main protagonists., it features dozens of cameo appearances from some of the top rap/hip-hop acts of the time, including (though not limited to) Busta Rhymes, Bushwick Bill, Guru, Eric B., House of Pain, Ice-T, Kris Kross, Queen Latifah, KRS-One and Run-D.M.C.. This film is also the feature film debut of Terrence Howard. Forbidden Quest () is a 2006 South Korean period drama film about a scholar during the Joseon Dynasty who begins to write erotic novels, and becomes the lover of the King's favorite concubine. Born in Tokyo, Japan, he attended Waseda University but left before graduating due to his involvement in a left-wing theater troupe. He joined P.C.L. (a precursor to Toho) in 1933 and began working as an assistant director to Kajirō Yamamoto alongside his longtime friend, acclaimed Japanese filmmaker, Akira Kurosawa. He made his feature film directing debut in 1947 with "Snow Trail," which was written by Kurosawa. "Snow Trail" starred Toshirō Mifune in his film debut and actress Setsuko Wakayama. It helped establish Taniguchi's reputation for action film. Obsessed | Hungry Horse is a chain of 225 pub-restaurants in England, Wales and Scotland, owned by Greene King Brewery. It was founded in 1995, and promotes itself as offering low cost meals for families and groups. Klenger Burger is an Indonesian restaurant chain based in Jakarta, operated by PT Kinarya Anak Negeri. The restaurant chain served halal hamburgers designed to suit the Indonesian palate. The company sold a variety of burgers, including a 25 cm Raja Klenger (King Klenger) designed to serve seven people. It also sold pizza, grilled and fried duck and chicken, and Indonesian snacks such as rempeyek and emping. VIP's, alternatively written Vip's, is a defunct restaurant chain in the Western United States that operated from 1968 until the late 1980s, based in Salem, Oregon. With more than 50 locations, it was once the largest restaurant chain based in Oregon. It was a Denny's-style restaurant, a type that was commonly known at that time as a "coffee shop" but is now more commonly known as a casual dining restaurant. Most restaurants were located near freeways and were open 24 hours. At its peak, the chain had locations in five states: Oregon, Washington, Idaho, Nevada and northern California. Fazoli's is an Italian-American fast casual restaurant chain based out of Lexington, Kentucky. It was founded in 1988 and is now owned by Seed Restaurant Group, Inc. Today, there are more than 200 Fazoli's located nationwide with plans to expand overseas. The restaurant chain specializes in Italian cuisine and dishes. Carl Howard is the company's president and CEO. Greene King is the UK's largest pub retailer and brewer. It is based in Bury St Edmunds, Suffolk, England. The company owns pubs, restaurants and hotels. It is listed on the London Stock Exchange and is a constituent of the FTSE 250 Index. Todai is a restaurant chain based in the United States. As of 2016, the chain had over 19 restaurants in the United States, 7 restaurants in South Korea, one restaurant in Hong Kong, one restaurant in Canada, one restaurant in Indonesia, one restaurant in Singapore and one restaurant in Malaysia. Chefette Restaurants is the largest fast food restaurant chain based in the Caribbean island nation of Barbados. Currently operating throughout the island in 14 locations, Chefette is known for its broasted chicken meals as well as a local cur

A.9 Attention Profiles Across Benchmark Tasks

Figure 7 extends the attention analysis in Figure 3 to RULER and LongBench tasks. Across these tasks, independent chunk construction exhibits pronounced attention peaks near chunk boundaries. Surrogate-based construction substantially attenuates these peaks, producing attention profiles that more closely resemble those of full recomputation.

A.10 Full Results Across Token Selectors

Table 12 extends Table 3 with additional recomputation budgets. We evaluate EPIC, CacheBlend, KVShare, and ProphetKV using independently constructed caches and caches constructed with a 1024-token surrogate prefix, while keeping their online procedures unchanged. The results provide a more detailed comparison of surrogate-based construction across token selection strategies.

Table 5: Performance on RULER at 16K using Llama-3.1-8B-Instruct and Qwen2.5-14B-Instruct with a 10% recomputation budget. SurroKV (n) denotes a surrogate prefix of n tokens. The task set and Avg. follow Table 1. The best and second-best results among cache reuse methods within each model are highlighted in **boldface** and underline, respectively.

Methods	MK1	MQ	MV	QA1	QA2	VT	Avg.
Base model: Llama3.1-8B-Inst.							
FullRecomp.	100.0	99.9	99.1	80.4	56.0	99.7	89.2
NaiveReuse	44.2	34.2	25.7	33.0	21.4	52.8	35.2
EPIC	66.8	61.7	38.5	73.2	40.6	50.2	55.2
CacheBlend	67.2	68.0	46.6	73.2	42.6	58.7	59.4
KVShare	67.8	66.6	48.2	73.4	40.8	74.2	61.8
ProphetKV	86.2	87.4	83.9	76.8	44.4	68.5	74.5
SurroKV (512)	93.8	**96.2**	92.8	**81.0**	48.4	**78.1**	81.7
SurroKV (1024)	**94.4**	96.0	**94.2**	80.4	**50.2**	76.5	**81.9**
Base model: Qwen2.5-14B-Inst.							
FullRecomp.	99.6	99.9	92.9	73.6	59.2	100.0	87.5
NaiveReuse	55.8	43.8	28.5	33.6	21.2	28.5	35.2
EPIC	76.6	82.0	34.4	57.0	43.0	20.4	52.2
CacheBlend	84.6	84.7	41.1	56.6	46.2	28.4	56.9
KVShare	81.4	81.0	37.6	58.4	41.2	43.3	57.2
ProphetKV	90.2	88.9	73.7	67.4	50.0	96.6	77.8
SurroKV (512)	97.4	96.0	83.7	69.6	53.6	97.4	82.9
SurroKV (1024)	**98.2**	**96.6**	**86.0**	**71.8**	**56.8**	**97.9**	**84.5**

Table 6: Performance on additional LongBench tasks at a 10% recomputation budget. SurroKV (n) denotes an n-token surrogate prefix. The best and second-best results among cache reuse methods are highlighted in **boldface** and underline, respectively.

