REUSING RETAINED HISTORY FOR FAST PREFILL AFTER AGENT CONTEXT COMPACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-running AI agents often shorten their conversations by replacing older messages with a summary while keeping recent messages unchanged. When the model resumes, it normally processes these retained messages again, even though it has already computed and saved their internal key–value (KV) states. Reusing those states could save time, but the messages now follow a different history and appear at different positions. We study a simple reuse policy: process the new summary, move the cached keys to the retained messages' new positions, and keep their cached values. To assess whether this shortcut preserves the agent's next decision, we compare its replies with fresh computation of the same compressed input. We measure overall reply similarity, content agreement in both directions, and differences in proposed tool actions. In targeted Qwen3-8B replays, full-reply BGE-M3 similarity ranges from 0.780 to 1.000, yet replies that score as similar can propose different code edits. Separately, at measured native Mistral-Medium-3.5-128B compaction points, reuse makes the input-processing step 13.80× faster in geometric mean than the fastest measured fresh implementation, once the caches are prepared. Reuse therefore saves substantial computation in this measured setting, but leaves a semantic gap that task success or a single similarity score alone cannot characterize.

1 INTRODUCTION

A coding agent can read files, run tests, and revise its plan over many turns. To keep its conversation within a model's input limit, systems such as Pi and LangChain replace older interactions with a summary while retaining recent messages (Pi contributors, 2026; LangChain, 2026). Before producing the next reply, the model processes this shortened input, a step called *prefill*. The retained messages may be long, so starting from the summary can repeat considerable work the model has already done.

A key–value (KV) cache stores internal states computed as the model reads each token. It can avoid recomputation when the same tokens appear after the same preceding text at the same positions. Compaction changes both conditions for the retained messages: a new summary replaces the old history before them, and the messages move to earlier positions. Their text is unchanged, but their cached states still reflect the old history. Reusing those states therefore saves computation at the risk of changing the next reply.

This opportunity is substantial in observed agent traces. In a local snapshot of the public Pi for Excel dataset, retained messages account for a median 85.2% of the compressed observed-message input under a fixed tokenizer. The traces do not expose the complete provider prompt, so this fraction is neither a cache-hit rate nor a measured speedup. It does show that the retained portion can be much longer than the new summary.

We compare two ways to process the *same* compressed input. FRESH computes new cache states for the summary and every retained message. REUSE computes the summary but takes the retained messages' states from before compaction, moves their cached keys to the new positions, and keeps their values. This positional correction follows prior cache-reuse work (Lee et al., 2025; Shi & Lou, 2026). It fixes where the retained messages appear without recomputing how the new summary would change their representations. The method thus exposes a specific quality–efficiency question: how closely does a continuation from this cheaper cache match one from fresh computation?

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.

We treat the meaning of the next reply as the primary quality target. Two agents may both complete a task while choosing different tools, explanations, or code edits; conversely, similar-sounding replies need not specify the same executable action. We use one score for overall reply similarity (BGE-M3 cosine) and another to estimate content agreement in both directions (bidirectional AlignScore) (Chen et al., 2024; Zha et al., 2023). We also inspect tool-call arguments and follow native agents through subsequent actions and task verdicts. These views keep semantic fidelity distinct from eventual task success.

The measurements show why both speed and semantics matter. In targeted Qwen3-8B replays, full-reply BGE-M3 similarity ranges from 0.780 to 1.000, yet a pair with a score of 0.865 discusses the same issue while proposing different code edits. Separately, measured native Mistral-Medium-3.5-128B compaction points yield a 13.80× geometric-mean prepared-cache prefill speedup against the fastest measured fresh implementation. In native task continuations, the policies have the same aggregate pass count, but their tool paths and some task verdicts differ. The replay, timing, and task studies use different runs; together they establish a computational opportunity and show why a pass count alone cannot certify preserved behavior.

Our contributions are threefold:

1. **A semantic target for cache reuse.** We compare reused-cache replies with fresh replies to the same compressed input, pairing text similarity and bidirectional alignment with inspection of the proposed tool actions. Task verdicts provide a separate measure of execution consequences.
2. **A reuse path for agent compaction.** We reuse the unchanged prefix, prefill the new summary, relocate cached keys for the retained messages, and carry this policy through later agent turns. We make explicit which positional difference this corrects and which dependence on the old history remains.
3. **Measured benefits and limits.** Real traces quantify the size of the retained portion; native timing measures saved prefill work; and separate replay and task-continuation studies reveal semantic and behavioral differences that speed alone would miss.

2 Problem formulation

2.1 Context compaction with retained history

We consider a policy that summarizes older interactions and keeps a recent suffix, as used in Pi and related agent frameworks (Pi contributors, 2026; LangChain, 2026). Write the old input as $B\|H\|T$ and the compressed input as $B\|S\|T$. Here B is an unchanged prefix, H is the history replaced by summary S, and T is the retained tail. All spans refer to rendered tokens, including message delimiters. The tail is present before compaction; it is not the continuation generated afterward.

2.2 The reuse opportunity in real traces

The public Pi for Excel traces record natural-language summaries and explicit retained-message boundaries (thomasmustier, 2026). Our local snapshot contains 172 session files and 192 compaction-shaped records. Excluding 30 split-turn records leaves 162 compaction events in 55 sessions. We reconstruct the visible message ancestry at each event and retain every message from the recorded boundary to the compaction parent.

Under a fixed Qwen message rendering, the median summary contains 3,422 tokens, whereas the median tail contains 20,730. The event-level retained share,

$$\rho = \frac{|T|}{|S| + |T|}, \tag{1}$$

has median 85.2% (Figure 1). Thus, most of the compressed observed-message input consists of content that was already processed. The share describes input composition under the audit tokenizer. Actual prefill savings also depend on the model, cache transformation, and execution backend. Appendix L gives reconstruction and provenance details.

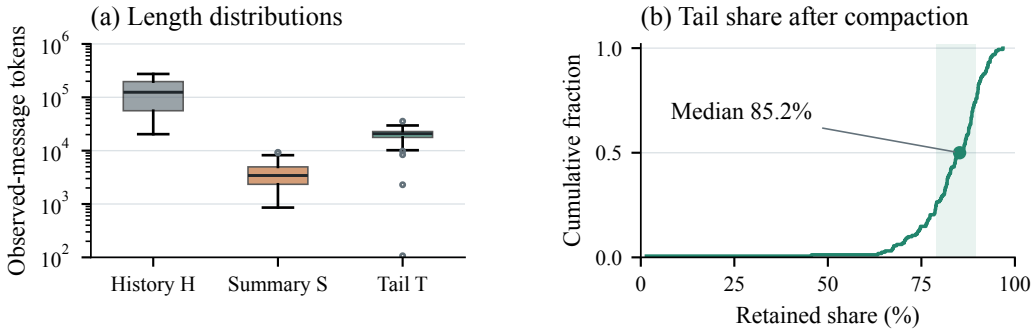


Figure 1: Retained-history opportunity in 162 Pi compaction events. Left: distributions of history, summary, and retained-tail lengths. Right: the retained share $|T|/(|S| + |T|)$ has median 85.2% and interquartile range 79.0%–89.5%. Counts use a fixed Qwen tokenizer on observed messages, excluding the unavailable system and tool prefix.

2.3 Reuse changes positions and context

Under causal attention, B keeps both its positions and its preceding context, so its KV states remain eligible for exact prefix reuse. The same is not true of T. Replacing H with S shifts the tail by $|S| - |H|$ positions and changes the context that produced its hidden representations. Correcting the positional encoding addresses the first difference; it does not recompute the history-dependent representations.

Evaluation objective. Our objective is to reduce prefill after compaction while keeping the continuation semantically close to fresh computation of the same compressed input. We compare complete replies using BGE-M3 cosine similarity and bidirectional AlignScore (Chen et al., 2024; Zha et al., 2023). These complementary measurements distinguish overall similarity from directional content agreement. Execution paths and task verdicts then assess consequences that reply-level metrics may miss. We use these scores for evaluation, rather than as an online guarantee or a threshold controlling reuse.

3 Position-repaired KV reuse

3.1 Constructing the compressed cache

Figure 2 contrasts fresh computation with reuse. Both paths start with the same unchanged prefix B. Fresh evaluates the summary and retained tail, $S\|T$, under that prefix. Reuse evaluates S, transforms the old tail cache, and joins the resulting spans. Shared spans are identified after message rendering and tokenization so that tool definitions and delimiters are included in the identity check.

3.2 Repairing the tail's positions

For a cached key at old position p, rotary position encoding gives $k_p = R(p)\bar{k}$. To place it at the new position q, we remove the old rotation and apply the new one:

$$\widetilde{k}_q = R(q)R(p)^\top k_p, \qquad \widetilde{v}_q = v_p. \tag{2}$$

This follows the positional relocation used in prior cache-reuse methods (Lee et al., 2025; Shi & Lou, 2026). With our fixed rotary convention and constant shift, the transform is $R(q - p)$. It corrects the key's positional phase, while both $\bar{k}$ and v_p retain their dependence on H. We denote this method REUSE. The UNROTATED ablation retains the same old cache without changing its positions.

