WHEN IS PREFIX KV REUSE COMPATIBLE WITH SPARSE PREFILL?

Anonymous authors
Paper under double-blind review

ABSTRACT

In sparse prefill with input-dependent attention selection, a prefix cache can depend on tokens outside that prefix. A selector may use later queries to choose the prefix's attention connections, even though the retained attention connections remain causal. We investigate when this dependence affects prefix key/value (KV) reuse in a reference implementation. Across 80 constructed document tasks, switching cache donors changed generated outputs in 11 tasks under sparse attention and 2 under dense attention. In a separate intervention, we tested 4 input pairs with identical prefixes and matched execution shapes. Dynamic routing produced different prefix KV states within each pair. These differences disappeared when the inputs shared attention connections while independently recomputing their own Q/K/V. We derive sufficient conditions for cache reuse to preserve the mathematical result. With matched deterministic execution and cache representation, these conditions also ensure that reuse and recomputation produce identical floating-point representations. On the task set, restricting reuse to complete prefill-call boundaries yielded 72 nonempty cache hits with unchanged reference outputs. Our results provide a diagnostic protocol and a conditional rule for recovering reference outputs during prefix-cache reuse.

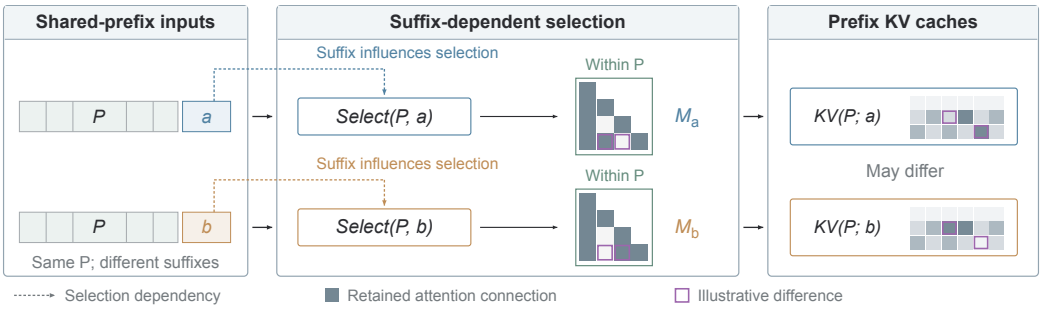


Figure 1: In adaptive sparse prefill, identical prefixes can produce different KV caches. Inputs share prefix P but have different suffixes a and b. Dashed arrows show how suffix queries can influence connection selection. The displayed blocks of M_a and M_b show connections within P: rows are query positions, columns are key positions, and filled cells mark retained connections. All retained connections remain causal, but the resulting prefix caches, $\mathrm{KV}(P; a)$ and $\mathrm{KV}(P; b)$, may differ. Outlined cells mark illustrative differences. Connection patterns and KV values are schematic, not measurements.

1 INTRODUCTION

Prefix caching reduces repeated prefill in serving systems such as vLLM and SGLang (Kwon et al., 2023; Zheng et al., 2024). Its mathematical basis is causal invariance: for fixed model weights, positions and configuration, dense causal attention produces the same prefix states in exact arithmetic. Adaptive sparse prefill raises a further question: under what conditions can a cached prefix still replace the states of a new request?

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.

The difficulty arises when a selector uses suffix queries to choose attention connections for earlier tokens. Although every retained edge is causal, the selection process can make prefix KV depend on the suffix (Figure 1). In one constructed code task, changing the cache donor changed an answer from `None` to `False`, a difference that aggregate accuracy can obscure. Explaining such changes requires separating the states supplied by the donor from the effects of restarting: continuation can regroup calls and change the tensor shapes governing floating-point arithmetic.

Shared-cache evaluations and question-independent cache construction make access to the current question an explicit design choice (Li et al., 2025; Kim et al., 2025). Dependency-controlled modularity and studies of numerical consistency further show why prompt organization and execution matter for reuse (Ordonez & Parnell, 2026; He & Thinking Machines Lab, 2025; vLLM contributors, 2026). These considerations motivate our controls for cache source and restarting, followed by a separate intervention on attention selection. We make three contributions:

- **A diagnostic protocol for prefix-cache reuse.** Donor-hot, self-hot and cold comparisons distinguish source and restart effects in a reference sparse-prefill adaptation, where changing the donor can change the same request's output.
- **Suffix dependence through attention selection.** On matched-shape inputs, sharing selected connections removes the observed prefix KV differences while each input independently recomputes Q/K/V.
- **Conditions for compatible cache reuse.** Sufficient conditions distinguish mathematical from bitwise agreement; complete-call reuse recovers each policy's reference outputs on the task set, including nonempty cache hits.

A pre-registered SCBench comparison examines the separate policy question of whether query-aware recomputation improves quality over context-only sharing.

2 Related work

Shared-prefix execution. ChunkAttention and Hydragen organize shared contexts through prefix trees and batched attention (Ye et al., 2024; Juravsky et al., 2024), while CacheGen compresses states for transfer (Liu et al., 2024b). Reusing these states requires compatibility with the receiving request's computation, including its sparse selection rules. Table 16 locates this requirement among the objectives of existing reuse methods.

Cache retention and question dependence. Retention criteria range from SnapKV's trailing observation window (Li et al., 2024) and KVzip's question-independent context reconstruction (Kim et al., 2025) to recency, attention history and layer budgets in StreamingLLM, H_2O and PyramidKV (Xiao et al., 2024; Zhang et al., 2023; Cai et al., 2025). SCBench evaluates query access in shared-cache quality, reporting an advantage for MInference over SnapKV (Li et al., 2025). Compatibility depends on both the retained subset and the dependencies that created its values.

Modular reuse and dependency control. Reuse across prompt modules requires explicit treatment of positions and cross-module interactions. Prompt Cache uses schemas and fixed positions (Gim et al., 2024), while MiniPIC controls dependencies and decouples stored keys from request-specific positions (Ordonez & Parnell, 2026). CacheBlend and EPIC address cross-document interactions and boundary attention sinks through recomputation (Yao et al., 2025; Hu et al., 2025). Block-Attention independently encodes documents before a fine-tuned query block (Ma et al., 2025). Adaptive connection selection adds a further dependency to this design problem, which we examine in Section 4.2.

Where sparsity enters the computation. Sparse prefill changes how token states are constructed: MInference, FlexPrefill and XAttention estimate attention connections (Jiang et al., 2024; Lai et al., 2025; Xu et al., 2025), MoBA selects blocks (Lu et al., 2025), and NSA combines compression with fine-grained selection (Yuan et al., 2025). Quest instead selects existing cache pages for the current query (Tang et al., 2024). This distinction matters when tracing prefix dependencies. QUOKA and CompactAttention also bring call grouping and paged execution into sparse-prefill de-

sign (Jones et al., 2026; Song et al., 2026), motivating our joint treatment of connection selection and call boundaries (Section 3.2).

Task quality and numerical execution. LongBench, RULER and Lost in the Middle measure performance across tasks, context lengths and evidence positions (Bai et al., 2024; Hsieh et al., 2024; Liu et al., 2024a). Reference preservation additionally depends on execution: FlashAttention changes memory access for exact attention (Dao et al., 2022), while Orca and Sarathi-Serve vary request or prefill grouping (Yu et al., 2022; Agrawal et al., 2024). Reports of shape-dependent arithmetic and batch-invariant kernels (He & Thinking Machines Lab, 2025; vLLM contributors, 2026) motivate dense controls that distinguish numerical effects from sparse selection (Section 5).

3 Diagnostic protocol and experimental scope

We test cache-source effects, selection dependence, reference recovery and encoding-policy choice. Table 1 separates their evidence; Appendix A identifies the implementations and archived records.