Methods	MFQA-en	MFQA-zh	Qasper	NQA	DuReader	SAMSum	PassCnt	LCC	RepoBench	MultiNews	VCSUM	GovReport	Avg.
Base model: Llama3.2-3B-Inst.													
FullRecomp.	50.5	54.6	40.8	25.6	36.1	5.7	3.0	26.6	26.8	25.9	15.5	33.3	28.7
NaiveReuse	23.1	25.3	25.8	2.2	5.4	**13.6**	0.4	**26.6**	17.5	23.7	6.7	14.9	15.4
EPIC	44.6	44.1	31.5	15.9	26.0	7.7	2.2	25.7	16.4	**24.6**	15.9	**30.8**	23.8
CacheBlend	44.5	44.4	35.5	18.2	26.3	7.5	**4.2**	26.0	18.1	**24.6**	15.0	30.1	24.5
KVShare	45.3	44.5	34.9	18.0	27.4	7.5	3.1	26.0	20.2	**24.6**	15.3	30.0	24.7
ProphetKV	44.3	46.0	34.5	20.4	27.9	6.7	3.5	26.0	21.6	24.3	15.6	**30.8**	25.1
SurroKV (512)	47.4	49.4	34.7	**21.5**	31.7	7.0	2.0	25.4	21.3	24.1	**16.8**	29.2	25.9
SurroKV (1024)	**48.1**	**50.5**	**36.5**	**21.5**	**32.7**	7.0	4.0	25.4	**21.8**	24.1	16.6	29.1	**26.4**
Base model: Llama3.1-8B-Inst.													
FullRecomp.	57.1	58.0	46.1	31.5	35.9	8.6	9.1	18.4	18.4	26.9	17.2	34.5	30.1
NaiveReuse	36.0	43.2	35.6	13.3	16.9	**31.9**	2.1	17.2	**19.4**	**26.5**	8.4	24.7	22.9
EPIC	**51.8**	54.6	43.8	20.0	26.3	15.5	2.5	15.6	17.2	26.4	15.7	32.0	26.8
CacheBlend	50.2	53.1	41.6	21.4	26.1	8.8	**5.0**	18.1	16.3	26.4	16.5	32.1	26.3
KVShare	50.4	52.5	43.3	22.3	25.8	9.2	3.5	17.6	17.3	26.4	16.4	31.9	26.4
ProphetKV	**51.8**	54.2	44.1	25.2	26.5	7.7	3.5	**18.9**	18.5	26.0	16.6	**32.5**	**27.1**
SurroKV (512)	51.0	54.0	43.8	24.0	**29.1**	7.4	2.5	15.6	15.8	26.3	16.9	31.3	26.5
SurroKV (1024)	50.2	**54.8**	**44.3**	**25.4**	28.5	7.2	3.5	15.3	14.3	26.3	**17.6**	31.3	26.6

A.11 Offline Construction Cost

Unlike independent construction, SurroKV encodes each chunk with a surrogate prefix. Table 13 compares the per-chunk encoding time and peak GPU memory on a single L40S. A 1024-token surrogate increases construction time by 12%–20%.

The surrogate KV cache is computed once and reused across chunks (90 ms for 1024 tokens on Llama-3.1-8B-Instruct). Since only chunk KV states are stored, the final cache size remains unchanged. Longer prefixes increase offline cost (20%–34% at 2048 tokens), with limited additional accuracy gains beyond 1024 tokens (Figure 6).

Table 7: Performance on the RAG tasks of HELMET at 16K and 32K with a 10% recomputation budget. SurroKV (n) denotes a surrogate prefix of n tokens. Avg. is the unweighted mean of the four tasks. The best and second-best results among cache reuse methods within each model are highlighted in **boldface** and underline, respectively.

	16K					32K				
Methods	**NQ**	**TriviaQA**	**HotpotQA**	**PopQA**	**Avg.**	**NQ**	**TriviaQA**	**HotpotQA**	**PopQA**	**Avg.**
Base model: Llama3.2-3B-Inst.										
FullRecomp.	53.0	85.5	59.3	58.7	64.1	51.0	87.8	54.7	53.8	61.8
NaiveReuse	9.3	35.0	10.3	24.8	19.9	3.7	14.0	4.0	7.7	7.3
EPIC	44.5	75.0	41.0	25.5	46.5	31.7	64.8	27.0	23.0	36.6
CacheBlend	46.5	77.0	42.3	27.7	48.4	39.0	72.3	34.0	24.2	42.4
KVShare	44.5	77.0	43.0	28.3	48.2	38.5	71.2	33.0	21.2	41.0
ProphetKV	**47.8**	78.0	42.0	35.0	50.7	42.3	71.7	34.3	30.8	44.8
SurroKV (512)	45.8	81.2	41.7	40.8	52.4	40.8	72.7	39.3	36.7	47.4
SurroKV (1024)	47.2	**83.8**	**43.7**	**43.3**	**54.5**	**43.2**	**77.0**	**43.7**	**38.2**	**50.5**
Base model: Llama3.1-8B-Inst.										
FullRecomp.	58.8	90.0	61.3	59.0	67.3	56.3	89.3	57.3	58.3	65.3
NaiveReuse	41.5	70.5	33.0	25.5	42.6	30.7	62.5	25.7	23.5	35.6
EPIC	48.3	82.0	50.7	31.7	53.2	45.5	77.8	42.3	22.5	47.0
CacheBlend	**50.7**	83.7	52.3	35.3	55.5	45.7	81.0	44.0	29.0	49.9
KVShare	48.7	83.0	50.7	36.3	54.7	44.5	81.2	42.3	27.2	48.8
ProphetKV	**50.7**	85.3	48.0	41.7	56.4	47.0	82.3	48.3	34.5	53.0
SurroKV (512)	50.5	**87.2**	54.3	46.5	59.6	48.7	84.7	**51.7**	39.2	56.0
SurroKV (1024)	50.0	87.0	**55.3**	**47.8**	**60.0**	**49.7**	**85.0**	**51.7**	**39.3**	**56.4**

Table 8: Performance on the RAG tasks of HELMET at 16K and 32K with a 20% recomputation budget. SurroKV (n) denotes a surrogate prefix of n tokens. Avg. is the unweighted mean of the four tasks. The best and second-best results among cache reuse methods within each model are highlighted in **boldface** and underline, respectively.