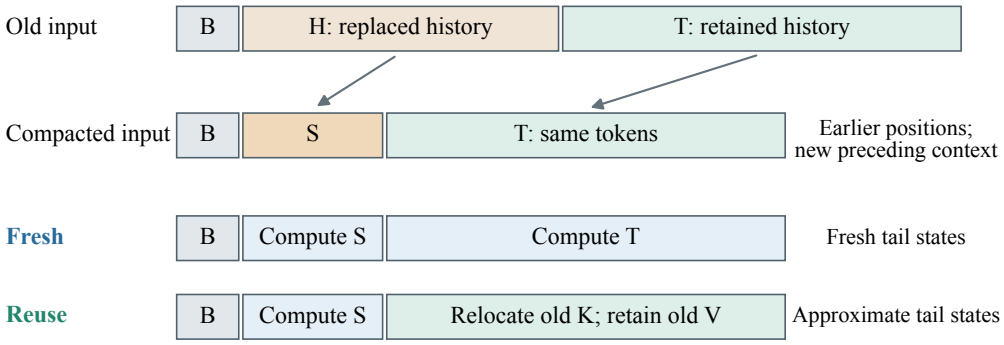


Figure 2: Cache construction after compaction. Fresh evaluates both S and T; reuse evaluates S and relocates the old keys of T while retaining its values. Both paths reuse the unchanged prefix B. Tail widths are schematic; reused states remain conditioned on the old history.

3.3 Computation saved and agent continuation

The saving comes from replacing the forward computation of most of T with key rotation and cache movement. Both fresh and reuse still prefill the already-generated summary S. Longer tails therefore offer more computation to avoid, while longer summaries increase the work common to both paths. Repair and summary computation can run independently once the summary length is known. We evaluate both serial execution and explicit overlap on separate CUDA streams.

To obtain the first-step logits under REUSE, let T^- denote the retained tail without its final token. We assemble the caches for B, S, and the repaired T^-, then evaluate the final token once against that cache. Fresh computation obtains logits at the same position as part of its ordinary prefill of T. Subsequent tokens use ordinary autoregressive decoding. In full task continuations, both arms begin with the same compressed input, tools, and workspace; after that, each follows its own generated actions and summaries. Reuse is applied again at later compactions. This setup tests a continuing reuse policy rather than forcing both agents to follow a fixed trajectory.

4 Experiments

4.1 Experimental setup

We evaluate three complementary aspects: native prefill efficiency, reply semantics under diagnostic replay, and native task continuation. Table 1 identifies the model and sample for each. The semantic replay uses Qwen3-8B caches rebuilt from Mistral text trajectories; KV tensors are never transferred between models.

Table 1: Experimental coverage. Timing, reply semantics, and task outcomes use separate runs. Controlled lengths change the timing inputs without adding quality samples.

Experiment	Model	Sample
Prefill efficiency	Mistral-Medium-3.5-128B	12 boundaries from 9 repositories
Length sensitivity	Same as above	12 summary/tail conditions
Reply semantics	Qwen3-8B	6 complete-history boundaries; 4 windowed supplements
Task continuation	Mistral-Medium-3.5-128B	46 verified pairs; 4 additional protocol exceptions

Timing and baselines. Native timing uses BF16 weights and KV on four H200 GPUs with tensor parallelism of four. The primary fresh baseline and REUSE share the same cached B and a 512-token chunk size. Two further baselines compute either the whole $S\|T$ suffix from cached

B or the whole $B\|S\|T$ input. The summary is generated before this comparison. Timing begins after cache preparation, including any uncached-tail fill and prefix preparation. REUSE includes summary prefill, key repair, cache assembly, and final-token recomputation; each fresh baseline includes its complete prefill through the final token. We use synchronized wall time, thirty repetitions per arm after five warmups, and balanced execution orders. Reported aggregates are geometric means of per-boundary median paired speedups. A controlled grid crosses three summary lengths (256, 1024, 4096) with four tail lengths (4096–16,384).

Semantic and numerical evaluation. Qwen3-8B replays use greedy generation on ten fixed boundaries. Six retain the complete history; four require history crops to fit the context ceiling and are reported separately. BGE-M3 cosine and bidirectional AlignScore compare complete first replies, including tool content. Cosine measures embedding similarity; for FRESH→REUSE alignment, fresh supplies the reference context and reuse supplies the claims to be scored; the reverse direction swaps these roles. Neither metric directly tests executable tool arguments. Natural-language-only scores are additional views, with missing values for pure-tool replies. We also compare cache tensors and next-token distributions along the same teacher-forced reply to isolate numerical effects. Details and complete scores appear in Appendix K and Appendix F.3.

Agent execution. We run native Pi continuations from a shared first summary and workspace, using a summary-plus-tail policy and saved SWE-bench harness verdicts (Jimenez et al., 2024). Subsequent actions and summaries follow each arm's own execution. The timing and quality backends differ in prefix handling: task runs recompute B, while timing enables prefix reuse in both arms. They test the same tail-reuse strategy through distinct computation paths. For local behavioral diagnostics, we also run short Qwen continuations of at most five tool-bearing turns. Appendix K records the full settings and cohort differences.

4.2 Prefill efficiency

Retained-tail reuse reduces prefill at every measured native boundary (Figure 3). The matched cached-prefix comparison yields a geometric-mean speedup of 18.54×, with boundary medians of 16.16–23.23×. These boundaries contain 11,697–15,704 retained tokens and 21–495 summary tokens. All primary ratios use overlapped REUSE as the denominator; the serial comparison appears in Section 4.4. After preparation, overlapped reuse takes 163–189 ms, compared with 3.03–4.26 s for matched fresh computation.

The benefit persists across fresh implementations. The geometric mean is 16.70× against whole-suffix computation with cached B and 13.80× against full-input recomputation. Full-input fresh is the fastest fresh path by median latency at all twelve boundaries, making 13.80× the comparison against the fastest measured baseline. This result also shows why matching the baseline matters: prefix reuse can change the attention execution path, so fewer computed tokens do not necessarily mean lower latency.

4.3 Semantic similarity and alignment

We next examine what is preserved when the cache is reused. In the complete-history Qwen replays, full-reply BGE cosine scores range from 0.780 to 1.000 (Figure 4), yet paired replies can differ in text and proposed actions. Bidirectional alignment provides a complementary view of their content: excluding task-08, the scores range from 0.433 to 0.783. Paired text inspection gives these scores a concrete interpretation. In task-04, both replies discuss preventing verification from modifying a FITS card image, but fresh proposes an edit with identical `oldText` and `newText`, whereas reuse proposes adding `card._verified = False`. Their cosine score is 0.865 despite these different proposed operations; the scores do not establish that either edit is correct.

The windowed task-07 illustrates a limitation of the metrics themselves. Its cosine is 0.971 and its directional alignment scores are 0.989 and 0.113, yet both replies issue the same test command. The reuse reply omits the phrase "to see if it passes" from the preceding sentence. The asymmetry therefore demonstrates sensitivity to a small wording change, rather than establishing a different intended action. These diagnostic comparisons identify possible mismatches; they do not estimate how often reuse preserves semantics across agent tasks.

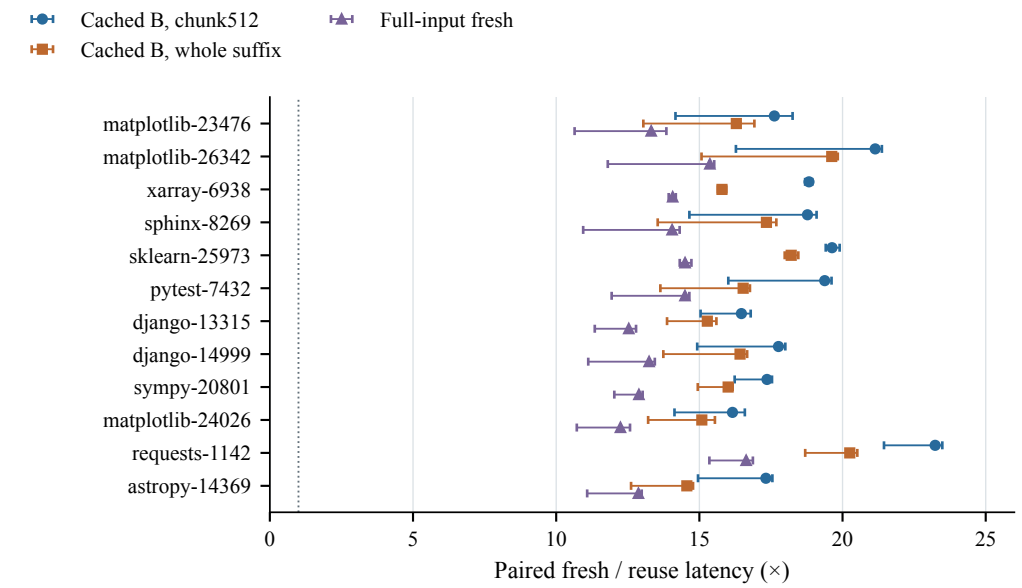


Figure 3: Prefill speedups at twelve native Mistral boundaries. Points are median paired fresh/reuse ratios; bars span the 5th–95th percentiles of thirty repetitions. The three baselines differ in prefix reuse and chunking. These ranges describe repeated timings within a boundary, not confidence intervals across tasks.

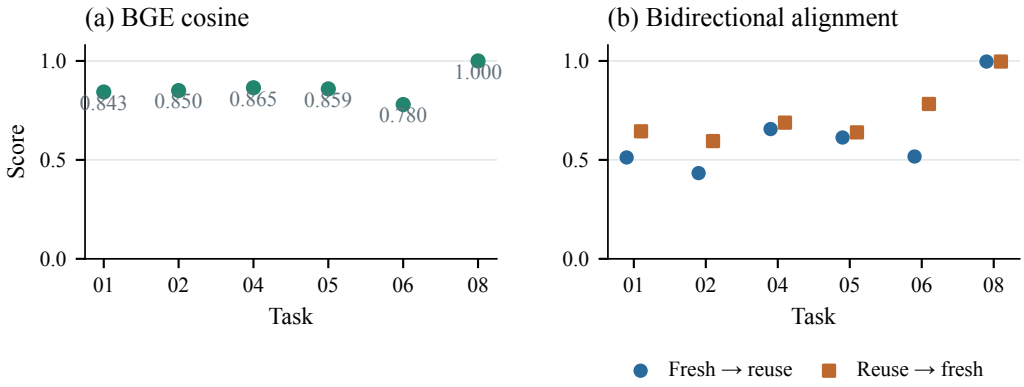


Figure 4: Reply semantics in six complete-history Qwen3-8B replays of Mistral trajectories. Each panel reports one metric on complete replies. AlignScore is shown in both fresh-to-reuse and reuse-to-fresh directions. Scores are complementary measurements with different meanings, rather than interchangeable measures of correctness.