Question	Comparison and sample	Supported inference
Can the cache source change outputs?	Donor-cache, own-cache and cold-run controls; 80 tasks	Output changes with matched continuation; donor content and computation shape can both vary
Can attention selection change prefix KV states?	Shared attention connections; 4 input pairs with matched computation shapes, two execution plans	Tests selection effects on prefix KV states; no answer generation
When can reuse recover the reference computation?	Reuse at complete prefill-call boundaries; the same 80 tasks	Matches each policy's cold-run logits and generated outputs; 72 tasks reuse a nonempty prefix cache
How do reuse policies compare?	Context-only sharing vs. recomputation using the current question; 12 documents, 47 questions	Compares task quality and token work; 35 of the questions also support four-policy comparisons

Table 1: Experimental comparisons and the questions they address. The cache-source and reference-recovery tests use the same 80 tasks. Appendix A maps the descriptive experiment names to the archived record identifiers.

3.1 Reference computation and substitution controls

Let x be the consumer request, d the cache donor, and J their longest common token-prefix length. Configuration θ fixes model weights, token positions and positional parameters, attention policy, routing groups, selection and tie-breaking rules, budgets and terminal-block handling. Execution plan σ specifies the forward-call boundaries; together (θ, σ) define the canonical computation. A cold run computes x from the beginning, producing layer states $\mathcal{K}_\ell(x; \theta, \sigma)$. A hot run reuses donor KV at $[0, L)$, $L \leq J$, and continues x under plan τ. We write $\mathcal{K}_{<L}$ for prefix keys and values across all layers. The donor plan σ_d may differ from the consumer reference σ.

Token matching identifies a candidate cache. Representation compatibility requires matching the consumer's reference prefix KV; continuation compatibility requires preserving its outputs after substitution. We distinguish mathematical from bitwise equality and measure logits, generated tokens and scores separately: equal scores can hide different answers, and equal answers can hide different logits.

The reference defines the comparison. The main cold reference uses calls of at most $B = 2048$ tokens. Call-local selection is shared within each call, so restarting inside a call changes its grouping. Fixed-grid routing instead uses absolute token boundaries that persist when a call combines complete units. Each policy has its own cold reference; dense attention and published whole-sequence sparse prefill define separate computations.

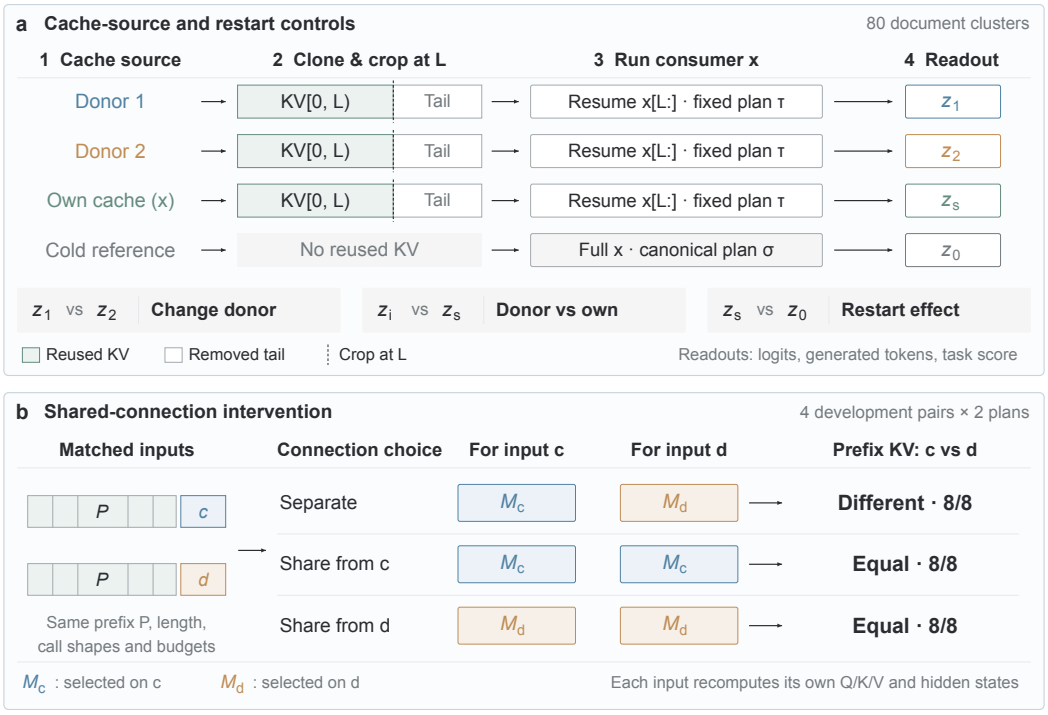


Figure 2: Cache-source, restart and shared-connection interventions. (a) Complete caches are cloned and cropped at L. All hot arms use the same consumer continuation $x[L:]$ and plan τ; the cold arm runs full x with canonical plan σ. z_1, z_2 denote donor-hot readouts, z_s self-hot, z_0 cold and z_i either donor. The three paired contrasts test donor choice, donor versus own cache, and restart effects. Shading marks reused KV positions, not equal values; strips are schematic. 80/80 donor pairs match L and continuation shapes, but producer lengths differ in 76/80. (b) An independent 4-pair experiment compares separately selected connections with shared M_c or M_d. Each input recomputes its own Q/K/V and hidden states. Counts cover 8 pair-plan settings (whole and 2048-chunked plans) using fp32 selection estimates; self-replay matches 16/16. No task answers are generated in (b), so this intervention does not establish the cause of answer changes in (a).

Cache-source and restart controls. We compare cold (z_0), self-hot (z_s) and donor-hot (z_d) readouts. Let $F_{\theta,\sigma}$ return the cold readout and $H_{\theta,\tau}$ the corresponding readout after continuation with supplied KV. The two hot arms fix x, L, τ and all other continuation settings:

$$\begin{aligned} z_0 &= F_{\theta,\sigma}(x), \\ z_s &= H_{\theta,\tau}(x; \mathcal{K}_{<L}(x; \theta, \sigma)), \\ z_d &= H_{\theta,\tau}(x; \mathcal{K}_{<L}(d; \theta, \sigma_d)). \end{aligned} \tag{1}$$

Self-hot uses the prefix of the consumer's complete reference cache; donor-hot uses the donor's prefix. Comparing z_d with z_s tests cache source; comparing z_s with z_0 tests restarting. These mismatch counts are not additive. A separate matched-shape intervention shares attention connections while each input recomputes its own Q/K/V and hidden states, testing selection dependence (Figure 2).

3.2 Executor, Tasks and Measurements

Reference executor and selector scope. We use Qwen2.5-7B-Instruct (Yang et al., 2024) on ascend910C with bfloat16 activations. The executor reuses KV across actual forward calls but computes dense scores before masking; its runtime is not native sparse-kernel latency. The MInference vertical/slash adaptation (Jiang et al., 2024) uses uniform $128/448$ budgets, at most 64 representative queries, 64-row execution blocks, call-length-dependent windows and budget clamps, and fp32 score estimation. On captured activations, it agrees with the released selector

in 840/840 matched-fp32 whole/first-chunk comparisons but differs from released bfloat16 estimation in 308/672 main-budget comparisons. These are input/layer/head/configuration combinations, not independent documents. The released per-head configuration, cached-key tails and native CUDA execution are outside this validation (Appendix C). Our local FLEXPREFILL adaptation (Lai et al., 2025) does not yet match the released implementation consistently; its results cannot characterize the published method (Appendix A).

Tasks and input geometry. The main sample has 80 document clusters, separate from 12 development documents. Each cluster has one fixed consumer and two preselected donors asking different questions about the same document. Python source supplies code-fact and function-call multihop tasks; WikiText-2 supplies prose cloze questions (Merity et al., 2016). Request-specific text varies the shared-prefix proportion while retaining answer evidence (Table 7). Greedy decoding produces at most 24 tokens, scored by fixed answer-type rules. Removing or replacing the document did not establish an aggregate document benefit for multihop tasks. All 80 documents remain the main analysis; excluding multihop after these checks gives the exploratory 63-document sensitivity subset (Appendix B).