	16K					32K				
Methods	**NQ**	**TriviaQA**	**HotpotQA**	**PopQA**	**Avg.**	**NQ**	**TriviaQA**	**HotpotQA**	**PopQA**	**Avg.**
Base model: Llama3.2-3B-Inst.										
FullRecomp.	53.0	85.5	59.3	58.7	64.1	51.0	87.8	54.7	53.8	61.8
NaiveReuse	9.3	35.0	10.3	24.8	19.9	3.7	14.0	4.0	7.7	7.3
EPIC	44.5	80.2	44.0	30.5	49.8	37.5	72.8	36.3	27.3	43.5
CacheBlend	**47.8**	80.0	45.0	32.3	51.3	40.0	75.0	35.3	23.2	43.4
KVShare	47.3	79.0	44.3	31.7	50.6	38.8	72.3	34.7	21.0	41.7
ProphetKV	46.5	80.7	44.0	45.0	54.0	41.8	73.3	40.0	38.5	48.4
SurroKV (512)	47.2	**84.2**	46.7	50.2	57.0	42.3	80.7	**45.0**	42.7	52.7
SurroKV (1024)	47.7	83.8	**48.7**	**51.0**	**57.8**	**43.5**	**82.5**	44.3	**43.7**	**53.5**
Base model: Llama3.1-8B-Inst.										
FullRecomp.	58.8	90.0	61.3	59.0	67.3	56.3	89.3	57.3	58.3	65.3
NaiveReuse	41.5	70.5	33.0	25.5	42.6	30.7	62.5	25.7	23.5	35.6
EPIC	48.7	84.2	53.0	37.3	55.8	46.8	80.3	48.0	27.0	50.5
CacheBlend	51.0	84.2	53.0	37.7	56.5	46.2	84.2	48.7	32.2	52.8
KVShare	48.8	84.2	52.0	38.2	55.8	43.0	83.2	46.0	29.8	50.5
ProphetKV	51.5	85.7	55.0	49.5	60.4	51.0	87.0	51.0	42.2	57.8
SurroKV (512)	**52.0**	87.7	57.3	53.0	62.5	**53.2**	88.8	54.0	47.8	61.0
SurroKV (1024)	**52.0**	**88.3**	**57.7**	**53.8**	**63.0**	52.7	**89.3**	**54.3**	**49.3**	**61.4**

B Additional Theory for Surrogate-Context Cache Construction

For notational simplicity, any fixed mandatory model tokens are absorbed into the baseline sequence x. Thus, $\emptyset$ denotes the absence of any additional construction context. Unless stated otherwise, we consider standard decoder-only attention with RoPE and no additional position-dependent attention bias.

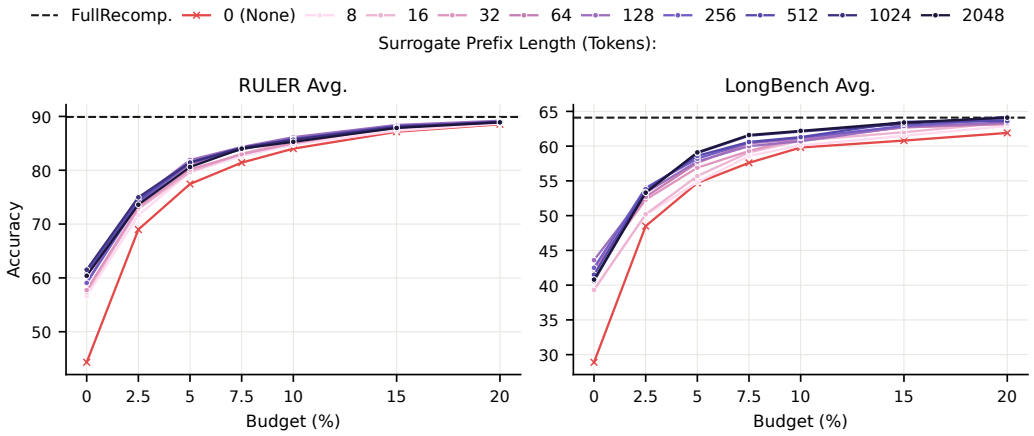

Figure 6: Performance across surrogate prefix lengths and recomputation budgets on Llama-3.1-8B-Instruct at 8K. Results are averaged over RULER (left) and LongBench (right). The red curve denotes independent chunk construction, and the dashed line denotes full recomputation.

Table 9: Detailed results across surrogate prefix lengths and recomputation budgets on Llama-3.1-8B-Instruct at 8K. Results correspond to Figure 6. A prefix length of zero denotes independent chunk construction.

Prefix	RULER							LongBench						
Length	0%	2.5%	5%	7.5%	10%	15%	20%	0%	2.5%	5%	7.5%	10%	15%	20%
0	44.3	69.0	77.4	81.4	84.0	87.1	88.5	28.9	48.6	54.7	57.6	59.8	60.7	61.9
8	56.7	71.6	79.6	82.6	84.8	87.3	88.7	39.5	50.1	54.8	58.6	60.1	61.5	62.9
16	57.4	72.8	79.7	83.1	85.5	87.9	88.9	39.3	50.2	55.7	59.1	60.8	62.0	63.4
32	57.7	73.1	80.1	83.0	85.0	88.2	88.9	42.1	52.3	56.9	59.3	61.2	62.7	63.2
64	59.2	74.4	81.0	84.0	85.8	88.3	89.0	42.4	52.7	57.6	60.4	61.0	62.7	63.2
128	59.0	74.5	81.9	84.3	86.1	88.4	89.2	43.6	53.3	57.8	60.0	60.7	62.8	63.4
256	59.1	73.9	81.3	83.9	85.8	87.9	89.1	42.5	54.0	58.2	60.5	61.2	62.9	63.9
512	60.9	74.4	81.5	84.2	85.8	88.2	88.9	41.5	53.8	58.6	60.6	61.3	63.5	63.6
1024	61.5	75.0	81.4	84.2	85.7	88.0	89.0	40.6	53.4	59.0	61.5	62.1	63.2	64.2
2048	60.4	73.6	80.6	84.0	85.3	87.9	88.9	40.8	53.3	59.1	61.6	62.2	63.4	64.1

B.1 Derivation of the First-Layer Construction Effect

First-layer construction effect (Equation (8)). For token i, the change induced by construction context c is

$$g_i(c) = o_i(c) - o_i(\emptyset) = \rho_i(c)\big(\mu_i(c) - u_i\big).$$

Proof. Consider token i in reusable chunk x. Let $\mathcal{X}_i$ denote the chunk tokens causally visible to i, and let $\mathcal{C}$ denote the preceding construction-context tokens.