4.4 Ablations and sensitivity analysis

Position repair reduces numerical mismatch. On every complete-history Qwen boundary, relocation reduces layer-averaged key error relative to fresh computation. The equal-task mean relative L2 error falls from 0.883 without relocation to 0.238 with it. Values retain a mean error of 0.372 because the transform leaves them unchanged. Along the common fresh-reply path, repaired per-task mean KL ranges from 5.11×10^{-6} to 0.210, compared with 8.02–13.54 for UNROTATED. Unrotated free replies all reach the 8192-token cap.

Repair thus addresses an important source of error, but it does not recover fresh states exactly (Figure 5). Repaired top-1 disagreement has an equal-task mean of 6.88% on complete-history boundaries, and task-04 contains an individual KL value of 23.15 despite a mean of 0.124. Numerical

controls and full distributions are retained in the appendix. These Qwen diagnostics explain the approximation in replay; they do not identify the cause of particular native Mistral task outcomes.

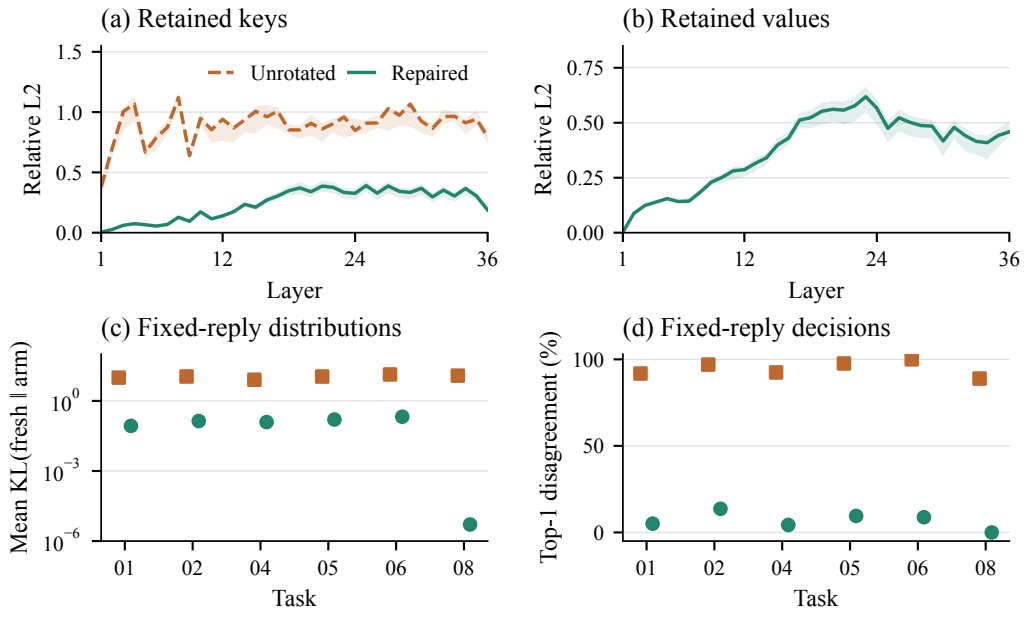


Figure 5: Diagnostic numerical replay on complete-history Qwen boundaries. Relocation reduces retained-key discrepancy compared with UNROTATED, while value and output-distribution differences remain. Task ranges and individual task points are descriptive observations, not confidence intervals over independent layers or tokens.

Long tails and short summaries favor reuse. The controlled length grid isolates the work avoided in T from the work still required for S (Figure 6). At $|S| = 256$, increasing $|T|$ from 4096 to 16,384 raises the matched speedup from 6.01× to 25.64×. At $|T| = 12,288$, increasing $|S|$ from 256 to 4096 reduces it from 18.28× to 4.41×. The smallest gain occurs at $|S| = |T| = 4096$: 1.93× against chunked fresh and 1.62× against either whole-input fresh path.

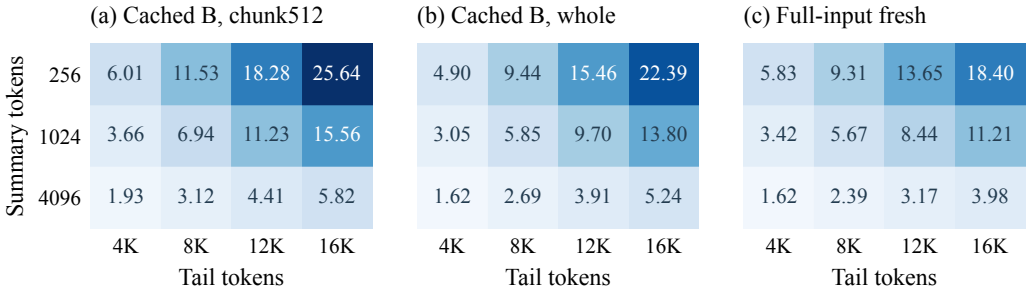


Figure 6: Prefill speedup as summary and tail lengths vary. Each cell is a median paired ratio against overlapped reuse; panels use the three fresh baselines. B is fixed. Modified token inputs isolate length effects and are used for timing only.

Explicit overlap adds no benefit here. Serial and overlapped reuse perform the same cache transformation. The geometric-mean serial/overlapped latency ratio is 0.95, with all twelve boundary medians below one. Overlap is therefore about 5% slower overall. The observed acceleration comes

from avoiding tail recomputation; overlapping summary evaluation and key repair does not improve this implementation. The comparison includes host dispatch and GPU contention, whose individual effects are not separately measured.

4.5 AGENT BEHAVIOR AND TASK OUTCOMES

Task outcomes complement the semantic measurements by showing whether changed continuations can still complete the recorded tasks. Among 46 verified native pairs, fresh and reuse each pass 29 (63.0%): 28 pairs pass in both arms, sixteen fail in both, and one passes in each discordant direction (Table 2). The four additional protocol-exception pairs pass in both arms and remain outside this primary count. These descriptive totals do not establish task-level equivalence.

Table 2: Recorded harness outcomes at the frozen cutoff. F only and R only count discordant passing outcomes. Cohorts have different scoring-script provenance, so totals are descriptive. Four pairs with incomplete all-compactions reuse verification remain in the full inventory.

Cohort	Pairs	FRESH	REUSE	F only	R only
Earlier Astropy, verified	10	9/10	9/10	0	0
Later queue, verified	36	20/36	20/36	1	1
Verified descriptive total	46	29/46	29/46	1	1
Full scored inventory	50	33/50	33/50	1	1

Of the 28 jointly passing pairs, 27 follow different tool-call paths. Twenty of these different-path pairs produce identical scored patches; seven produce distinct passing patches. Thus, a shared verdict can accompany either convergence to the same edit or different accepted edits (Figure 7). Appendices E and C retain the saved observations and reconstruction details.

Short Qwen continuations reveal a local functional failure in windowed task-11 despite identical first replies: the shared fork passes 0/6 checks, fresh reaches 6/6, and reuse remains at 0/6. Other replay cases include no actions and parse/tool failures, so equal endpoints cannot automatically be read as successful behavior preservation. This functional failure identifies a practical boundary of position-only reuse: correcting locations does not ensure that the retained states preserve every decision-relevant detail.

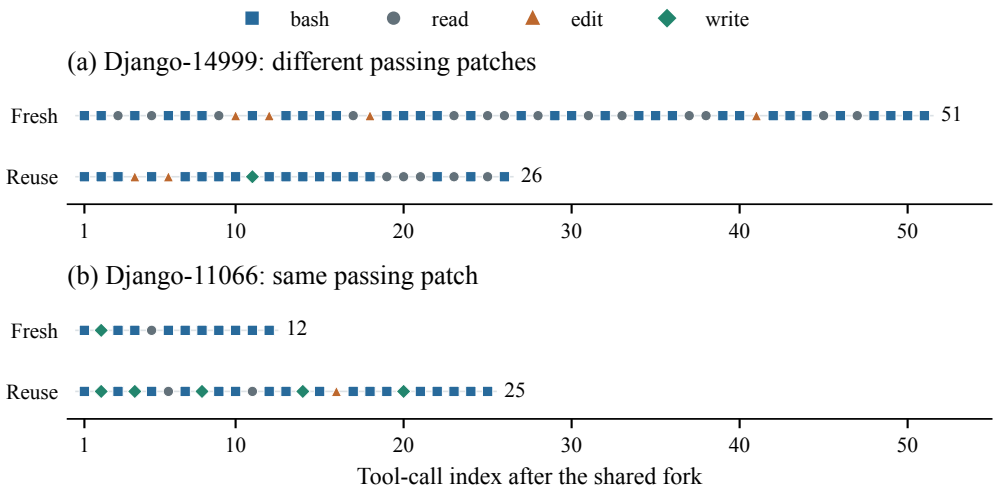


Figure 7: Illustrative native continuations. Markers show tool-call names in order; action comparison also checks arguments. Both pass: django-14999 has different submitted patches, whereas django-11066 has the same inherited source patch. Appendix C details the inherited state.