Reuse policies and measurements. We construct complete donor and consumer caches, then clone and crop the selected cache. Each continuation resets routing state and reuses only the selected KV; hashes verify that the original caches remain unchanged. Page-aligned reuse uses $L_{\text{page}} = 64\lfloor J/64 \rfloor$; boundary-restricted reuse uses $L_{\text{safe}} = B\lfloor J/B \rfloor$ and resumes at canonical call boundaries. Dense and sparse caches are constructed under their respective attention policies.

We compare prefix KV, final-prefill full-vocabulary logits, generated token IDs, and task correctness. Paired confidence intervals use document-cluster bootstrap resampling (Appendix B). Mask equality requires identical retained causal connections. Appendix G reports token work separately from instrumented runtime; Table 6 identifies the input collections and execution plans.

4 SOURCE SENSITIVITY AND SELECTION DEPENDENCE

We first test whether switching cache donors changes the output of a fixed consumer request. A separate experiment matches producer shapes and intervenes on attention connections to test selection-dependent changes in prefix KV.

4.1 CACHE SOURCES CHANGE FIXED-CONSUMER OUTPUTS

Policy	Logits differ	Generation differs	Score differs
Dense hot	15/80	2/80	0/80
Dense safe	0/80	0/80	0/80
Sparse reuse	76/80	11/80	4/80
Sparse safe	0/80	0/80	0/80

Table 2: Changing donor 1 to donor 2 for the same consumers. Entries count unequal readouts over all 80 documents. Reuse lengths and continuation shapes match in every pair. Safe denotes boundary-restricted reuse.

Switching donors changed sparse generated sequences in 11/80 cases and correctness in 4/80 (Table 2). Dense reuse changed logits in 15/80 comparisons and generated sequences in 2/80, with no correctness changes. Both donors had the same reuse length and continuation shapes in 80/80 comparisons, but producer lengths differed in 76. The source comparison therefore includes both donor-content and producer-shape changes; Section 4.2 controls the latter.

Relative to self-hot, donor 1 and donor 2 changed 11/80 and 12/80 generated sequences. Self-hot differed from cold in 16/80 sequences (Table 3), showing that restarting can change outputs even with the consumer's own cache.

Correctness changes included improvements, content errors and formatting failures. One answer about the default `info` parameter of `add` changed from `None` to `False`; another retained `False` but appended ===, failing the literal scorer. Appendix D preserves every case and its original score.

Test vs. reference	Logits	Tokens	Score	$\times \to \checkmark$ / $\checkmark \to \times$
Donor 1 vs. self-hot	76/80	11/80	1/80	1 / 0
Donor 2 vs. self-hot	76/80	12/80	5/80	2 / 3
Self-hot vs. cold	76/80	16/80	4/80	2 / 2

Table 3: Sparse controls, comparing the test arm with its reference on the same 80 consumers. Logits, tokens and score columns count unequal readouts; $\times \to \checkmark$ and $\checkmark \to \times$ count incorrect-to-correct and correct-to-incorrect changes from reference to test. Donor/self-hot pairs match continuation starts and shapes; self-hot/cold pairs deliberately change the restart plan.

4.2 Selection dependence under matched producer shapes

For one head at layer ℓ, scaled dot-product attention (Vaswani et al., 2017) with an input-dependent binary mask is

$$\begin{aligned} M_\ell &= C \odot R_\ell(Q_\ell, K_\ell; \theta, \sigma), \\ O_\ell &= \mathrm{softmax}\left(\frac{Q_\ell K_\ell^\top}{\sqrt{d_h}} + \log M_\ell\right) V_\ell. \end{aligned} \tag{2}$$

Here $C_{ij} = \mathbf{1}[j \le i]$, d_h is the head dimension, and $\log 0 = -\infty$. Retained edges are causal, but R_ℓ can use later queries to select them. Those queries can therefore affect $O_{\ell,i}$ and the KV computed by subsequent layers. Appendix I gives a scalar counterexample.

The intervention uses 4 development pairs, shortened to $T = \min(S_c, S_d)$ while preserving their common prefix $[0, J)$, where S_c, S_d are the original input lengths. We test whole-sequence and 2048-split plans separately, matching input lengths, call shapes, routing units and budget clamps within each plan. For each input, dynamic routing produces a graph (M_c or M_d), which is replayed on both inputs without selector calls. Each replay recomputes its own Q/K/V and hidden states; only connections are shared. Self-replay checks the originating run. Dense controls and six sparse runs yield 64 prefills across 8 pair-plan settings. These shortened inputs test representations; no answers are generated.

Intervention	Compared region	Whole: equal	Chunked: equal
Dense, changed suffix	Prefix $[0, J)$	4/4	4/4
Dynamic routing, changed suffix	Prefix $[0, J)$	0/4	0/4
Shared consumer graph M_c	Prefix $[0, J)$	4/4	4/4
Shared donor graph M_d	Prefix $[0, J)$	4/4	4/4
Self replay (both references)	Full $[0, T)$ + logits	8/8	8/8

Table 4: All-layer equality of prefix keys, values and hidden inputs, shown separately for the two plans. Each plan uses the same four development pairs. Self-replay checks the full originating run and final-position logits, with two references per pair. No answers are generated.

Dynamic routing changed prefix KV in 8/8 settings, with differences in 27/28 layers (all layers after layer 0). Dense prefixes agreed. Sharing either graph removed the prefix differences, and all 16/16 self-replays matched their origins (Table 4). For input u, let $\mathcal{K}_{<J}(u \mid M)$ denote prefix KV when the full stack of layer masks is fixed to M. Every tested pair-plan setting satisfied

$$\begin{aligned} &\mathcal{K}_{<J}(c \mid M_c) \neq \mathcal{K}_{<J}(d \mid M_d), \\ &\mathcal{K}_{<J}(c \mid M_g) = \mathcal{K}_{<J}(d \mid M_g), \quad g \in \{c, d\}. \end{aligned} \tag{3}$$

This equality concerns the shared prefix; suffix states and final logits can differ. Under the split plan, dynamic routing preserved the complete shared call $[0, 2048)$, with differences arising in the call containing nonshared content. The intervention identifies suffix dependence through selection in the tested fp32 adaptation. Because it uses separate inputs and generates no answers, it does not quantify the contribution of selection to task-level changes.

5 CONDITIONS AND COSTS OF REFERENCE RECOVERY

5.1 SEMANTIC AND BITWISE REQUIREMENTS

Matching selection dependencies alone does not ensure bitwise invariance. A fixed-grid probe on 4 development pairs, using budgets $96/320$, matched source swaps at complete shared-block boundaries in 16/16 comparisons, but matched producer-plan changes in only 0/16. Dense plan-changing controls also differed. The following conditions separate mathematical substitution from its numerical implementation.

Proposition 1 (Semantic cache substitution)**.** *Let donor d and consumer x share prefix $[0, J)$, with reuse length $L \leq J$. For their deterministic canonical computations, assume:*

(i) Dependency closure. At every layer and position below L, each reused KV state's transitive token and routing dependencies lie within $[0, J)$.
(ii) Matching prefix computation. The reused states have identical dependency subgraphs and leaf values in donor and consumer, including non-token inputs: sequence-length parameters, candidate domains, normalization constants, and external statistics.
(iii) Canonical continuation. Continuation can perform the consumer's canonical cold-run operations. Every required non-KV state is restored or recomputed to its canonical consumer-cold value.

Then donor prefix KV can replace consumer prefix KV without changing the canonical consumer result in exact arithmetic.

Proposition 2 (Bitwise substitution under matched execution)**.** *Under Proposition 1, assume:*

(i) The compared prefix and continuation operations have identical inputs, position state and cache representation. No lossy cache conversion or unmodelled external state intervenes.
(ii) These operations execute deterministically with matching operator graphs, precisions, kernels, reduction choices, tensor layouts and relevant batch/call shapes.