Because c is prepended before the entire chunk, relative positions among chunk tokens remain unchanged. At the first attention layer, their pairwise RoPE attention logits therefore remain unchanged.

Let $Z_{x,i} = \sum_{j\in\mathcal{X}_i} \exp(\ell_{ij})$ and $Z_{c,i} = \sum_{j\in\mathcal{C}} \exp(\ell_{ij}(c))$. The total attention mass assigned to the construction context is $\rho_i(c) = Z_{c,i}/(Z_{x,i} + Z_{c,i})$. Consequently, for any $j \in \mathcal{X}_i$,

$$a_{ij}(c) = (1 - \rho_i(c))a_{ij}^{\text{ind}}.$$

Let $u_i = \sum_{j\in\mathcal{X}_i} a_{ij}^{\text{ind}} v_j$ denote the independent chunk attention output. When $\rho_i(c) > 0$, define the attention-weighted context representation as

$$\mu_i(c) = \frac{1}{\rho_i(c)} \sum_{j\in\mathcal{C}} a_{ij}(c) v_j.$$

Table 10: Detailed performance across recomputation budgets on three models. Results are reported as average scores on RULER at 8K (left) and LongBench (right), corresponding to Figure 5. Columns denote recomputation budgets from 2.5% to 30%.

	RULER							LongBench						
Methods	**2.5%**	**5%**	**7.5%**	**10%**	**15%**	**20%**	**30%**	**2.5%**	**5%**	**7.5%**	**10%**	**15%**	**20%**	**30%**
Base model: Llama3.2-3B-Inst.														
FullRecomp.	83.1	83.1	83.1	83.1	83.1	83.1	83.1	42.2	42.2	42.2	42.2	42.2	42.2	42.2
NaiveReuse	23.5	23.5	23.5	23.5	23.5	23.5	23.5	13.1	13.1	13.1	13.1	13.1	13.1	13.1
EPIC	52.2	53.6	54.5	56.3	56.9	58.1	62.1	21.4	22.3	23.4	24.6	26.1	26.9	28.9
CacheBlend	55.6	58.3	60.6	63.5	65.8	68.7	72.6	28.7	28.8	28.0	27.7	28.5	29.7	30.4
KVShare	54.6	58.5	60.2	61.7	64.2	66.2	69.0	26.0	29.4	28.2	27.9	27.7	27.9	28.5
ProphetKV	64.0	69.2	72.6	74.7	78.0	79.0	81.0	30.4	33.7	34.5	35.3	35.5	37.2	37.7
SurroKV (512)	72.2	77.6	78.9	79.8	80.7	81.6	82.1	33.8	37.2	37.7	37.9	39.5	40.1	40.5
SurroKV (1024)	74.2	78.7	80.3	80.5	81.4	82.0	82.3	34.1	37.9	37.9	39.0	40.4	40.2	41.0
Base model: Llama3.1-8B-Inst.														
FullRecomp.	89.9	89.9	89.9	89.9	89.9	89.9	89.9	64.1	64.1	64.1	64.1	64.1	64.1	64.1
NaiveReuse	44.3	44.3	44.3	44.3	44.3	44.3	44.3	28.9	28.9	28.9	28.9	28.9	28.9	28.9
EPIC	58.1	59.2	60.5	62.7	64.4	66.5	70.3	40.1	41.6	43.7	43.8	45.1	47.6	50.4
CacheBlend	58.4	61.8	64.0	66.2	70.0	71.8	75.8	43.8	45.5	47.7	48.8	50.4	52.3	53.5
KVShare	61.8	65.2	66.9	68.3	71.1	72.4	75.4	44.9	47.2	48.4	49.3	50.5	50.8	52.2
ProphetKV	69.0	77.4	81.4	84.0	87.1	88.5	89.4	48.6	54.7	57.6	59.8	60.7	61.9	62.8
SurroKV (512)	74.4	81.5	84.2	85.8	88.2	88.9	89.6	53.8	58.6	60.6	61.3	63.5	63.6	64.1
SurroKV (1024)	75.0	81.4	84.2	85.7	88.0	89.0	89.7	53.4	59.0	61.5	62.1	63.2	64.2	63.8
Base model: Qwen2.5-14B-Inst.														
FullRecomp.	89.2	89.2	89.2	89.2	89.2	89.2	89.2	65.4	65.4	65.4	65.4	65.4	65.4	65.4
NaiveReuse	47.6	47.6	47.6	47.6	47.6	47.6	47.6	17.2	17.2	17.2	17.2	17.2	17.2	17.2
EPIC	55.4	56.3	56.7	57.5	60.4	63.4	68.4	35.4	35.7	37.0	37.4	39.3	40.6	41.4
CacheBlend	57.3	58.7	60.8	62.6	65.1	67.2	72.2	36.9	39.0	40.7	42.6	44.7	45.2	47.1
KVShare	57.9	62.4	65.4	66.7	67.9	69.8	73.0	29.5	36.6	38.9	41.1	44.4	45.7	45.8
ProphetKV	75.5	79.7	81.5	83.0	85.0	86.4	88.0	46.9	50.3	53.3	55.0	60.0	63.0	64.0
SurroKV (512)	81.8	83.8	85.1	86.1	87.1	87.5	87.8	51.6	54.0	55.9	58.3	61.7	63.0	64.4
SurroKV (1024)	81.7	84.6	85.6	86.7	87.5	87.9	88.2	51.3	52.9	54.9	57.9	61.0	62.9	64.7

Table 11: Chunk size on Llama-3.1-8B-Instruct at 8K. Upper: FullRecomp., NaiveReuse, EPIC, CacheBlend, and KVShare. Lower: ProphetKV and SurroKV, where None denotes no recomputation.