What a passing verdict can conceal. The illustrated django-14999 fork already contains changes to implementation and tests. FRESH continues by testing and revising both; REUSE explores a different path and ultimately submits the inherited tracked diff. Both scored candidates pass, but their submitted patches and tool sequences differ. In django-11066, the scored patches are instead byte-identical to an inherited source patch, although the agents take different tool paths and both remove inherited diagnostic scripts. Patch identity therefore says less about the intervening agent behavior than the complete action record. The inherited patches also mean that these passing verdicts cannot be interpreted as independent repairs from an unmodified starting point.

What a discordant verdict can mean. The paired outcomes also require case-level interpretation. On xarray-6938, FRESH fails an object-identity assertion while REUSE changes the copy behavior and passes. On sphinx-8269, the submitted patches are byte-identical and the target repair test passes in both runs; a regression test in the REUSE arm instead receives a TLS error from an external image URL. The saved verdict remains a failure under its original scoring protocol, but it is not evidence of a different implementation patch. These examples show why task verdicts complement, rather than replace, semantic and action-level comparison.

5 Related work

Agent context management. Pi and LangChain summarize older messages while retaining recent history (Pi contributors, 2026; LangChain, 2026); online compaction also explores eviction and attention matching (Liu et al., 2026). We examine the computation that survives summary replacement when recent tokens are unchanged.

KV reuse and positional correction. Rotary embeddings encode position through rotations of queries and keys (Su et al., 2021). CacheFocus relocates keys, while KVShareArena studies position-corrected reuse across contexts (Lee et al., 2025; Shi & Lou, 2026). CacheBlend selectively recomputes retrieved representations (Yao et al., 2025); ReCache adapts resource representations, and AgentKVShift adds context-dependent residual correction (Fang et al., 2026; Pandey et al., 2026). We isolate position-only reuse after agent compaction and measure both prefill savings and reply semantics.

Efficiency and quality assessment. CacheGen addresses cache compression and transfer (Liu et al., 2024), while StreamingLLM retains attention sinks and recent tokens (Xiao et al., 2024). DistServe separates prefill from decoding (Zhong et al., 2024), motivating our prepared-cache prefill timing. BGE-M3 and AlignScore provide complementary reply-quality measures (Chen et al., 2024; Zha et al., 2023), which we pair with numerical diagnostics and task continuations.

6 Conclusion

Agent compaction preserves recent text and some of the computation spent on it. Position-repaired reuse processes the new summary and relocates retained keys. In native Mistral measurements, prepared-cache prefill is 18.54× faster than matched fresh computation and 13.80× faster than the fastest measured fresh path, with the greatest benefit for a long retained tail and short summary.

Reply quality remains distinct from speed. Diagnostic Qwen replays show that similar replies can differ in content or proposed actions. Native task runs show that passing verdicts can conceal changed paths, while a windowed Qwen continuation differs functionally after identical first replies. These separate studies motivate semantic similarity and bidirectional alignment as primary quality measures, with task outcomes as complementary evidence.

AI USE STATEMENT

Generative AI assistants, including Codex, supported methodological feedback, experiment implementation, analysis and interpretation of saved results, literature lookup, figure construction, and manuscript drafting and revision. Quantitative summaries were checked against recorded experiment outputs, and citations were checked against accessible primary sources. AI-generated prose is not treated as independent experimental evidence. The authors are responsible for the final content, including AI-assisted claims, code, and artifacts.

REPRODUCIBILITY STATEMENT

The appendix specifies input spans, cache and continuation controls, semantic coverage, task-level verdicts, and timing conditions. The accompanying local artifact retains frozen inputs, source hashes, model/template identifiers, numerical token series, tool records, scored patches, and build scripts. Figure and table generation uses these saved results; regenerating summaries does not rerun model inference or the task harness. Public artifact availability is not asserted in this submission draft.

ETHICS STATEMENT

The study evaluates code-repair agents on existing software tasks and records their execution inside isolated task environments. Reported success is limited to the stated harness or local checks. The evaluation preserves failures and protocol exceptions and distinguishes approximate cache reuse from guaranteed functional equivalence.

REFERENCES

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.137. URL https://aclanthology.org/2024.findings-acl.137/.

Yichu Fang, Sitong Wei, Haozhe Hu, and Xiaoyu Shen. ReCache: Efficient KV Cache Reuse and Compression for Tool-Augmented LLM Agents. arXiv preprint arXiv:2608.19662, 2026. URL https://arxiv.org/abs/2608.19662.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

LangChain. LangChain Built-in Middleware: Summarization. Framework documentation, 2026. URL https://docs.langchain.com/oss/python/langchain/middleware/built-in. Accessed 24 September 2026.

Kun-Hui Lee, Eunhwan Park, Donghoon Han, and Seung-Hoon Na. CacheFocus: Dynamic Cache Re-Positioning for Efficient Retrieval-Augmented Generation. arXiv preprint arXiv:2502.11101, 2025. URL https://arxiv.org/abs/2502.11101.

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*. Association for Computing Machinery, 2024. doi: 10.1145/3651890.3672274. URL https://doi.org/10.1145/3651890.3672274.

Yujian Liu, Jiabao Ji, Li An, Rohit Jain, Gungor Polatkan, Siyu Zhu, and Shiyu Chang. Practical Online KV Cache Compaction for LLM Agents: An Empirical Study. arXiv preprint arXiv:2608.00902, 2026. URL https://arxiv.org/abs/2608.00902.

Mistral AI. Mistral Medium 3.5 128B. Official model card, 2026. URL https://huggingface.co/mistralai/Mistral-Medium-3.5-128B. Accessed 23 September 2026.

Nilesh Prasad Pandey, Jason Kong, Lanxiang Hu, Quanling Zhao, Yujie Zhao, Onat Gungor, Hao Zhang, and Tajana Rosing. AgentKVShift: Efficient KV Cache Reuse for Agentic Memory Systems. arXiv preprint arXiv:2607.21604, 2026. URL https://arxiv.org/abs/2607.21604.

Pi contributors. Pi Coding Agent: Context Compaction. Project documentation, pinned source revision, 2026. URL https://github.com/earendil-works/pi/blob/4bd3f48df0b14c82e8df2640645e94f82f125f44/packages/coding-agent/docs/compaction.md. Accessed 24 September 2026.

Xi Shi and Qian Lou. KVShareArena: KV-Cache Reuse Across Contexts and Model Checkpoints. arXiv preprint arXiv:2609.10266, 2026. URL https://arxiv.org/abs/2609.10266.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. arXiv preprint arXiv:2104.09864, 2021. URL https://arxiv.org/abs/2104.09864.

thomasmustier. Pi for Excel coding agent session traces. Hugging Face dataset, 2026. URL https://huggingface.co/datasets/thomasmustier/pi-for-excel-sessions.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient Streaming Language Models with Attention Sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. arXiv preprint arXiv:2505.09388, 2025. URL https://arxiv.org/abs/2505.09388.

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*. Association for Computing Machinery, 2025. doi: 10.1145/3689031.3696098. URL https://doi.org/10.1145/3689031.3696098.

Yuheng Zha, Yichi Yang, Ruichen Li, and Zhiting Hu. AlignScore: Evaluating Factual Consistency with A Unified Alignment Function. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11328–11348. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.634. URL https://aclanthology.org/2023.acl-long.634/.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 193–210. USENIX Association, 2024. URL https://www.usenix.org/conference/osdi24/presentation/zhong-yinmin.

A FROZEN BOUNDARY INVENTORY

All ten instances belong to Astropy. In this table, B and T denote the maximal shared token prefix and suffix of the rendered full and compacted inputs; H and S are the intervening token spans. These spans can include message delimiters and are not independently tokenized natural-language lengths. B is 1524 tokens for every boundary. The four nonzero crop lengths identify supplemental windowed conditions; cropping affects old-cache construction, not the frozen compacted input.

Task	Astropy issue	H tokens	S tokens	T tokens	Old-history crop
task-01	14365	9428	285	6110	0
task-02	14182	19260	351	11918	0
task-03	14539	19700	36	11981	437
task-04	14598	18792	457	11952	0
task-05	14508	19164	30	11921	0
task-06	14096	19866	259	10246	0
task-07	13236	20920	409	11824	1500
task-08	13453	18040	294	12566	0
task-10	13033	19333	345	12059	148
task-11	13579	19588	539	11924	268

The original summaries and message cuts remain unchanged. In particular, task-03 and task-05 have short replacement spans; the evaluation does not replace them with newly written summaries. Independent instance count is ten, with one boundary per instance. These Astropy boundaries constitute a fixed convenience sample, not a random cross-repository sample. Source-run identifiers, image digests and input hashes are retained in the input inventory.

B CONDITIONAL OUTPUT-DISTRIBUTION DETAIL

Each row is one boundary. KL values are in nats. The path determines the common preceding tokens supplied to both cache constructions; it does not change the stated KL direction. Positions are zero-indexed, with position 0 predicted from each arm's own boundary logits. None means no top-1 disagreement, not missing data. Small signed floating-point residuals remain in raw records. For these within-reply KL tables, Median denotes the discrete 0.5 quantile and P95 the discrete 0.95 quantile: sort the n observations and select zero-based index round($p(n-1)$), with ties rounded to the even integer. No interpolation is used; for even n, the displayed Median can differ from the average of the two central observations. This convention does not apply to the timing medians, which use the usual central-value average for even sample counts.

B.1 COMPLETE-HISTORY CONDITIONS

Fresh-reply conditioning path; KL(fresh ‖ reuse).