Then the compared states and outputs are bitwise identical.

The proofs follow induction over dependency graphs and deterministic operations (Appendix I). The contract includes selector inputs and non-token state as well as retained attention edges.

Complete-block reuse. Suppose canonical absolute blocks have size B, and routing at every layer depends only on the current complete block and preceding blocks. Position, normalization, quantization and budget rules must introduce no nonshared dependency. Then

$$L_{\text{safe}} = B \lfloor J/B \rfloor \tag{4}$$

satisfies prefix dependency closure and allows continuation on canonical routing boundaries. A terminal short block may be incomplete for a later extension. This sufficient rule adds $0 \leq J - L_{\text{safe}} < B$ recomputed shared tokens per hit; compatible nonaligned hits can also exist. For page size p dividing B and $L_{\text{page}} = p\lfloor J/p \rfloor$, a request of length N requires

$$\begin{aligned} W(L) &= N - L, \\ \Delta W &= W(L_{\text{safe}}) - W(L_{\text{page}}) = L_{\text{page}} - L_{\text{safe}} \in \{0, p, \ldots, B - p\}. \end{aligned} \tag{5}$$

Here ΔW is extra work relative to page reuse, whereas $J - L_{\text{safe}}$ counts all recomputed shared tokens. Both exclude donor construction and measure token work rather than latency.

5.2 REFERENCE RECOVERY INCREASES TOKEN WORK

On the same 80 tasks used for the source comparisons, boundary-restricted reuse matched each policy's cold logits and generated sequences in 80/80 dense and 80/80 sparse cases (Figure 3). All 72 nonempty hits reused bitwise-equal KV at $L = 2048$; the remaining 8 cases had $J < B = 2048$, hence $L_{\text{safe}} = 0$ and no complete shared call to reuse. They therefore recomputed the full request. Recovery controls both dependency coverage and restart grouping, without separating their individual effects on answers.

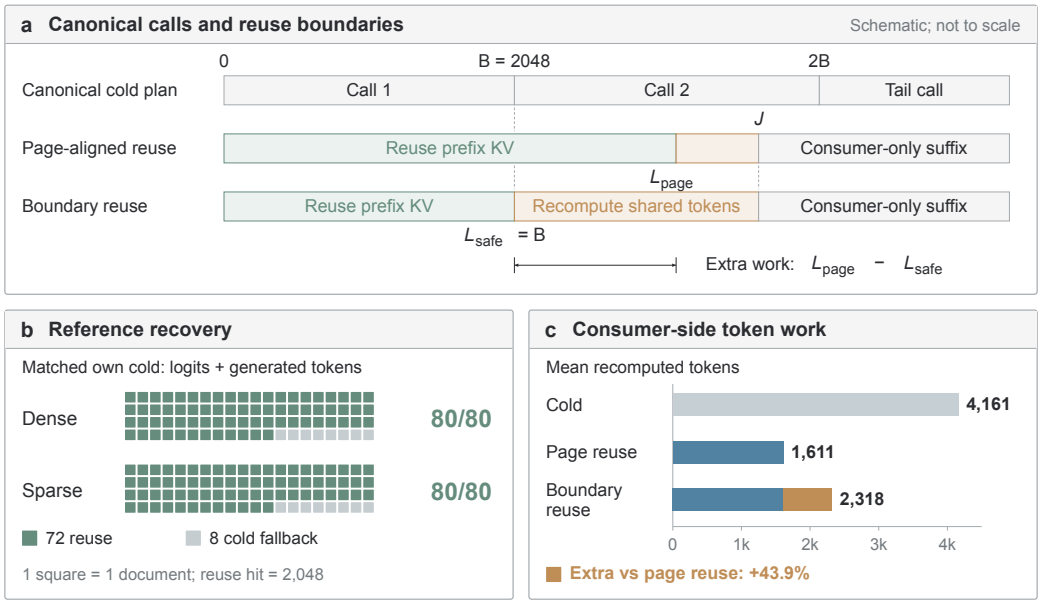


Figure 3: Canonical-boundary reuse and its token-work cost. (a) Restricting reuse to complete calls recomputes shared tokens. Only the cold call schedule is shown; page-aligned continuation may regroup calls. Extra computation relative to page reuse is $L_{\text{page}} - L_{\text{safe}}$, whereas $J - L_{\text{safe}}$ counts all recomputed shared tokens. (b) Each square represents one document matching its policy's own cold reference: green denotes nonempty reuse at 2048, gray a cold fallback. Squares are grouped by execution path. (c) Zero-based bars show descriptive mean token work; orange marks the additional work relative to page reuse. Means exclude donor construction and do not measure native latency.

Restricting reuse increased mean consumer token work from 1611 to 2318, a 43.9% increase under both attention policies. Fewer retained edges do not eliminate the extra token processing (Appendix G). Dense cold accuracy was 60.0%; sparse cold and sparse-safe achieved 57.5% and 57.5%. The dense-safe minus sparse-safe difference was +2.50 pp with paired 95% interval [-3.75, +8.75] pp. The interval spans advantages for either policy. The established benefit is preserving each policy's own cold outputs; the dense–sparse quality ranking remains unresolved. Appendix D reports all arms and the exploratory subset. A partial admission prototype is evaluated in Appendix F.

6 Policy Choice Beyond Exact Recovery

Preserving a reference does not establish which encoding policy to use. A pre-registered comparison uses the official `scbench_choice_eng` loader, prompts and scorer (Li et al., 2025), with 8/8 development and 12/12 confirmation documents. It compares dense execution (`D`), sparse context-only encoding with dense question continuation (`G`), sparse query-aware encoding (`Q`), and conservative reuse of the latter (`S`). Appendix H defines their cold/self references.

The designated `G.hot` versus `Q.cold` comparison has 47 paired questions and 11 discordant scores: the arms answer 19/47 and 20/47 correctly. The registered question-level McNemar test gives $p = 1.0000$. An exploratory bootstrap across 12 document clusters gives a question-weighted accuracy difference of -2.13 percentage points (`G.hot−Q.cold`), with 95% interval $[-13.33, +10.64]$ (Appendix H). The interval spans gains and losses; neither superiority nor equivalent quality is established. The registered rule therefore retains context-only sharing as the baseline (`pause`), pending evidence of a query-aware quality advantage.

Naive query-aware reuse changed outputs in 29/35 comparisons and scores in 6/35; conservative reuse changed neither (0/35 output changes). Dense reuse also changed outputs in 31/47 and scores in 1/47. On the common four-policy subset, `G.hot` and `Q.cold` had equal observed accuracy. Median recomputation was 53, 380 and 104384 tokens for `G`, `S` and `Q.cold`, respectively (Table 5).

Arm / encoding	Correct / n	Accuracy	Median reused tok.	Median recomputed tok.
`D.cold`: dense	16/35	45.7%	0	104384
`G.hot`: context-only	13/35	37.1%	104335	53
`Q.cold`: query-aware	13/35	37.1%	0	104384
`S.hot`: compatible	13/35	37.1%	103936	380

Table 5: Policy comparison on the 35 confirmation questions with all four arms. Each document's donor question is excluded because `S.hot` is not run for it. Accuracy and median token work use this same subset; the designated two-policy test uses all 47 questions. Work is marginal consumer work and excludes initial cache construction. The reference-executor sample and input handling differ from published SCBench evaluations.

This comparison uses call-local routing with $B = 512$: complete context calls precede the question. `G` uses dense question continuation; `Q` uses sparse continuation. Quality differences therefore cannot be attributed solely to access to the question. The 12-document comparison has incomplete records of the loaded RoPE scaling and maximum-position settings (Appendix H).

7 Discussion

Reuse guarantees depend on all inputs to prefix construction, including selector statistics and positional state. Block-Attention and MiniPIC offer precedents for controlling these dependencies (Ma et al., 2025; Ordonez & Parnell, 2026). Proposition 1 supplies a sufficient criterion for a particular implementation; Table 17 connects each reuse objective to its required checks.