			EPIC						CacheBlend						KVShare					
Chunk	**Full**	**Naive**	**2.5%**	**5%**	**7.5%**	**10%**	**15%**	**20%**	**2.5%**	**5%**	**7.5%**	**10%**	**15%**	**20%**	**2.5%**	**5%**	**7.5%**	**10%**	**15%**	**20%**
RULER																				
256	90.0	35.7	53.5	52.8	54.3	55.5	57.5	60.5	52.7	54.0	56.5	59.1	63.5	66.6	47.9	56.8	60.6	63.2	66.2	67.9
512	89.9	44.3	58.1	59.2	60.5	62.7	64.4	66.5	58.4	61.8	64.0	66.2	70.0	71.8	61.8	65.2	66.9	68.3	71.1	72.4
1024	90.0	54.9	66.1	67.0	67.0	67.5	68.8	70.7	66.4	68.4	70.3	71.7	74.2	76.0	67.8	70.7	72.1	73.1	75.4	76.4
LongBench																				
256	64.2	23.2	34.3	35.3	36.5	37.8	38.8	40.5	38.2	41.2	42.4	43.4	45.3	46.8	35.3	42.0	44.8	45.1	46.6	46.6
512	64.1	28.9	40.1	41.6	43.7	43.8	45.1	47.6	43.8	45.5	47.7	48.8	50.4	52.3	44.9	47.2	48.4	49.3	50.5	50.8
1024	64.2	37.1	47.1	49.6	50.0	51.1	52.6	53.5	49.8	51.7	53.5	54.3	55.4	56.1	50.8	51.8	52.8	53.3	54.4	55.5

	ProphetKV						SurroKV (512)							SurroKV (1024)						
Chunk	**2.5%**	**5%**	**7.5%**	**10%**	**15%**	**20%**	**None**	**2.5%**	**5%**	**7.5%**	**10%**	**15%**	**20%**	**None**	**2.5%**	**5%**	**7.5%**	**10%**	**15%**	**20%**
RULER																				
256	60.3	68.3	73.8	77.3	82.3	85.2	60.1	69.4	75.9	80.4	82.1	84.6	86.6	59.5	71.0	76.7	80.6	83.0	85.0	86.7
512	69.0	77.4	81.4	84.0	87.1	88.5	60.9	74.4	81.5	84.2	85.8	88.2	88.9	61.5	75.0	81.4	84.2	85.7	88.0	89.0
1024	77.6	83.5	86.0	87.3	88.9	89.3	65.5	79.5	85.2	87.6	88.3	88.9	89.3	66.2	79.5	85.0	87.1	88.3	89.1	89.2
LongBench																				
256	44.5	50.8	55.5	57.2	58.9	60.8	34.8	48.1	55.4	59.1	61.4	61.5	63.3	35.4	48.8	57.2	60.3	61.6	62.7	63.3
512	48.6	54.7	57.6	59.8	60.7	61.9	41.5	53.8	58.6	60.6	61.3	63.5	63.6	40.6	53.4	59.0	61.5	62.1	63.2	64.2
1024	54.7	57.6	60.5	61.1	62.9	63.1	47.5	57.5	60.4	61.9	62.4	63.8	64.0	47.3	57.4	60.8	61.8	62.8	63.8	63.9

The attention output is therefore $o_i(c) = (1 - \rho_i(c))u_i + \rho_i(c)\mu_i(c)$. Since $o_i(\emptyset) = u_i$, we obtain

$$g_i(c) = o_i(c) - o_i(\emptyset) = \rho_i(c)\big(\mu_i(c) - u_i\big),$$

which establishes Equation (8). For $c = \emptyset$, we have $\rho_i(c) = 0$ and $g_i(\emptyset) = 0$, without requiring $\mu_i(\emptyset)$ to be defined. □