Task	Tokens	Mean	Median	P95	Max	First flip
task-01	415	0.08518	1.409e-08	0.3701	7.403	6
task-02	66	0.1375	0.002101	0.3365	5.316	0
task-04	278	0.1236	3.561e-07	0.2322	23.15	44
task-05	84	0.1598	3.223e-07	0.7001	3.602	1
task-06	57	0.21	0.01821	0.9932	2.683	6
task-08	36	5.112e-06	1.269e-10	2.987e-05	7.005e-05	None

B.2 WINDOWED SUPPLEMENTS

Fresh-reply conditioning path; KL(fresh ‖ reuse).

Task	Tokens	Mean	Median	P95	Max	First flip
task-03	160	0.1854	0.003191	1.223	3.193	6
task-07	66	0.03062	1.008e-08	0.02658	1.246	7
task-10	32	0.0001443	2.173e-10	9.203e-07	0.004526	None
task-11	31	3.393e-07	2.121e-12	7.986e-07	7.328e-06	None

B.3 Direction and conditioning-path sensitivity

D(F,R) denotes KL(fresh ‖ reuse); D(R,F) is the reverse direction. F/R path denotes the owner of the fixed continuation tokens.

Task	Windowed	F path: D(F,R)	F path: D(R,F)	R path: D(F,R)	R path: D(R,F)
task-01	False	0.08518	0.1155	0.1063	0.08227
task-02	False	0.1375	0.1376	0.06297	0.06332
task-03	True	0.1854	0.194	0.6055	0.6257
task-04	False	0.1236	0.06177	0.04206	0.03075
task-05	False	0.1598	0.126	0.1061	0.09618
task-06	False	0.21	0.4183	0.1514	0.1669
task-07	True	0.03062	0.04927	0.03391	0.05427
task-08	False	5.112e-06	6.14e-06	Same reply path	Same reply path
task-10	True	0.0001443	3.947e-05	Same reply path	Same reply path
task-11	True	3.393e-07	2.582e-07	Same reply path	Same reply path

A second conditioning path is measured when free fresh/reuse replies differ. When their token sequences are identical, it would duplicate the first path. Means summarize within-reply distributions, not acceptable-loss judgments. Large maxima alongside small medians motivate retaining the per-token curves; neither statistic alone measures the correctness of a reply or action.

C Native continuation cases and inherited state

The two illustrated native cases were selected to inspect different passing paths with, respectively, different and identical scored patches. They are case studies, not a random subsample. Complete tool names and arguments determine action equality; tool-name colors in the figure are only a visualization of their order.

C.1 django-14999: different submitted patches

The fork already contains changes to the migration implementation and its tests. FRESH begins with migration tests, while REUSE begins with a search for `RenameModel`. FRESH makes 51 post-fork calls and REUSE 26. Compaction counts are two and one, including the shared first boundary. Relative to the inherited files, the FRESH candidate changes the migration implementation by +34/−32 lines and the tests by +81/−1. It revises the same-table handling so that later many-to-many processing remains reachable and expands test coverage. The final REUSE tracked diff is byte-identical to the fork tracked diff. This does not imply no intermediate activity: its saved actions include temporary test edits, a diagnostic script, and later cleanup. Both scored candidate patches pass the saved harness evaluation.

C.2 django-11066: inherited source patch with cleanup

The fork already modifies `contenttypes/management/__init__.py` and contains three untracked diagnostic scripts. FRESH makes 12 post-fork calls and REUSE 25. Both submitted patches are byte-identical to the fork tracked diff. Both action records also

show successful removal of the same inherited scripts: `test_fix_verification.py`, `test_rename_contenttype_db.py`, and `test_rename_contenttype_db2.py`. Thus identical submitted patches do not imply the full workspace is unchanged. Both scored patches pass the saved harness evaluation.

C.3 Reconstruction scope

Candidate file states were reconstructed by applying the scored patches to baseline blobs captured from each content-addressed fork image. Source hashes and the resulting deltas are retained. Reconstruction does not recover the entire original endpoint, arbitrary ignored files, POSIX metadata, or transient files later removed. The recorded actions independently support the script removals. Neither fork has its own saved harness verdict: a later passing candidate that matches an inherited tracked patch is not an independent observation that the fork workspace passed.

C.4 Snapshot intake and development checks

The later queue was frozen at 2026-09-24 08:02:40 UTC. Its sixty instances contain forty completed pairs, eleven runs completed without compaction, four inference errors and five pending instances. Thirty-nine of the forty pairs have both saved scores; one remains unscored. Only scored pairs enter the outcome table. Runs without compaction contribute no paired reuse verdict. Historical development checks are retained separately: astropy-13977 passes the REUSE development check but fails both harness arms; astropy-14096 fails both development receipts but passes both harness arms. The comparison uses the saved scored candidates, not a more favorable verdict selected across protocols.

D Native generation and stopping settings

Native generation settings were checked in the saved code, model configuration, agent requests, load events, and terminal receipts for all 50 scored pairs. The historical Astropy cohort contributes 1,044 logged agent requests and the later cohort 5,507. Both request temperature 0 and seed 0; the backend actually selects tokens by argmax, without stochastic sampling. No top-p or top-k parameter is supplied. Pi is configured with thinking off and the bridge model is marked non-reasoning. This describes the Pi configuration: the backend renders the checkpoint's default chat template without an explicit reasoning-effort argument. Agent replies are capped at 8192 generated tokens, with lower per-request caps retained in the logs; the respective requested-cap ranges are 4112–8192 and 1–8192. Summary generation has a separate 1024-token cap. Both cohorts load the FP8 checkpoint with dequantization enabled into BF16, use BF16 KV storage and four-way tensor parallelism, and log the TP4 BF16-dequant CUDA-graph backend.

These settings do not mean that every run used an identical backend revision. The historical cohort contains three preserved service revisions: two runs use the earlier bucketed graph decoder, eight use the later lazy-bucket revision, and one uses the revision shared by all 39 later-cohort pairs. Their argmax selection, output cap, and BF16 loading are unchanged; graph scheduling and tool-parser implementation differ. Every preserved source file matches its per-run code-hash manifest. The historical harness-v7 designation refers to rescoring saved patches, not new model generation.

All 100 scored branch receipts end with a natural stop reason. The preserved continuation driver contains no fixed assistant/tool-round cap. The saved stage drivers impose a 6000-second watchdog per Pi stage; the later cohort's preserved outer runner additionally requests a 330-minute resource lease. The executed outer runner is not archived for the historical cohort, so identical outer lease enforcement is not claimed. These timing limits are infrastructure guards, distinct from the five-tool-round Qwen replay protocol. Detailed request locations, hashes, and verification limits are retained in the runtime audit.

E Native task-line snapshot

F/R denotes fresh/reuse. Calls count post-fork tool calls with full arguments; compactions include the shared first boundary. P/F are the saved harness pass/fail verdicts. Patch equality is byte-level

identity of scored patches, rather than a full workspace comparison. "Verified" requires both the shared-fork and all-compactions reuse checks in the original receipt.

Instance	Verdict F/R	Verified	Calls F/R	Comp. F/R	Same patch
Earlier Astropy					
astropy-12907	P/P	Yes	5/5	1/1	Yes
astropy-13033	P/P	Yes	11/11	1/1	Yes
astropy-13236	P/P	Yes	42/47	1/2	Yes
astropy-13453	P/P	Yes	27/38	1/1	Yes
astropy-13579	P/P	Yes	33/31	2/2	No
astropy-13977	F/F	Yes	117/81	4/3	No
astropy-14096	P/P	Yes	12/8	1/1	Yes
astropy-14182	P/P	No	23/18	2/2	Yes
astropy-14309	P/P	Yes	5/8	1/1	Yes
astropy-14369	P/P	Yes	11/9	1/1	Yes
astropy-14508	P/P	Yes	16/13	1/1	Yes
Later queue					
django-11477	F/F	Yes	99/94	3/2	No
django-11066	P/P	Yes	12/25	1/1	Yes
django-15957	F/F	Yes	47/59	2/2	No
django-13012	P/P	Yes	51/64	2/2	No
django-12663	P/P	Yes	77/60	2/2	No
django-15930	P/P	Yes	81/65	2/2	No
django-11433	F/F	Yes	55/29	2/1	No
django-15525	F/F	Yes	78/53	2/2	Yes
django-14580	P/P	Yes	36/41	1/1	Yes
django-11292	P/P	Yes	46/53	1/1	Yes
django-14999	P/P	Yes	51/26	2/1	No
django-13810	P/P	Yes	20/10	1/1	No
django-11087	F/F	Yes	22/32	1/1	Yes
django-11299	P/P	Yes	19/19	1/1	Yes
matplotlib-23476	F/F	Yes	47/41	1/1	Yes
matplotlib-24026	P/P	Yes	3/5	1/1	Yes
matplotlib-26342	P/P	Yes	18/27	1/1	Yes
xarray-6938	F/P	Yes	57/52	1/1	No
pytest-7432	P/P	Yes	11/13	1/1	Yes
sklearn-25973	P/P	Yes	18/18	1/1	Yes
sphinx-10323	P/P	No	24/20	1/2	Yes
sphinx-7462	F/F	Yes	9/9	1/1	Yes
sphinx-8595	F/F	Yes	46/28	2/1	Yes
sphinx-8269	P/F	Yes	11/9	1/1	Yes
sympy-17318	F/F	Yes	14/8	1/1	No
sympy-14531	P/P	No	167/177	4/3	No
sympy-19040	F/F	Yes	162/114	4/3	No
django-11138	F/F	Yes	111/80	3/2	No
django-13315	F/F	Yes	32/19	1/1	No
django-16136	P/P	Yes	11/15	1/1	Yes
django-16612	P/P	Yes	3/7	1/1	Yes
sympy-23413	F/F	Yes	198/96	6/3	No
sympy-20801	P/P	Yes	37/36	1/1	Yes
sphinx-11510	F/F	Yes	30/21	1/2	No
matplotlib-24177	P/P	No	44/45	1/2	No
sklearn-14894	P/P	Yes	9/9	1/1	Yes
django-14053	F/F	Yes	15/26	1/1	No
django-14855	P/P	Yes	14/46	1/1	No
django-15103	P/P	Yes	5/0	1/1	Yes

Four scored pairs lack all-compactions reuse verification: astropy-14182, sphinx-10323, sympy-14531 and matplotlib-24177. All four have pass/pass verdicts. The later queue's summary field named protocol verification records shared-fork identity only; the primary subset instead uses the original receipt's two separate checks. In the three later exceptions, the saved compaction/relocation

counts are respectively 2/1, 3/2 and 2/1. These pairs stay visible in the inventory and are excluded from the primary continuation-policy comparison.