Boundary restriction recovers the reference with shorter cache hits. The relative token overhead, $(L_{\text{page}} - L_{\text{safe}})/(N - L_{\text{page}})$, depends on the unreused suffix as well as prompt length (Appendix G). Applications requiring identical outputs need exact-reuse checks; policies allowing output variation need paired quality evaluation and full-workload cost, including cache construction.

Evidence comes from one model and a masked-dense adaptation with uniform budgets. Extending the graph intervention to the answer tasks and testing additional reuse boundaries would establish how broadly these mechanisms apply. Independent task collections and native-backend measurements are needed to assess generality and serving costs beyond the dense-score computation and diagnostic overhead recorded here.

8 Conclusion

Prefix-cache compatibility depends on both state construction and continuation. Our controls expose these dependencies in adaptive sparse prefill, and sufficient conditions yield a complete-call rule that recovers reference outputs at additional token work. The evaluated sample leaves the quality advantage of query-aware recomputation over context-only sharing unresolved.

Reproducibility statement

Tables and numeric macros are regenerated from archived records for a frozen task collection, with source hashes documenting provenance. Appendices B and D document scoring, uncertainty and correctness-changing pairs; Appendix A identifies implementations. The pre-registered policy protocol, subsequent releases and replica-field corrections are archived. Table regeneration checks the recorded analysis without rerunning device experiments.

Use of large language models

AI tools were used for manuscript polishing, experimental code implementation, and figure preparation. The authors are responsible for the final text, code, and figures.

REFERENCES

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 117–134, 2024. URL https://www.usenix.org/conference/osdi24/presentation/agrawal. arXiv:2403.02310.

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 (Volume 1: Long Papers)*, pp. 3119–3137, 2024. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=ayi7qezU87.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 16344–16359, 2022. doi: 10.52202/068431-1189. URL https://proceedings.nips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 6, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/a66caa1703fe34705a4368c3014c1966-Abstract-Conference.html.

Horace He and Thinking Machines Lab. Defeating nondeterminism in LLM inference. Thinking Machines Lab: Connectionism, 2025. URL https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/. Engineering blog, September 10, 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? In *Conference on Language Modeling (COLM)*, 2024. URL https://arxiv.org/abs/2404.06654.

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. In *International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24391–24402, 2025. URL https://proceedings.mlr.press/v267/hu25j.html.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 52481–52515, 2024. doi: 10.52202/079017-1663. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5dfbe6f5671e82c76841ba687a8a9ecb-Abstract-Conference.html. arXiv:2407.02490.

Dalton Jones, Junyoung Park, Matthew Morse, Mingu Lee, Chris Lott, and M. Harper Langston. QUOKA: Query-oriented KV selection for efficient LLM prefill. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/760fe89b5a5e5792b33f23a2db1b9fbf-Abstract-Conference.html. arXiv:2602.08722.

Jordan Juravsky, Bradley Brown, Ryan Ehrlich, Daniel Y. Fu, Christopher Ré, and Azalia Mirhoseini. Hydragen: High-throughput LLM inference with shared prefixes, 2024. URL https://arxiv.org/abs/2402.05099. arXiv preprint, arXiv:2402.05099.

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 (NeurIPS)*, volume 38, 2025. doi: 10.52202/085713-5585. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/f4eaa4b8f2d08edb3f0af990d56134ea-Abstract-Conference.html.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *ACM Symposium on Operating Systems Principles (SOSP)*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. FlexPrefill: A context-aware sparse attention mechanism for efficient long-sequence inference. In *International Conference on Learning Representations (ICLR)*, pp. 963–989, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/03645743ea35690f30d795d6bac149a5-Abstract-Conference.html. arXiv:2502.20766.

Yucheng Li, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir H. Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, and Lili Qiu. SCBench: A KV cache-centric analysis of long-context methods. In *International Conference on Learning Representations (ICLR)*, pp. 66063–66093, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a540b17fb2295c736d5afd6c507acf66-Abstract-Conference.html.

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 (NeurIPS)*, volume 37, pp. 22947–22970, 2024. doi: 10.52202/079017-0722. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/28ab418242603e0f7323e54185d19bde-Abstract-Conference.html.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024a. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. CacheGen: KV cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pp. 38–56, 2024b. doi: 10.1145/3651890.3672274. URL https://doi.org/10.1145/3651890.3672274.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Yutao Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. MoBA: Mixture of block attention for long-context LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025. doi: 10.52202/085713-0602. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19eae75beed66321d62272e794a9c2ac-Abstract-Conference.html.

Dongyang Ma, Yan Wang, and Tian Lan. Block-attention for efficient prefilling. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a03037317560b8c5f2fb4b6466d4c439-Abstract-Conference.html.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843. arXiv preprint.

Nathan Ordonez and Thomas Parnell. MiniPIC: Flexible position-independent caching in $<$ 100LOC, 2026. URL https://arxiv.org/abs/2606.13126. arXiv preprint, arXiv:2606.13126.

Jiwon Song, Dongwon Jo, Beomseok Kang, and Jae-Joon Kim. CompactAttention: Accelerating chunked prefill with block-union KV selection, 2026. URL https://arxiv.org/abs/2605.16839. arXiv preprint, arXiv:2605.16839.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context LLM inference. In *International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47901–47911, 2024. URL https://proceedings.mlr.press/v235/tang24l.html. arXiv:2406.10774.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

vLLM contributors. Batch invariance. vLLM documentation, 2026. URL https://docs.vllm.ai/en/latest/features/batch_invariance/. Accessed 2026-09-25; verified against official source commit 711fc55; backend support is version-dependent.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, pp. 21875–21895, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5e5fd18f863cbe6d8ae392a93fd271c9-Abstract-Conference.html.

Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. XAttention: Block sparse attention with antidiagonal scoring. In *International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 69819–69831, 2025. URL https://proceedings.mlr.press/v267/xu25ag.html. arXiv:2503.16428.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115v1. arXiv preprint, version 1.

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 *European Conference on Computer Systems (EuroSys)*, pp. 94–109, 2025. doi: 10.1145/3689031.3696098. URL https://doi.org/10.1145/3689031.3696098.

Lu Ye, Ze Tao, Yong Huang, and Yang Li. ChunkAttention: Efficient self-attention with prefix-aware KV cache and two-phase partition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11608–11620, 2024. doi: 10.18653/v1/2024.acl-long.623. URL https://aclanthology.org/2024.acl-long.623/.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for transformer-based generative models. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 521–538, 2022. URL https://www.usenix.org/conference/osdi22/presentation/yu.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, 2025. doi: 10.18653/v1/2025.acl-long.1126. URL https://aclanthology.org/2025.acl-long.1126/.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710, 2023. doi: 10.52202/075280-1506. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/6ceefa7b15572587b78ecfcebb2827f8-Abstract-Conference.html.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 62557–62583, 2024. doi: 10.52202/079017-2000. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html. arXiv:2312.07104.

A EXPERIMENT OVERVIEW AND ARCHIVED RECORDS

Appendices A–C describe experiment records, tasks and implementation checks. Appendices D–H present source, mechanism, admission, cost and policy results. Appendices I–J provide theoretical justification and diagnostic guidance.

Table 6 maps descriptive experiment names to their archived records. Archive identifiers preserve provenance; they are not consecutive experiment numbers in this paper. The main source and recovery experiment uses release `V4-r5` of archive `V4`.