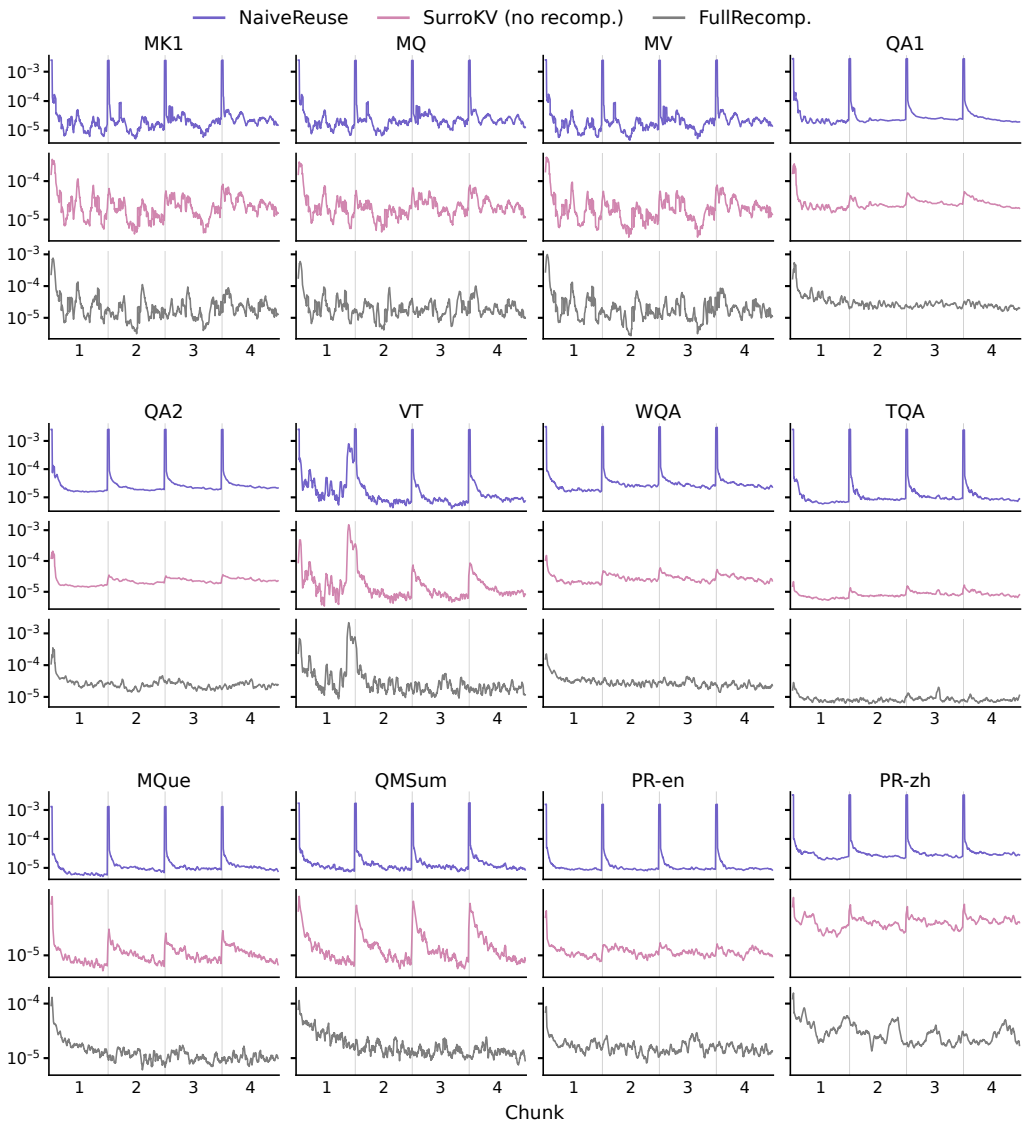

Figure 7: Query-to-context attention profiles across RULER and LongBench tasks (Llama-3.1-8B-Instruct, 8K). Each group represents a task, with separate rows for different cache construction methods. Attention is averaged over examples and smoothed using a 16-token moving average. Tasks are ordered as in Table 1. HotpotQA is shown in Figure 3.

B.2 Proof of Theorem 5.1

Theorem 5.1 (Future-context effect decomposition). Let p denote the realized preceding context and $\bar{g}_i = \mathbb{E}_p[g_i(p)]$. For any fixed surrogate s,

$$\mathbb{E}_p\|g_i(s) - g_i(p)\|_2^2 = \mathbb{E}_p\|g_i(p) - \bar{g}_i\|_2^2 + \|g_i(s) - \bar{g}_i\|_2^2.$$

Proof. Since $g_i(s) - g_i(p) = (g_i(s) - \bar{g}_i) + (\bar{g}_i - g_i(p))$ and $\mathbb{E}_p[\bar{g}_i - g_i(p)] = 0$, expanding the squared norm gives

$$\mathbb{E}_p\|g_i(s) - g_i(p)\|_2^2 = \mathbb{E}_p\|g_i(p) - \bar{g}_i\|_2^2 + \|g_i(s) - \bar{g}_i\|_2^2.$$

□

Table 12: Detailed results across token selectors and recomputation budgets on Llama-3.1-8B-Instruct at 8K. We compare independent chunk construction with surrogate-based construction using a 1024-token prefix. At a 0% recomputation budget, no online token selection or recomputation is performed.

	EPIC		CacheBlend		KVShare		ProphetKV	
Budget	**Chunk**	**SurroKV**	**Chunk**	**SurroKV**	**Chunk**	**SurroKV**	**Chunk**	**SurroKV**
RULER								
0%	44.3	61.5	44.3	61.5	44.3	61.5	44.3	61.5
2.5%	58.1	63.1	58.4	66.8	61.8	67.7	69.0	75.0
5%	59.2	64.1	61.8	68.5	65.2	70.1	77.4	81.4
7.5%	60.5	65.1	64.0	69.5	66.9	71.1	81.4	84.2
10%	62.7	66.4	66.2	70.1	68.3	72.0	84.0	85.7
15%	64.4	67.9	70.0	71.0	71.1	73.4	87.1	88.0
20%	66.5	68.9	71.8	72.2	72.4	75.2	88.5	89.0
FullRecomp.	89.9	89.9	89.9	89.9	89.9	89.9	89.9	89.9
LongBench								
0%	28.9	40.6	28.9	40.6	28.9	40.6	28.9	40.6
2.5%	40.1	41.9	43.8	44.7	44.9	47.0	48.6	53.4
5%	41.6	42.7	45.5	46.2	47.2	50.7	54.7	59.0
7.5%	43.7	43.4	47.7	48.3	48.4	51.1	57.6	61.5
10%	43.8	44.5	48.8	48.1	49.3	51.5	59.8	62.1
15%	45.1	45.6	50.4	49.9	50.5	52.3	60.7	63.2
20%	47.6	46.3	52.3	51.2	50.8	53.1	61.9	64.2
FullRecomp.	64.1	64.1	64.1	64.1	64.1	64.1	64.1	64.1

Table 13: Offline construction time and peak GPU memory for 512-token chunks on a single L40S. Upper: encoding time per chunk. Lower: peak GPU memory. Parentheses indicate the ratio to independent construction. The one-time surrogate prefill is excluded from the encoding time.

Model	**Encoding time per chunk (ms)**			
	None	**512**	**1024**	**2048**
Llama3.2-3B-Inst.	23.1	26.0 (1.13×)	27.7 (1.20×)	31.0 (1.34×)
Llama3.1-8B-Inst.	48.9	54.0 (1.10×)	55.8 (1.14×)	60.0 (1.23×)
Qwen2.5-14B-Inst.	97.2	104.4 (1.07×)	108.7 (1.12×)	117.0 (1.20×)

Model	**Peak GPU memory (GB)**			
	None	**512**	**1024**	**2048**
Llama3.2-3B-Inst.	6.53	6.63 (1.01×)	6.66 (1.02×)	6.79 (1.04×)
Llama3.1-8B-Inst.	16.20	16.26 (1.00×)	16.40 (1.01×)	16.48 (1.02×)
Qwen2.5-14B-Inst.	29.87	30.00 (1.00×)	30.11 (1.01×)	30.30 (1.01×)

B.3 Additional Properties of Surrogate Conditioning

Comparison with independent construction. Since $g_i(\emptyset) = 0$, a surrogate has smaller expected construction mismatch than independent construction exactly when $\|g_i(s) - \bar{g}_i\|_2^2 < \|\bar{g}_i\|_2^2$, or equivalently,

$$2\langle g_i(s), \bar{g}_i \rangle > \|g_i(s)\|_2^2.$$

Writing $g_i(s) = \rho_i(s) w_i(s)$ with $w_i(s) = \mu_i(s) - u_i$ shows that a necessary condition for beneficial conditioning is $\langle w_i(s), \bar{g}_i \rangle > 0$.