Discordant verdicts. On xarray-6938, fresh fails the object-identity assertion in `test_to_index_variable_copy`, whereas reuse changes `to_index_variable` to return `self.copy(deep=False)` and passes. The arms make 57 and 52 post-fork calls, respectively, with one compaction each. On sphinx-8269, the scored patches are byte-identical and the target repair test passes in both runs. The reuse regression test receives a TLS EOF from an external Google image URL instead of the expected 404. The original saved verdict remains fail, including its original non-infrastructure-failure classification. This observation concerns a network-dependent test result; it is not evidence of a different implementation patch caused by reuse. No harness was rerun for this intake.

F Additional controls and scoring detail

F.1 Engineering controls and provenance

Noop cache reconstruction and the two additional same-path fresh constructions have zero measured discrepancies in their tested cache/logit fields. Whole-input versus chunked fresh computation retains the same first-token argmax in all ten cases while showing nonzero BF16 logits differences; the largest measured absolute difference is 0.875. That control concerns the first prediction and is reported separately from reply-wide reuse KL. Tiny signed KL residuals from floating-point arithmetic remain in raw control records rather than being treated as negative quality loss.

Earlier v4 quality results are withdrawn because its message template omitted list-valued user text, including the summary. The v5 normalizer preserves textual blocks, rejects unsupported content and checks rendered system/user/tool contents. Actual full and compact token sequences are compared with an independent CPU rendering audit before joining results. Old outputs and their replacement reasons remain archived; they are not pooled with v5.

F.2 First-reply semantic observations

All ten boundaries have full-reply BGE and bidirectional AlignScore measurements. Seven also have natural-language-only scores; the pure-tool replies at task-08/10/11 have NA natural-language measurements. All scored inputs passed saved length and no-truncation checks. AlignScore uses explicit directional segmentation with retained spans and score matrices, rather than silently shortening a long input.

Task	Full cosine	NL cosine	Full Align F→R	Full Align R→F
task-01	0.843	0.778	0.512	0.645
task-02	0.850	0.850	0.433	0.595
task-04	0.865	0.894	0.656	0.688
task-05	0.859	0.731	0.613	0.639
task-06	0.780	0.780	0.517	0.783
task-08	1.000	NA	0.997	0.997

For the six complete-history cases, full-text cosine spans 0.780–1.000. The five cases with natural-language content span 0.731–0.894 in the natural-language view. These values describe reply text and do not establish equivalence of complete tool arguments or workspace effects. Task-08's identical first reply also does not determine its later execution behavior.

The supplemental windowed task-07 illustrates why a single cosine is insufficient: full-text cosine is 0.971, while full-text AlignScore is 0.989 in the FRESH→REUSE direction and 0.113 in reverse. Its natural-language directional scores are 0.984 and 0.006. Both replies contain the same pytest command. Their introductory sentences differ only by the purpose phrase "to see if it passes", which is absent in REUSE. The saved natural-language alignment matrices each contain one score for the two complete short sentences, with no truncation. This case shows that the scorer can be highly

sensitive to a small wording difference; it does not demonstrate a change in the commanded action. It remains outside the complete-history summary. All per-view and per-direction measurements are retained, including the two other windowed cases with identical replies.

F.3 Complete first-reply semantic coverage

Full = raw complete reply; NL = natural-language view. NA indicates absence of natural language. Win. marks a windowed condition. Align columns retain direction. Token counts describe the saved BGE scoring inputs. No row has silent truncation.

Task	Win.	View	BGE	Align F to R	Align R to F	Input tokens F/R
task-01	False	Full	0.843	0.512	0.645	593/1135
task-01	False	NL	0.778	0.407	0.484	77/73
task-02	False	Full	0.850	0.433	0.595	80/310
task-02	False	NL	0.850	0.433	0.595	80/310
task-03	True	Full	0.891	0.829	0.627	189/159
task-03	True	NL	0.891	0.829	0.627	189/159
task-04	False	Full	0.865	0.656	0.688	376/1250
task-04	False	NL	0.894	0.732	0.660	112/157
task-05	False	Full	0.859	0.613	0.639	120/118
task-05	False	NL	0.731	0.301	0.630	31/44
task-06	False	Full	0.780	0.517	0.783	71/222
task-06	False	NL	0.780	0.517	0.783	71/222
task-07	True	Full	0.971	0.989	0.113	96/90
task-07	True	NL	0.927	0.984	0.006	16/10
task-08	False	Full	1.000	0.997	0.997	61/61
task-08	False	NL	NA	NA	NA	NA
task-10	True	Full	1.000	0.997	0.997	56/56
task-10	True	NL	NA	NA	NA	NA
task-11	True	Full	1.000	0.998	0.998	55/55
task-11	True	NL	NA	NA	NA	NA

G Execution controls and local functionality

G.1 Short-horizon execution

All ten paired executions have matching fork workspaces and first replies verified against their numerical runs. Workspace checks cover saved file contents and hashes, including untracked files. The six complete-history cases are reported individually below; action counts are tool calls, not assistant rounds. Parse or tool failure takes precedence over the path classification.

Task	Calls F/R	Classification	Equal endpoint	Changed files F/R
task-01	5/5	Parse/tool failure	No	1/0
task-02	0/0	No action in either arm	Yes	0/0
task-04	4/5	Parse/tool failure	Yes	0/0
task-05	1/1	Different path, same endpoint	Yes	0/0
task-06	0/0	No action in either arm	Yes	0/0
task-08	5/5	Parse/tool failure	No	1/0

Changed-file counts are relative to the fork, rather than cumulative patches inherited from the original trajectory. All four equal-endpoint main cases leave both workspaces unchanged. Task-05 therefore demonstrates different actions with no captured file changes, rather than different edits converging to a new patch. Tasks 02 and 06 provide no observation of tool-behavior preservation. Task-04 remains a failure case despite its equal endpoint.

The four windowed supplements contain one no-action pair (03), one identical action path (10), and two parse/tool-failure cases (07 and 11). The equal endpoints in 03 and 10 also remain unchanged from the fork. Strict endpoint differences concern source files in 01 and 07, a diagnostic test script in 08, and source plus diagnostic scripts in 11. File differences alone do not establish a functional regression.

Tasks 08 and 11 have identical first replies but diverge later in execution. First-reply cosine or conditional KL cannot characterize the full execution path. Two independent fresh repeats were executed for each of tasks 08, 10 and 11. In task-08, the first observed difference is a Python function address in the first returned tool text, and the repeated executions reach different endpoints. Task-10 shows no observed divergence. In task-11, the first difference occurs in the fourth returned tool text, while endpoints remain equal. These directed controls demonstrate execution variation within fresh itself; they neither establish a population stability rate nor isolate a causal effect of reuse. The frozen local checks below assess selected issue behavior at the fork and both short endpoints.

A separate task-08 diagnostic supplies the original fresh tool observation verbatim to both arms without executing another tool. The second requests have identical token hashes and 14,666 tokens. Replies differ (636 fresh tokens versus 638 reuse tokens), but both propose the same single write call with identical full arguments after ignoring call IDs and JSON object-key ordering. Both requests use the recorded same-arm reconstruction path with the identical-feedback setup described in Section 4.3. This directed probe finds a text difference under matched feedback without a difference in that action; it neither predicts subsequent endpoints nor quantifies the causal contribution of feedback variation across tasks.

G.2 Frozen local functionality and endpoint content

The checks for tasks 04, 08 and 11 were frozen before inspection of the new endpoint contents. They run in separate containers with workspace identity verified against each recorded state. Each cell reports passed checks over checks attempted within one task; checks are not independent tasks and are not pooled into a task success rate.

Task	Replay condition	Fork	Fresh endpoint	Reuse endpoint
task-04	Complete history	20/40	20/40	20/40
task-08	Complete history	2/2	2/2	2/2
task-11	Windowed supplement	0/6	6/6	0/6

Task-04 retains the same partially failing behavior already present at the fork. In task-08, the differing file is a diagnostic script, while both endpoints retain the two tested issue behaviors. Task-11 supplies a contrasting observation: fresh replaces a constant in the coordinate conversion with a dropped-dimension value lookup and passes all six local roundtrip checks; reuse retains the original expression, adds diagnostic scripts, and passes none. This is a functional difference in one supplemental windowed replay, not an estimate of complete-task success or an isolated causal effect of the cache transform.

Content verification also distinguishes strict file differences from logic changes: the task-07 source difference consists only of added comments. All six endpoint-differing files across the four cases have contents verified against their saved hashes. Unchanged tracked files are recovered from the frozen image's git blobs; missing embedded content is never interpreted as a deletion. The source artifact preserves the selected complete execution paths.