Experiment	Archive	Inputs	Setup and readout
Released-selector check	D1	4 dev inputs	Released vs. local selected indices and masks
Source and recovery	V4-r5	80 documents	Call-local, $B = 2048$, $128/448$; KV, logits, outputs and scores
Shared-connection intervention	D2	4 dev pairs	Matched shapes; whole and 2048-split; prefix KV, no answers
Earlier fixed-plan check	N2	16 pairs, 14 documents	Fixed-grid, $B = 2048$, $96/320$; four option logits
Execution-plan intervention	V1	4 dev pairs	Fixed-grid, $B = 2048$, $96/320$; state/output agreement
Main-sample re-examination	V7	Previously measured main-sample documents	Layer and row of first KV difference
New-document diagnostics	V8	64 new documents	Call-local, $B = 2048$, $128/448$; prefix KV and first-difference locations
Cache-admission prototype	D0	2 dev inputs	Recorded calls and metadata; decisions and accepted-run agreement
Policy comparison	D3	8 dev + 12 confirm documents	Call-local, $B = 512$, $128/448$; outputs, scores and token work

Table 6: Experiment records in the order of their supporting material. The shared-connection intervention reuses the selector-check inputs; the admission inputs overlap both. The policy comparison has 47 paired questions, of which 35 support all four policies. Rows are not pooled. Full identities, overlaps and source hashes remain in the archived records.

The five prespecified predictions in Table 11 follow the archived order H3–H7 for the new-document diagnostics. Earlier labels H1–H2 refer to prior block-boundary checks. The table uses prediction content to identify each test.

Backend. Records identify model snapshot `a09a35458c702b33eeacc393d103063234e8bc28`. The admission prototype and shared-connection intervention use bfloat16 activations on `npu:0`. Their backend identifier is `npu:torch2.11.0+cu130:eager`; `cu130` identifies the package build, while the recorded device is an NPU.

The released MInference configuration varies patterns and budgets by head. Our main experiment uses uniform budgets for vertical/slash selection. An upstream audit verifies the call-length budget clamp and at most the last 64 representative queries. The released-selector check executes the subroutine within its supported shapes (Appendix C). Mask expansion, coordinate-adjusted cached-key cases, fp32 estimation and masked-dense execution remain local implementation choices.

Our FlexPrefill adaptation initially omitted the query-aware block union. After restoring it, only 7/12 real-activation comparisons matched the reference. These local implementation mismatches prevent attributing its results to the published FlexPrefill method.

Routing and call grouping. The earlier fixed-plan check and execution-plan intervention hold logical $B = 2048$ routing units fixed, use $96/320$ budgets and can combine complete units in one physical call. The main source experiment and new-document diagnostics instead use calls of at most 2048 rows, $128/448$ budgets clamped by call length, and new suffix-anchored calls on restart. The shared-connection intervention uses the latter budget rule on equal-length pairs, testing whole and 2048-split plans separately. Restarting away from a canonical call boundary changes sparse

grouping. These configurations answer different questions, so their effects and quality scores are reported separately.

B TASK CONSTRUCTION AND SCORING

Main sample	Observed coverage
Tasks	fact: 27, multihop: 17, cloze: 36
Consumer tokens	2151–7940 (median 3694.5)
Document tokens	1808–3392 (median 2500.5)
Page-aligned hit	1856–3392 (median 2560)
Boundary-restricted hit	0 tokens: 8/80; 2048 tokens: 72/80

Table 7: Coverage of the main 80-document sample. Fact and multihop questions use code; cloze questions use prose. Token lengths include the recorded prompt components where applicable.

The frozen tasks use Python source and WikiText-2, with stored evidence offsets. Code-fact answers come from syntax-tree properties; multihop questions ask about a module-level function called by another function. Each cloze answer occurs uniquely in a sentence retained verbatim from the source. Requests contain a document, a question and, where specified, additional request-specific text that varies the shared-prefix proportion. The 80 paired units are identified jointly by cluster name and document SHA256; a repeated filename alone does not identify the document text (Table 7).

Scoring. Greedy decoding produces at most 24 new tokens. The scorer uses the first answer line. Integer answers use the first matched signed integer and exact comparison. Literal answers remove surrounding whitespace, trailing periods and optional symmetric quotes before exact comparison. Span answers collapse whitespace, strip edge punctuation and case-fold before exact comparison. Substring matching is disabled. Archived outputs retain both raw text and the fixed scorer's verdict.

Uncertainty and exploratory selection. For paired arms a, b, let Y_i^a be the generated sequence and $s_i^a \in \{0, 1\}$ its correctness. We report

$$\widehat{r}_{\mathrm{out}}(a,b) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[Y_i^a \neq Y_i^b], \qquad \widehat{\Delta}_{\mathrm{acc}}(a,b) = \frac{1}{n}\sum_{i=1}^{n}(s_i^a - s_i^b). \tag{6}$$

Output changes and accuracy differences answer different questions: improvements and regressions can cancel in the latter. These descriptive statistics do not assume that questions sharing a document are independent.

Paired quality intervals use 10,000 document-cluster percentile bootstrap resamples, seed 20260921 and a nominal 95% level. Both arms use the same sampled document indices. An all-zero interval establishes neither zero population risk nor non-inferiority. Donor-disagreement and mechanism tables report descriptive counts; donors, layers and calls do not add independent samples. Document-removal and wrong-document controls motivated the secondary 63-document analysis excluding multihop tasks. This selection followed observed validity checks, so it is exploratory; all 80 documents remain in the main analysis. Document hashes separate exact duplicates; they do not establish independence of related source material or exclude pretraining overlap.

C RELEASED-SELECTOR CORRESPONDENCE

The released-selector check executes a pinned MInference vertical/slash selection subroutine on post-RoPE Q/K captured from 4 development inputs at layers 0, 13 and 27, across all query heads. An index collector replaces the terminal GPU attention kernel; other unavailable kernel entry points fail if reached. Collector outputs are used only to inspect selected indices, not as attention results.

The supported domain covers whole prefill and its initial full chunk, with $q_{\mathrm{len}} = kv_{\mathrm{len}}$. Both selectors receive the same captured bfloat16-model activations converted to fp32; model execution remains bfloat16. Indices and expanded masks agree in 840/840 comparisons under matched arithmetic. At $128/448$ budgets, released bfloat16 and local fp32 score estimation instead yield different

Setting	Domain	Vertical diff.	Slash diff.	Mask diff.
Same fp32 arithmetic	Defined	0/840	0/840	0/840
Released bf16, main budget	Defined	309/672	532/672	308/672
Released bf16, default budget	Defined	124/168	0/168	0/168
fp32, chunk tail (global)	Adapted	0/420	0/420	0/420
fp32, short tail	Adapted	0/420	420/420	420/420

Table 8: Released-selector correspondence checks. Cells are input/layer/head/configuration comparisons, not independent documents. The supported domain has equal query and key lengths. Both sides use the same index-to-mask expansion, so mask agreement does not validate the released GPU kernel. Default-budget masks can saturate; cached-key tail cases remain adaptations rather than upstream validation.

masks in 308/672 comparisons. Additional default-budget checks saturate slash selection, so their unchanged masks do not establish numerical robustness.

Cached-key tails, where key length exceeds query length, lie outside the tested release entry point. Coordinate-adjusted ordinary tails have a correspondence check; very short tails expose forced-budget and tie-breaking differences. These remain serving adaptations. The check also excludes the released per-head configuration and native attention kernels.

D SOURCE SENSITIVITY AND COMPLETE QUALITY RESULTS

Table 9 reports all quality arms of the main 80-document experiment; Table 10 and cases C1–C4 detail every correctness-changing sparse donor pair.

Arm	Correct / all	All 80	Exploratory 63
Dense cold	48/80	60.0%	68.3%
Dense hot, donor 1	49/80	61.3%	69.8%
Dense hot, donor 2	49/80	61.3%	69.8%
Dense safe, donor 1	48/80	60.0%	68.3%
Dense safe, donor 2	48/80	60.0%	68.3%
Sparse cold	46/80	57.5%	63.5%
Sparse self-hot	46/80	57.5%	65.1%
Sparse reuse, donor 1	47/80	58.8%	66.7%
Sparse reuse, donor 2	45/80	56.2%	61.9%
Sparse safe, donor 1	46/80	57.5%	63.5%
Sparse safe, donor 2	46/80	57.5%	63.5%

Table 9: All quality arms of the main source and recovery experiment. Cold is the declared chunked reference. The 63-document column is an empirically selected sensitivity analysis; all 80 documents remain the main sample.