Preferred construction strength. For a fixed nonzero direction w, minimizing $\|\rho w - \bar{g}_i\|_2^2$ over $\rho \in [0, 1]$ gives

$$\rho_i^\star = \mathrm{clip}_{[0,1]} \left(\frac{\langle w, \bar{g}_i \rangle}{\|w\|_2^2} \right).$$

Thus, even an aligned direction generally has a finite preferred strength.

Effect of surrogate length. Consider controlled nested prefixes $s_1 \subset s_2 \subset \cdots$, where each longer prefix is formed by adding tokens strictly before the existing prefix and reusable chunk. Previously existing first-layer RoPE logits remain unchanged, while each new token contributes positive unnormalized attention mass. Therefore, $\rho_i(s_{\ell+1}) > \rho_i(s_\ell)$.

Hence, surrogate length monotonically increases first-layer context exposure under this controlled construction. If the induced direction $w_i(s_\ell)$ remains approximately stable, increasing length primarily increases construction strength, yielding the qualitative regime

$$\text{under-correction} \;\rightarrow\; \text{appropriate correction} \;\rightarrow\; \text{over-correction}.$$

This conclusion does not imply monotonic behavior for arbitrary prefixes, whose induced directions may also change.

B.4 Layerwise Propagation to Reusable KV States

The preceding analysis describes the direct first-layer effect. During cache construction, however, the preceding context remains visible throughout the Transformer and can introduce additional changes at later layers.

Let $H_x^\ell(c)$ denote the reusable chunk hidden states at layer ℓ under context c, let $\mathcal{T}_c^\ell$ denote the corresponding layer transformation, and define $\Delta H^\ell(c) = H_x^\ell(c) - H_x^\ell(\emptyset)$. Then

$$\Delta H^{\ell+1}(c) = \underbrace{\mathcal{T}_c^\ell(H_x^\ell(c)) - \mathcal{T}_c^\ell(H_x^\ell(\emptyset))}_{\text{propagated representation effect}} + \underbrace{\mathcal{T}_c^\ell(H_x^\ell(\emptyset)) - \mathcal{T}_\emptyset^\ell(H_x^\ell(\emptyset))}_{\text{fresh context effect}}.$$

Thus, later-layer changes arise from both propagation of earlier representation differences and fresh context-dependent interactions.

For token i, under the standard linear key projection, the corresponding key change in canonical chunk-local coordinates is

$$K_i^\ell(c) - K_i^\ell(\emptyset) = R_i W_K^\ell \big[\mathrm{Norm}(H_i^\ell(c)) - \mathrm{Norm}(H_i^\ell(\emptyset))\big],$$

where R_i is the RoPE rotation at the canonical local position. An analogous expression holds for values, without RoPE rotation.

Shared effects in key space. To compare caches obtained under different future placements, we express their keys in the same canonical chunk-local coordinates before request-specific positional placement.

Let $G^\ell(p) = K^\ell(p) - K^\ell(\emptyset)$ and $\bar{G}^\ell = \mathbb{E}_p[G^\ell(p)]$. Applying the same decomposition as Theorem 5.1 gives

$$\mathbb{E}_p\|G^\ell(s) - G^\ell(p)\|_F^2 = \mathbb{E}_p\|G^\ell(p) - \bar{G}^\ell\|_F^2 + \|G^\ell(s) - \bar{G}^\ell\|_F^2.$$

Hence, the shared-effect interpretation extends from the first-layer representation to the reusable key cache.

B.5 Proof of Theorem 5.2

Theorem 5.2 (Cache-to-routing bound). With the query fixed to its full-prefill value, identical attention masks and realized positions, and all non-reusable keys matching full prefill,

$$\mathcal{L}(z^{\mathrm{sur}}) \le \frac{\|q\|_2^2}{4d}\|K^{\mathrm{sur}} - K^{\mathrm{full}}\|_F^2.$$

Proof. Let $F(z) = \log \sum_j e^{z_j}$. Its gradient is $\nabla F(z) = \mathrm{softmax}(z)$ and its Hessian is $H(z) = \mathrm{diag}(a) - aa^\top$, where $a = \mathrm{softmax}(z)$. The Hessian has spectral norm at most $1/2$, so F is $1/2$-smooth.

Moreover, $D_{\mathrm{KL}}(a^{\mathrm{full}}\|a^{\mathrm{sur}}) = D_F(z^{\mathrm{sur}}, z^{\mathrm{full}})$, where D_F is the Bregman divergence induced by F. Smoothness therefore implies

$$\mathcal{L}(z^{\mathrm{sur}}) \le \frac{1}{4}\|z^{\mathrm{sur}} - z^{\mathrm{full}}\|_2^2.$$

Because the caches are placed at the same realized online positions, $z^{\mathrm{sur}} - z^{\mathrm{full}} = (K^{\mathrm{sur}} - K^{\mathrm{full}})q/\sqrt{d}$. Using the matrix–vector norm bound,

$$\mathcal{L}(z^{\mathrm{sur}}) \le \frac{1}{4}\|z^{\mathrm{sur}} - z^{\mathrm{full}}\|_2^2 \le \frac{\|q\|_2^2}{4d}\|K^{\mathrm{sur}} - K^{\mathrm{full}}\|_F^2.$$

For RoPE, corresponding surrogate and full-prefill keys at the same realized position undergo the same orthogonal rotation. Their row-wise Euclidean distances, and therefore the Frobenius distance between the two key matrices, are preserved. □

B.6 Proof of Theorem 5.3

Theorem 5.3 (Exact routing effect). Let $\Delta z^{\mathrm{sur}} = z^{\mathrm{sur}} - z^{\mathrm{ind}}$, and define $A^{\mathrm{sur}} = \langle a^{\mathrm{full}} - a^{\mathrm{ind}}, \Delta z^{\mathrm{sur}}\rangle$ and $C^{\mathrm{sur}} = D_{\mathrm{KL}}(a^{\mathrm{ind}}\|a^{\mathrm{sur}})$. Then

$$\mathcal{L}(z^{\mathrm{sur}}) - \mathcal{L}(z^{\mathrm{ind}}) = -A^{\mathrm{sur}} + C^{\mathrm{sur}}.$$

Hence, surrogate construction improves routing if and only if $A^{\mathrm{sur}} > C^{\mathrm{sur}}$.