H TIMING DETAIL AND SENSITIVITY

H.1 RESIDENT-CACHE PREFILL TIMING

We measure the ten frozen Qwen3-8B replay boundaries on one RTX PRO 6000 Blackwell Server Edition GPU. Each boundary uses five warmups per arm and thirty timed repetitions, balanced across all six execution orders. The old cache remains resident in every arm; model loading, construction of that old-cache fixture, decoding and tool execution are excluded. REUSE includes new-prefix computation, cache slicing/copying, key relocation, cache assembly and the final-token bootstrap. These measurements describe boundary prefill latency, not end-to-end agent acceleration.

The GPU also contains other users' resident processes. Pre/post process inventories and one-second device telemetry are retained; aggregate activity during timing includes the benchmark itself and cannot establish absence of external interference. Device temperature reaches 91 degrees Celsius, and sampled SM clocks range from 1537 to 2430 MHz over the run, including fixture construction. Neither thermal stationarity nor isolated performance has been established. Balanced orders reduce systematic order imbalance but do not remove these limitations. All repetitions are included. The following results are explicitly shared-environment measurements.

Complete-history main group.

Task	Whole FRESH ms	Chunked FRESH ms	REUSE ms	Whole / REUSE	Chunked / REUSE
task-01	483.79	1109.65	141.10	3.43	7.87
task-02	1004.75	2514.29	161.39	6.46	15.72
task-04	1121.99	2565.41	175.33	6.26	14.56
task-05	1087.90	2488.34	159.02	6.64	15.62
task-06	909.52	2113.65	149.40	5.69	14.03
task-08	1150.97	2761.13	163.88	6.70	16.96

The equal-boundary geometric mean of median paired ratios is 5.72 against whole-input FRESH and 13.73 against frozen chunked FRESH. Repetitions do not add independent tasks.

Windowed supplements.

Task	Whole FRESH ms	Chunked FRESH ms	REUSE ms	Whole / REUSE	Chunked / REUSE
task-03	1087.27	2477.67	160.93	6.63	15.48
task-07	1068.69	2563.64	171.55	6.11	15.05
task-10	1118.19	2627.51	163.01	6.50	16.10
task-11	1146.53	2603.15	186.01	6.00	14.14

The equal-boundary geometric mean of median paired ratios is 6.31 against whole-input FRESH and 15.18 against frozen chunked FRESH. Repetitions do not add independent tasks.

A post-hoc order sensitivity analysis uses all samples: each arm has ten observations in each execution position, and each full order has five repetitions. Across the ten boundaries, the maximum-to-minimum position-specific median latency ratio ranges from 1.005 to 1.177 for whole-input FRESH and from 1.012 to 1.153 for REUSE. The smallest whole-input FRESH/REUSE order-specific median ratio across boundaries is 3.38. Thus the observed direction of the latency advantage persists across the measured orders, while its magnitude is order-sensitive. This analysis does not identify whether temperature, clocks, preceding kernels or external activity caused the variation; the balanced-order aggregate is not an isolated-hardware constant.

Latencies are medians; ratios are medians of within-repeat FRESH/REUSE ratios, not ratios of the displayed medians. Both baselines are retained because the frozen quality implementation and whole-input last-logit implementation follow different computation paths. The whole-input baseline is not a separate demonstration of identical free-generation quality. The source artifact preserves all recorded repetitions.

H.2 NATIVE MISTRAL SINGLE-BOUNDARY TIMING SUPPLEMENT

A separately frozen native Mistral-Medium-3.5-128B boundary from `django__django-14999` measures the original task backend on four H200 GPUs with tensor parallelism of four. The backend dequantizes the FP8 checkpoint to BF16 and uses BF16 KV caches. This experiment does not rebuild Qwen caches or pool results with the Qwen timing experiment. The compact input contains 14,095 tokens; REUSE retains 12,178 cached tokens, shifts their key positions by −18,778, and performs the 275 boundary-fill tokens required by the original task implementation inside the timed interval.

Each arm restores the frozen checkpoint before timing. Disk loading and fixture restoration are outside the interval; CUDA synchronization brackets prefill on each rank, and the reported duration is the maximum of the four rank-local durations. Five warmups per arm precede thirty paired measurements, alternating FRESH-first and REUSE-first orders fifteen times each. The FRESH comparator is the task-frozen implementation with chunk size 512. This earlier experiment did not measure a whole-input Mistral baseline; the expanded study in the main text does. Decode, static-cache graph preparation and tool execution are excluded.

Native single-boundary statistic	Value
FRESH median latency	3475.75 ms
REUSE median latency	735.44 ms
Median paired FRESH/REUSE ratio	4.73
Observed paired ratio range	4.32–4.79
FRESH-first median paired ratio	4.73
REUSE-first median paired ratio	4.72

All thirty pairs are retained. The ratio range describes repeated measurements of one boundary, not variation across tasks or a confidence interval over tasks. Both arms predict the same first token throughout the measurements; REUSE logits hashes vary across repetitions, so first-token agreement does not imply identical distributions or full-reply quality. This is the earlier single-boundary, full-prefix-recomputation experiment. Its ratios are retained under that original protocol and are not pooled with the expanded cached-prefix measurements.

I AGENT RUNTIME AND COMPACTION IMPLEMENTATION

The configured agent uses Pi 0.85.1 from pinned revision `4bd3f48d`. Its compaction hook selects among complete user/assistant message boundaries, retaining tool results with their calls. The custom extension uses the frozen model tokenizer and rendered prompt to choose the closest retained-tail length to 12,288 tokens, with shorter tails breaking ties and a required net reduction. This overrides upstream approximate recent-history selection. Summary generation has a 1024-token cap; only the latest summary is included on subsequent compactions. The prompt window is 40,960 with an 8192-token reserve.

Five relevant installed source files match the pinned upstream revision, and three runtime source maps contain matching source text. The current installed runtime bundles are fingerprinted in the artifact. Historical runs did not capture every bundle's hash at invocation time, so this verifies the preserved installation and driver references rather than a complete per-run binary attestation. Native quality preserves the recorded historical backend revisions; timing runs use the common later revision in a separate adapter.

J PREPARED-CACHE PERFORMANCE PROTOCOL AND DETAIL

The main experiment uses the same model checkpoint and native cache operators in a separate performance adapter. Before each timed arm, the reusable tail is available and the independent B cache is ready. The source boundary's uncached tail fill is explicitly performed before measurement; T's final token is intentionally recomputed inside it. For natural boundaries, source token identities, old

and new positions, and fill counts match the original audit. The controlled conditions truncate/repeat the anchor summary tokens and use an old cached suffix, with zero fill.

The five arms are cached-B chunk512 fresh, cached-B whole S‖T fresh, full B‖S‖T fresh, serial reuse, and overlapped reuse. Thirty formal repetitions follow five warmups. Ten balanced orders, each used three times, balance both execution position and immediate predecessor. All measured repetitions are retained. Reported latencies are medians of four-rank maximum wall times; reported ratios are medians of paired per-repeat ratios. Their geometric means weight native boundaries equally. Timing repetitions do not add independent tasks.

The original implementation-inclusive measurements are separate. They reconstruct and align the old history, perform boundary fill, and copy caches inside their timer. The prepared experiment measures a different starting state and does not establish that a serving scheduler can hide those operations. Setup diagnostics and fill counts remain in the artifact. The setup diagnostic includes copying the saved CPU fixture to GPU and is not an estimate of production preparation latency.

The main timer ends after CUDA synchronization. Separate profiled passes record S, rotation, assembly and bootstrap with device events. Rotation is local to each rank and uses a worker thread and separate CUDA stream; summary computation retains the model's TP collectives on its usual stream. Assembly waits for the repair-completion event. Stream-window intersections can include idle gaps and are not kernel-occupancy measurements. Qualification verifies exact B preservation, exact repaired T before the final token, independent output storage, and same-arm first-token agreement under reversed order.

Absolute latencies below are in milliseconds. F chunk and F cache both reuse B, computing the suffix in chunks or one forward call, respectively. F full recomputes B as well as S and T. The source artifact also retains per-repeat order, device telemetry, first tokens and every preparation record.

J.1 Native prepared boundaries

Condition	B	S	T	F chunk	F cache	F full	R serial	R overlap
astropy-14369	1474	495	12385	3271.7	2749.1	2427.7	186.3	188.7
django-13315	1474	431	11716	3028.3	2809.7	2300.3	177.0	183.9
django-14999	1474	442	12179	3174.4	2932.0	2366.8	172.5	178.7
matplotlib-23476	1474	21	12124	3029.8	2801.0	2289.9	158.3	172.0
matplotlib-24026	1474	468	11697	3032.3	2837.1	2297.3	182.7	188.0
matplotlib-26342	1474	205	13134	3447.3	3199.1	2501.9	148.7	162.9
requests-1142	1474	490	15704	4255.1	3708.0	3046.9	182.2	183.1
xarray-6938	1474	311	12287	3148.7	2637.1	2349.5	157.1	167.3
pytest-7432	1474	338	12834	3310.7	2815.8	2469.9	159.9	170.8
sklearn-25973	1474	334	12639	3302.5	3063.3	2440.9	157.7	168.2
sphinx-8269	1474	332	12313	3176.5	2934.1	2374.2	158.2	169.2
sympy-20801	1474	449	12002	3150.4	2903.3	2335.2	177.6	181.4