Case	Gold	Donor 1	Donor 2	Score change
C1	`4`	`2`	`4`	Improved
C2	`Ashkenazi Jewish`	`Ashkenazi Jewish`	`Polish`	Regressed
C3	`None`	`None`	`False`	Regressed
C4	`False`	`False`	`False ===`	Regressed

Table 10: Every correctness-changing sparse donor pair in the main sample, ordered by the frozen task collection. Displayed answers collapse whitespace only; archived raw text and the original scorer determine the verdict. Change is donor 1 to donor 2.

Case C1 improves; C2 and C3 change answer content and regress. C4 retains the core answer but appends ===, failing the literal scorer. Its practical impact depends on the consumer's parser. Every case retains its original score. The evidence locations below use zero-based, half-open character

offsets in the stored document. Source identifiers and complete text are preserved in the frozen collection.

C1: sympy:continued_fraction.py **Question.** In the document above, the function ‘continued_fraction‘ calls exactly one other function that is defined at module level in this same file. How many named parameters does that called function take? Count every named parameter including ‘self‘ or ‘cls‘ if present, and exclude ‘*args‘ and ‘**kwargs‘. Answer with a single integer. **Evidence.** Document character offsets $[300, 5537)$; ast: the unique module-level Name-call inside 'continued_fraction' is 'continued_fraction_periodic'; gold = len(named params of 'continued_fraction_periodic'). **Recorded evidence text.** caller ‘continued_fraction‘ at 300..2781; callee ‘continued_fraction_periodic‘ at 2784..5537

C2: wikitext2:Plum cake **Question.** One span has been replaced by ____ in the following sentence, which appears verbatim in the document above. What is the missing span? Answer with the missing text only.

Plum cake prepared with plums is also a part of ____ cuisine , and is referred to as Pflaumenkuchen or Zwetschgenkuchen . **Evidence.** Document character offsets $[1133, 1149)$; the span occurs exactly once in the document (text.count == 1); the sentence is kept verbatim. **Recorded evidence text.** Plum cake prepared with plums is also a part of Ashkenazi Jewish cuisine , and is referred to as Pflaumenkuchen or Zwetschgenkuchen .

C3: sympy:matchpy_connector.py **Question.** In the document above, what is the default value of the parameter ‘info‘ of the function ‘add‘? Answer with the Python literal exactly as it appears in the source. **Evidence.** Document character offsets $[9527, 9531)$; ast: default expression of parameter 'info' of the unique FunctionDef named 'add'. **Recorded evidence text.** None

C4: sympy:basisdependent.py **Question.** In the document above, what is the default value of the parameter ‘rational‘ of the function ‘as_coeff_Mul‘? Answer with the Python literal exactly as it appears in the source. **Evidence.** Document character offsets $[4518, 4523)$; ast: default expression of parameter 'rational' of the unique FunctionDef named 'as_coeff_Mul'. **Recorded evidence text.** False

E SELECTION AND EXECUTION DIAGNOSTICS

The diagnostics in Table 11 are grouped as exploratory execution-plan interventions, prespecified predictions on new documents, and post hoc matched-shape comparisons. Their input collections and execution settings are distinguished in Table 6.

Evidence category	Comparison / implication	Count / status
Exploratory plans	Source swap, matched plans: identical	16/16
Exploratory plans	Producer plan changed: identical	0/16
Prespecified test	Sparse: layer $\geq 1 \Rightarrow r = L_s$	46/46; observed
Prespecified test	Sparse: layer $= 0 \Rightarrow r > L_s$	6/7; falsified
Prespecified test	Dense: difference $\Rightarrow$ layer $= 0$	7/16; falsified
Prespecified test	Dense: difference $\Rightarrow r > L_s$	14/16; falsified
Prespecified test	Dense: all-layer KV equal on $[0, L_s)$	54/54; observed
Post hoc subset	Same-shape subset: sparse KV differs	21/21
Post hoc subset	Same-shape subset: dense KV differs	0/21

Table 11: Selection and execution diagnostics. The exploratory plan intervention uses previously constructed pairs and fixed-grid routing; the prespecified predictions use new documents and call-local routing. Here $L_s = L_{\text{safe}}$ and r is the first differing row within the first differing layer. Hypotheses have different conditional denominators; failed predictions are retained. These rows are not pooled task-quality evidence.

Re-examination of the main sample. The row-level probe uses the part of the main sample not used by an earlier detailed probe. Its protocol predates that run, but the documents had already con-

tributed coarse compatibility measurements. This is a specified re-examination, not an independent confirmation on unseen documents.

Predictions on new documents. The new-document diagnostics use 64 documents with no exact hash overlap with development or the main sample. Each before-run prediction is evaluated separately. When the first differing sparse layer is after layer 0, the predicted first-row alignment with the shared block boundary holds in 46/46 applicable cases. The complementary layer-0 prediction holds in 6/7; dense layer and row predictions hold in 7/16 and 14/16. The latter three fail their stated counterexample criteria. Dense common-complete-block comparisons agree in 54/54 nonempty cases.

These denominators depend on observed outcomes. The two dense counterexamples to a strictly later-than-boundary prediction have both shared-block and same-shape boundaries at row 2048; neither occurs inside a matched-shape region. Partitions inspected after the new-document diagnostics are exploratory. First-difference locations are measured after execution and cannot alone authorize reuse before execution.

F CACHE-ADMISSION PROTOTYPE

The prototype records semantic dependencies, call histories, numerical settings and cache storage state. Token matching alone does not authorize reuse. Its prefix-KV check covers 2 development inputs and 8 decisions: 2 accepts, 2 shortened hits and 4 rejections. All 4/4 executed accepts or shortened hits match cold KV, final logits and generation. Two bypass controls change KV and logits while leaving generation unchanged.

Action	Decision	n	Actual L	Cold match
Accept	ACCEPT	2	2048	2/2
Shorten	ACCEPT_REDUCED	2	2048	2/2
Plan drift	REJECT	2	0	–
Missing metadata	REJECT	2	0	–

Table 12: Cache-admission decisions on the NPU. All decisions require bitwise-equal prefix KV. Accepted and shortened continuations were executed and compared with cold runs; rejected cases are admission decisions, not successful cache-substitution trials. The two inputs overlap the selector and shared-connection checks and add no independent quality samples.

Recorded calls and cache state provide stronger checks than nominal chunk size, but rule names alone do not determine dependencies. A sequence-level budget, for example, can change with total length while retaining the same rule label. Fixed-grid representative windows and page alignment also impose joint constraints that independent rounding steps may not satisfy. The prototype incompletely enforces these joint constraints and budget dependencies; applying the sufficient conditions requires checking them together.

G WORK AND TIMING MEASUREMENTS

Marginal consumer work excludes initial cache construction. To illustrate amortization in token units, consider $R \geq 1$ requests of length N with one initial cold run and $R-1$ compatible hits of length L_{safe}. Their total prefill work is $N + (R-1)(N - L_{\text{safe}}) = RN - (R-1)L_{\text{safe}}$, compared with RN for independent cold runs. This accounting assumes equal request and hit lengths; the actual workload must also charge cache storage, transfer and eviction.

For $N > L_{\text{page}}$, the per-hit token overhead relative to page reuse is $\Delta W/(N - L_{\text{page}})$, with $\Delta W = L_{\text{page}} - L_{\text{safe}} < B$. Thus B/N bounds the penalty relative to full cold token work, but the penalty relative to a page-reuse hit can remain large when little suffix work remains. A serving break-even point requires measured construction, routing and continuation costs over a workload trace; prompt length alone supplies no latency threshold.