Proof. Using the log-partition function F, the routing loss can be written as $\mathcal{L}(z) = D_F(z, z^{\mathrm{full}})$. Therefore,

$$\mathcal{L}(z^{\mathrm{sur}}) - \mathcal{L}(z^{\mathrm{ind}}) = F(z^{\mathrm{sur}}) - F(z^{\mathrm{ind}}) - (a^{\mathrm{full}})^\top \Delta z^{\mathrm{sur}}.$$

By the definition of Bregman divergence, this becomes

$$-\left\langle a^{\mathrm{full}} - a^{\mathrm{ind}}, \Delta z^{\mathrm{sur}}\right\rangle + D_F(z^{\mathrm{sur}}, z^{\mathrm{ind}}).$$

Since $D_F(z^{\mathrm{sur}}, z^{\mathrm{ind}}) = D_{\mathrm{KL}}(a^{\mathrm{ind}}\|a^{\mathrm{sur}})$, we obtain Equation (10). The routing loss therefore decreases if and only if $A^{\mathrm{sur}} > C^{\mathrm{sur}}$. □

B.7 Proof of Proposition 5.1

Proposition 5.1 (Interpreting the alignment benefit). Let $\Delta z^{\mathrm{full}} = z^{\mathrm{full}} - z^{\mathrm{ind}}$ and

$$\overline{H}^{\mathrm{full}} = \int_0^1 H(z^{\mathrm{ind}} + t\Delta z^{\mathrm{full}})\, dt,$$

where $H(z) = \mathrm{diag}(a) - aa^\top$ for $a = \mathrm{softmax}(z)$. Then

$$A^{\mathrm{sur}} = (\Delta z^{\mathrm{sur}})^\top \overline{H}^{\mathrm{full}} \Delta z^{\mathrm{full}}.$$

Proof. By the fundamental theorem of calculus,

$$a^{\mathrm{full}} - a^{\mathrm{ind}} = \left[\int_0^1 H(z^{\mathrm{ind}} + t\Delta z^{\mathrm{full}})\, dt\right] \Delta z^{\mathrm{full}}.$$

Substituting this expression into the definition of A^{sur} gives

$$A^{\mathrm{sur}} = (\Delta z^{\mathrm{sur}})^\top \overline{H}^{\mathrm{full}} \Delta z^{\mathrm{full}},$$

which proves Proposition 5.1. □

Since $\overline{H}^{\mathrm{full}} \succeq 0$, this expression measures alignment between the surrogate-induced and full-prefill logit changes in the geometry induced by softmax attention. The alignment benefit need not be positive for an arbitrary surrogate.

B.8 Proof of Corollary 5.3.1

Corollary 5.3.1 (Partial full-prefill correction is sufficient). If $\Delta z^{\mathrm{sur}} = \alpha \Delta z^{\mathrm{full}}$ with $0 < \alpha \le 1$, then

$$\mathcal{L}(z^{\mathrm{sur}}) \le (1-\alpha)\mathcal{L}(z^{\mathrm{ind}}).$$

Thus, any nonzero partial correction along the full-prefill direction reduces routing discrepancy whenever the initial discrepancy is nonzero.

Proof. Under the stated condition, $z^{\mathrm{sur}} = (1-\alpha)z^{\mathrm{ind}} + \alpha z^{\mathrm{full}}$. Since $\mathcal{L}$ is convex and $\mathcal{L}(z^{\mathrm{full}}) = 0$,

$$\mathcal{L}(z^{\mathrm{sur}}) \le (1-\alpha)\mathcal{L}(z^{\mathrm{ind}}) + \alpha\mathcal{L}(z^{\mathrm{full}}) = (1-\alpha)\mathcal{L}(z^{\mathrm{ind}}).$$

□

B.9 Extension to Multi-Chunk RAG

Suppose a request contains retrieved chunks $x_1, \ldots, x_M$. For chunk x_m, let p_m denote its realized preceding context, which may include instructions and previously retrieved chunks. The construction-side analysis applies chunk-wise under the corresponding distribution of p_m. These distributions may differ with retrieval rank, preceding-context length, or chunk order.

At reuse time, the cached chunks are assembled into a global key matrix, e.g.,

$$K^{\mathrm{sur}} = [K_1^{\mathrm{sur}}; \ldots; K_M^{\mathrm{sur}}],$$

with analogous definitions for independent and full-prefill caches. The routing analysis then applies to the entire assembled context.

Partitioning the attention vectors and surrogate-induced logit changes by chunk, the alignment benefit decomposes as

$$A^{\mathrm{sur}} = \sum_{m=1}^{M} \left\langle a_m^{\mathrm{full}} - a_m^{\mathrm{ind}}, \Delta z_m^{\mathrm{sur}} \right\rangle.$$

In contrast, $D_{\mathrm{KL}}(a^{\mathrm{ind}} \| a^{\mathrm{sur}})$ remains a global quantity because all chunks share the same softmax normalization. Thus, construction can be analyzed chunk-wise while its routing effect remains globally coupled.

When keys outside the reusable region match full prefill, the chunk blocks occupy disjoint rows and

$$\|K^{\mathrm{sur}} - K^{\mathrm{full}}\|_F^2 = \sum_{m=1}^{M} \|K_m^{\mathrm{sur}} - K_m^{\mathrm{full}}\|_F^2.$$

The cache-to-routing bound therefore becomes

$$\mathcal{L}(z^{\mathrm{sur}}) \leq \frac{\|q\|_2^2}{4d} \sum_{m=1}^{M} \|K_m^{\mathrm{sur}} - K_m^{\mathrm{full}}\|_F^2.$$