J.2 Controlled lengths

Condition	B	S	T	F chunk	F cache	F full	R serial	R overlap
T12288/S1024	1474	1024	12288	3328.2	2871.8	2496.9	293.8	296.4
T12288/S256	1474	256	12288	3140.3	2650.1	2341.3	158.2	171.6
T12288/S4096	1474	4096	12288	4284.8	3795.3	3078.2	968.2	972.1
T16384/S1024	1474	1024	16384	4613.8	4095.8	3327.1	296.6	296.7
T16384/S256	1474	256	16384	4409.7	3852.2	3160.4	160.9	171.9
T16384/S4096	1474	4096	16384	5661.7	5105.1	3873.4	969.4	973.2
T4096/S1024	1474	1024	4096	1131.1	947.7	1053.5	294.1	308.7
T4096/S256	1474	256	4096	969.8	790.8	935.3	144.7	161.6
T4096/S4096	1474	4096	4096	1900.9	1597.7	1585.7	968.2	981.0
T8192/S1024	1474	1024	8192	2172.7	1829.1	1777.8	295.6	312.9
T8192/S256	1474	256	8192	1995.1	1639.4	1617.1	158.9	173.5
T8192/S4096	1474	4096	8192	3030.2	2618.5	2321.9	966.1	972.3

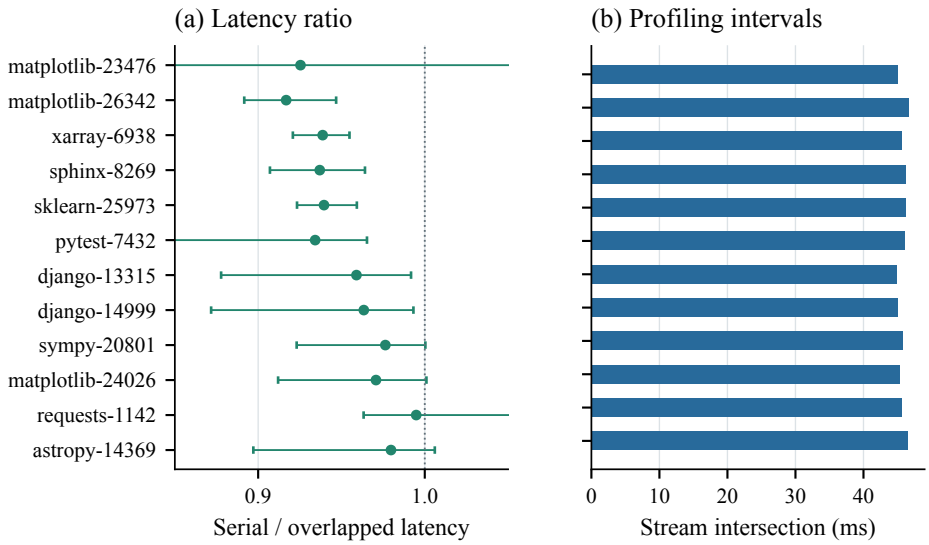


Figure 8: Sequential versus overlapped prepared reuse. Left: median paired serial/overlap ratios with the 5th–95th percentile repetition range. Right: median per-rank S/repair stream-window intersection in separately instrumented passes. The intersection need not equal simultaneous kernel execution time.

K Experimental protocol

K.1 Native timing

The model checkpoint is Mistral-Medium-3.5-128B (Mistral AI, 2026), dequantized from FP8 to BF16, with BF16 KV and tensor parallelism of four. We select twelve intact boundary checkpoints using a rule frozen before timing: one seeded choice per repository, retaining the earlier Django timing anchor, followed by the remaining maximum boundary-fill, minimum tail and maximum tail cases. The eligible pool contains 45 boundaries from the same backend revision. Selection uses neither outcomes nor latency.

Each arm starts from a prepared state: the old reusable tail is resident, required boundary fill and tail selection are complete, and an independent copy of B is ready. Two fresh paths reuse its unchanged B prefix: one computes S‖T in a single forward call, and the other uses chunks of at most 512 tokens. REUSE computes S in chunks from the same cached B, relocates T excluding its final token, assembles the cache and then computes that final token to obtain logits. Fresh paths obtain logits from their ordinary prefill without a separate bootstrap. A serial reuse path measures these operations in order; an overlapped path repairs T on a separate CUDA stream while the main thread computes S. It waits for both before assembly.

K.2 Timing boundary and repetition

Input rendering, tokenization, boundary fill, T slicing and B-cache preparation precede timing and are actually completed. Summary prefill, repair, assembly and the final-token bootstrap are inside the REUSE timer; fresh paths time their full prefill through the final token. Summary generation precedes all timed paths. Preparation costs are recorded separately; recovery latency is conditional on that work being complete. CUDA synchronization brackets each arm; we report the maximum rank-local duration. A third fresh path recomputes the full B‖S‖T input without prefix reuse. Five warmups precede thirty repetitions per arm in ten balanced orders; each of the five arms occupies every execution position six times, and each ordered immediate-predecessor pair occurs six times. Separate instrumented passes record summary and repair stream intervals; reported gains use synchronized wall time.

K.3 CONTROLLED LENGTHS

The controlled study uses tail lengths 4096, 8192, 12,288 and 16,384, and summary lengths 256, 1024 and 4096. It fixes the anchor's B, takes a suffix of the resident old cache, and truncates or repeats the saved summary tokens to set S. This isolates length dependence with zero boundary fill; the resulting token sequences are timing inputs, not additional agent-quality samples. Earlier Qwen and native single-boundary timings are preserved in the appendix under their original baselines.

K.4 NATIVE TASK CONTINUATIONS

The task snapshot combines eleven earlier Astropy pairs with 39 later scored pairs, without duplicate instances. The later queue is frozen at 2026-09-24 08:02:40 UTC. Four pairs lack confirmation of reuse at every compaction and remain outside the 46-pair primary subset. Both cohorts use a 40,960-token window, 32,768-token trigger, 12,288-token tail target and 1024-token summary cap. A pinned Pi 0.85.1 installation supplies the agent loop; a custom extension chooses the tail at complete message boundaries using the model tokenizer. This is a configured summary-plus-tail policy, not Pi's unmodified defaults.

Native runs use greedy argmax decoding, temperature 0, seed 0, Pi thinking off, and at most 8192 generated tokens per reply. The first compressed input, tools and fork workspace are shared; later summaries follow each arm's own execution. We inspect saved SWE-bench harness reports and their scored patches (Jimenez et al., 2024). Earlier Astropy scoring uses locally supplied harness dataset fields and a script that restores tests and applies the official test patch; the later cohort preserves official per-instance scripts. We report the cohorts separately and their total descriptively. Local development checks and runs ending before compaction remain separate.

The quality runs use their saved task backend, which recomputes B on compaction. The timing adapter additionally reuses B in both arms. It tests the same retained-tail strategy with ordinary prefix reuse enabled, rather than providing a second task-quality measurement of that floating-point computation path.

K.5 DIAGNOSTIC REPLAY

Ten frozen Astropy boundaries retain their original summaries and message cuts but use the Qwen3-8B tokenizer and rebuilt BF16 caches (Yang et al., 2025). Six fit the 32,768-token experimental ceiling; four require old-history crops of 437, 1500, 148 and 268 tokens and are supplements. Replay is greedy with thinking disabled and an 8192-token reply cap subject to the context ceiling. Per-layer tail K/V relative L2 uses the fresh tensor norm. The complete fresh reply supplies a common teacher-forcing path: position zero uses each arm's boundary logits, and each later position conditions on identical preceding reply tokens. We report $\mathrm{KL}(P_F \| P_A)$, its reverse, top-1 disagreement and the fresh top-1/top-2 probability margin. A different reuse reply supplies an additional fixed path.

K.6 REPLIES AND SHORT EXECUTION

BGE-M3 cosine and bidirectional AlignScore compare complete first replies; natural-language-only views are separate, with NA for pure-tool replies (Chen et al., 2024; Zha et al., 2023). Each behavioral arm starts from the same Docker fork and executes at most five assistant turns containing tools, including the first. Calls within a turn execute sequentially, with no second compaction. Action equality preserves tool names, parsed arguments, strings and array order, ignoring only call IDs and JSON object-key order. Workspace deltas include tracked, staged and nonignored untracked content relative to the fork.

L PI TRACE RECONSTRUCTION

The Pi for Excel analysis uses a local snapshot of 172 parseable session JSONL files. Among 192 compaction-shaped entries, 30 split-turn records are excluded using the export's leading "no prior history" summary marker. The remaining 162 events occur in 55 sessions. For each event, we follow its parent chain, materialize the visible messages including earlier summaries, and resolve `firstKeptEntryId`. Messages before that entry form H; all retained messages through the

compaction parent form T, including user messages. The event's natural-language summary is rendered as a user message S. No separate post-compaction query is required.

We render each complete segment once with the message loop from a fixed local Qwen2.5-32B-Instruct-GPTQ-Int4 tokenizer snapshot, then encode it once. The upstream acquisition revision of that tokenizer is unknown; the local tokenizer files are preserved with the research artifacts. The 162 event-level ratios yield a median retained share of 0.8522, with quartiles 0.7898 and 0.8951. Median H, S, and T lengths are 124,391, 3,421.5, and 20,730 tokens, respectively. These medians are calculated independently; the ratio of component medians is not the median event ratio.

Source-manifest entries identify 168 of the 172 local files. Four files have local identity records only; one of them contributes 22 compaction events. Thus, 140 of 162 events have source-manifest linkage. All events remain in the primary local-snapshot denominator. The snapshot is not a complete revision-pinned upstream census or a population sample of Pi use. Repeated events within a session are correlated; the distributions describe events, not independent tasks. Complete provider system prompts, serialized tool definitions, original token IDs, and KV block identities are unavailable. The measured share concerns observed messages and cannot establish an actual cache hit rate, serving latency, or task quality.