Complete-block reuse preserves the chosen cold output while reusing fewer tokens than page-aligned caching. At matched hit lengths, sparse-safe can retain fewer edges than dense-safe while

Arm	Computed tokens	Calls	Causal edges (10^9)	Ref. seconds
Dense cold	4161	2.55	7.854	2.53
Dense hot	1611	1.40	5.234	1.45
Dense safe	2318	1.65	6.373	1.81
Sparse cold	4161	2.55	4.232	9.42
Sparse reuse	1611	1.40	2.365	5.24
Sparse safe	2318	1.65	3.060	6.45

Table 13: Consumer-side work and diagnostic reference-executor timing in the main source and recovery experiment on all 80 documents. Token work, retained causal connections, and dense score arithmetic are distinct quantities. Timings include the reference path and its diagnostic overhead; they are not native sparse-kernel latency or serving TTFT.

still incurring routing overhead and more token work than dense page-aligned reuse. Translating these counts into a deployment comparison requires a native backend, a common workload and a specified quality constraint.

H Policy comparison and sensitivity analysis

Reference runs and paired comparisons. `D.cold` and `Q.cold` encode the full request without reuse; `S.cold` uses the corresponding `Q.cold` record. Policy `G` encodes context before seeing the question, then uses dense question continuation. Its cold/self control independently builds the same context cache and reuses it for the questions. With matching context KV and continuation settings, the two independently constructed copies produce 0/47 output changes. This tests repeatability within context-only sharing; both arms reuse context KV.

Comparison	b	c	Discordant	p
`G.hot` vs `D.cold`	2	5	7	0.4531
`Q.cold` vs `D.cold`	2	5	7	0.4531
`Q.cold` vs `G.hot`	3	3	6	1.0000
`S.hot` vs `Q.cold`	0	0	0	1.0000

Table 14: Exact paired McNemar comparisons on the 35 four-policy questions. For each ordered comparison, b counts first-arm correct/second-arm incorrect and c the reverse. Only `G.hot` versus `Q.cold` was designated in advance; its primary test uses the larger 47-question set in Section 6. The other comparisons are descriptive.

Exploratory document-level sensitivity. The exploratory analysis includes all 47 paired confirmation questions, including each document's donor question. Archived official scores contain 5 `G`-only and 6 `Q`-only correct answers. The point estimate weights questions equally. In each of 10000 replicates, we sample 12 documents with replacement, retaining all paired questions in each selected document. The replicate estimate divides the summed correctness difference by the sampled question count. Sampling uses Python's standard-library random generator with seed 20260925. The 95% percentile interval uses linear interpolation at the 2.5th and 97.5th percentiles.

Table 15 reports document-level counts and pooled estimates after omitting each document. These leave-one-document-out estimates range from -6.98 to 0.00 percentage points and measure influence, not confidence limits. With only 12 clusters, percentile coverage is uncertain. This post hoc analysis is separate from the preregistration, preserves the registered McNemar decision and does not establish equivalence. The script `scripts/make_d3_sensitivity.py` records input hashes and generates the numbers and table from archived scores without model reruns or rescoring.

Historical model configuration. All 12/12 confirmation documents completed against the registered threshold of 10/12, spanning 90,741–127,909 context tokens. Records identify snapshot `a09a35458c702b33eeacc393d103063234e8bc28` and a loaded-model value of `rope_theta=1000000.0`. Source hashes identify a loader passing model path, dtype

Document	n_d	G correct	Q correct	Δ_d (pp)	Δ_{-d} (pp)
1	3	2	3	-33.33	+0.00
2	4	2	2	+0.00	-2.33
3	4	2	2	+0.00	-2.33
4	4	1	2	-25.00	+0.00
5	4	2	2	+0.00	-2.33
6	4	1	0	+25.00	-4.65
7	4	2	0	+50.00	-6.98
8	4	2	2	+0.00	-2.33
9	4	1	1	+0.00	-2.33
10	4	1	2	-25.00	+0.00
11	4	3	3	+0.00	-2.33
12	4	0	1	-25.00	+0.00
All	47	19	20	-2.13	—

Table 15: Exploratory sensitivity of the designated `G.hot−Q.cold` comparison. Documents 1–12 correspond to archived ranks 9–20; ranks 1–8 belong to the development set. The leave-one-document-out column recomputes the pooled, question-weighted accuracy difference, in percentage points.

and eager attention without an explicit positional override. The loaded `rope_scaling`, `max_position_embeddings` and `config.json` were not archived. The recorded `scaling=none` is a constant descriptor, not a read-back value; the 131,072-token driver gate checks input length rather than the loaded position configuration. Snapshot and loader records therefore recover only part of the historical configuration. Quality comparisons remain conditional on it, while recorded hot/cold outputs can be checked directly.

Sparse selection uses call-local $128/448$ budgets and 512-token calls. The question can affect its own call, but earlier complete calls are already encoded. `G` uses dense question continuation and `Q` uses sparse continuation, so the policies differ beyond question access during encoding. The original protocol, dense-control stop-rule release, interruption records and replica-field correction are archived separately. The correction excludes timing fields, not cases selected by task score.

I JUSTIFICATION OF THE SUFFICIENT CONDITIONS

Semantic substitution. Topologically order the canonical dependency graph. Shared tokens and all required non-token leaves agree. At each node, equal inputs and identical rules yield equal values, including selector inputs and decisions. Induction gives equal prefix KV at every layer. Substitution supplies those same states to the canonical continuation, preserving its outputs. The graph includes both mask-selection dependencies and retained attention edges.

Bitwise substitution. For Proposition 2, apply the induction to deterministic implemented operations. Matching execution and cache representation supplies identical input bits at each node, giving bitwise-equal reused states and continuation outputs.

Illustrative counterexample. Take an attention-only residual block with identity scalar projections, no normalization, and one-based inputs $x^{(t)} = (0, 1, t)$ for $t \in \{-1, +1\}$. The shared prefix has $k_1 = v_1 = 0$ and $k_2 = v_2 = 1$. Let row 2 retain only

$$j_*(t) = \arg\max_{j \in \{1,2\}} q_3 k_j, \qquad M_{2j} = \mathbf{1}[j = j_*(t)].$$

Both choices are causal: $j_*(-1) = 1$ and $j_*(+1) = 2$. With one retained entry, the softmax weight is one and the attention outputs are 0 and 1. Adding the shared residual $x_2 = 1$ gives states 1 and 2; an identity key projection in the next layer preserves the difference. Current-layer prefix keys remain identical. This analytical construction illustrates selection dependence; it is not a measured model run.

J SUPPLEMENTARY COMPARISONS AND DIAGNOSTIC GUIDANCE

Work	Object or objective	Relation to this study
SCBench	Shared-context task quality across queries	Motivates the policy comparison in Section 6
MINFERENCE	Dynamic sparse attention for prefill	Provides the selector adapted in our cache-compatibility experiments
SnapKV	KV retention guided by a trailing observation window	Motivates separating cache retention from the construction of reusable prefix states
KVzip	Query-agnostic eviction of computed KV	Chooses retained states; we test how their production affects substitution
CacheBlend	Selective KV recomputation for cross-document interactions	Recomputes to restore interactions; we test recovery of a specified reference
Prompt Cache, Block-Attention, MiniPIC	Modular execution with position or dependency control	Supply design precedents; we test adaptive selection and numerical execution
This study	A fixed consumer with a substituted prefix cache	Separates source, restart and selection effects; tests conditional reference recovery

Table 16: Representative related work and its relation to prefix-cache compatibility. The table summarizes the objectives of the cited studies and their connections to our diagnostic question.

Reuse objective	Required check	Evidence in this study
Same mathematical computation	Include selector, position and budget dependencies	Common-graph intervention; conditional dependency-closure rule
Same implementation output	Also match numerical execution and cache representation	Canonical-boundary recovery; cross-plan counterexamples
Quality-constrained cost reduction	Paired task changes and full workload cost	Local answer changes and token work; native performance untested

Table 17: Choosing a reuse guarantee determines the validation required. The objectives are distinct; task accuracy alone cannot certify either exact-reuse objective.