

Cheap but Confused?

The Accuracy Cost of Cache-Optimized Context Layouts in LLM Agents

Mohammed Abraar
 Vizudara Research
 abraar@vizz.vizudara.ai

Abstract

Prompt caching bills input tokens a provider has already seen at about a tenth of the list price, and only when a request starts with exactly the bytes of an earlier one. The layout this rewards is now the production default for agents: system prompt and tools first and frozen, every observation appended at the end, nothing reordered, nothing deleted. Because nothing is deleted, a value that changes leaves its old value in the context beside the new one. We ask whether that layout costs accuracy on its own, with the information content held fixed, and whether the cache discount pays for the cost. StaleTrap is a frozen testbed of 144 scripted tool-using episodes (720 question steps, 2,376 tool calls) in which every question is annotated as a stale-trap step, with the update either the agent’s own write or an external change revealed only inside a listing row, or a neutral step. Each episode is replayed under four layouts that carry the same information: append-only, relevance-reordered, superseded values redacted in place, and append-only plus a pointer-only session note. Every call is billed and cache-metered by the provider; five directional predictions were logged before the first scored call. First, on Gemini 3.1 Flash-Lite, own-update traps and neutral steps are answered perfectly under every layout, while on external-update traps the append-only layout answers 65.5 percent correctly against 96.0 for redaction, 70.5 for reordering and 73.5 for the note; 34 of 40 audited errors are a superseded value read back, and the gap is as large at six tool calls as at thirty. Second, the append-only layout is the cheapest per episode (1.05 cents against 1.16 for redaction) but redaction is the cheapest per correct episode (1.22 against 1.52) at the billed price and at every counterfactual cache-read price from free to full, because the provider’s best-effort cache serves the history on a minority of turns under any layout. Third, across three Gemini generations the append-only layout’s external-trap accuracy rises from 40 to 66 to 80 percent and its gap to redaction shrinks from 27 to 18 to 11 points without closing, while redaction stays the cheapest layout per correct episode on every model. The testbed, every episode log, the analysis script and the audit are released.

1 Introduction

Production agent stacks are now built around the key-value cache. Providers bill input tokens they have already processed at about a tenth of the list price, and the discount applies only when a request starts with exactly the bytes of an earlier request (Zheng et al., 2023; Lumer et al., 2026). The practitioner guidance that followed is specific: put the system prompt and the tool declarations first and never change them, append every new observation at the end, mask tools instead of removing them, and never reorder or delete earlier blocks (Ji, 2025; Anthropic, 2025). This paper calls that layout the cache-optimal layout.

The layout has a side effect that the guidance does not price. Nothing is ever removed, so when a value changes, the old value stays in the context next to the new one. A model that must answer from its context then has to prefer the later of two values for the same field. We know this is hard in general: the latest of several revisions of one fact is retrieved less reliably as revisions accumulate (Qiao et al., 2026; Bajaj et al., 2025), and agents answer from observations the world has overtaken (Liu et al., 2026b). We do not know whether the cache discipline itself, with the information content held fixed, costs accuracy, or what that cost is worth against the cache it protects. The context-management literature prunes or masks old observations and reports accuracy, but prices tokens at list rates with no cache term (Lodha et al., 2026; Lindenbauer et al., 2025; Chintalapati et al., 2026); the caching literature reports cost and latency and no accuracy (Lumer et al., 2026; Song, 2026).

We answer the question with a controlled replay. StaleTrap is a deterministic tool environment with 144 scripted episodes in which, for every question the agent answers, we know by construction whether a superseded value for the queried field is in its context and whether the update was the agent’s own write or an external change revealed only inside a listing row. Each episode is replayed under four layouts that carry the same information: the cache-optimal append-only layout (L1), a relevance-reordered layout (L2), a layout that redacts superseded values in place (L3), and the append-only layout plus a pointer-only session note (L4). Every call is billed and cache-metered by the provider, and five directional predictions were logged before the first scored call.

The answer has two halves. On the model with the full grid, Gemini 3.1 Flash-Lite, own-update traps and neutral steps are answered perfectly under every layout, so the cache discipline is free whenever the agent’s own tool replies carry the current value. On external-update traps the append-only layout answers 65.5 percent correctly, the redacting layout 96.0, the reordered layout 70.5 and the note 73.5, and 34 of 40 audited wrong answers are a superseded value read back. The cost half reverses the expected trade: the append-only layout is the cheapest per episode, but the redacting layout is the cheapest per correct episode at the billed price and at every counterfactual cache-read price, because on Google’s best-effort cache the history is rarely served from cache under any layout. Across three Gemini generations the gap shrinks from 27 to 11 points but does not close, and redaction stays cheapest per correct episode on every model.

Our contributions are:

1. StaleTrap, a frozen, hash-manifested testbed of 144 scripted tool-using episodes (720 questions, 2,376 tool calls) in which every question step is annotated as a stale-trap step (own or external update) or a neutral step (Section 3).
2. A content-fixed layout factorial (order only, content only, recitation only) scored against provider-billed cost and cached tokens, with a within-task paired analysis and one pre-registered test family (Section 3, Appendix E).
3. The finding that the accuracy cost of the cache-optimal layout is confined to external updates: zero on own updates and neutral steps, 30.5 points on external updates for Gemini 3.1 Flash-Lite, with 34 of 40 audited errors being a superseded value read back (Section 4).
4. The finding that redaction is cheaper per correct episode at every cache-read price on the tested models, which reverses the pre-registered price-threshold prediction and is explained by the measured per-turn cache behaviour (Section 4).
5. A scored pre-registration (Section 4.3) and a narrated account of what the design cannot show (Section 5).

2 What was known and what is new

Three things are established, and we do not claim them. Models use the start and the end of a long context better than its middle (Liu et al., 2023a; Hsieh et al., 2024), and the latest value of a fact revised several times inside one context is retrieved less reliably as revisions accumulate (Qiao et al., 2026; Bajaj et al., 2025). Agents answer from tool

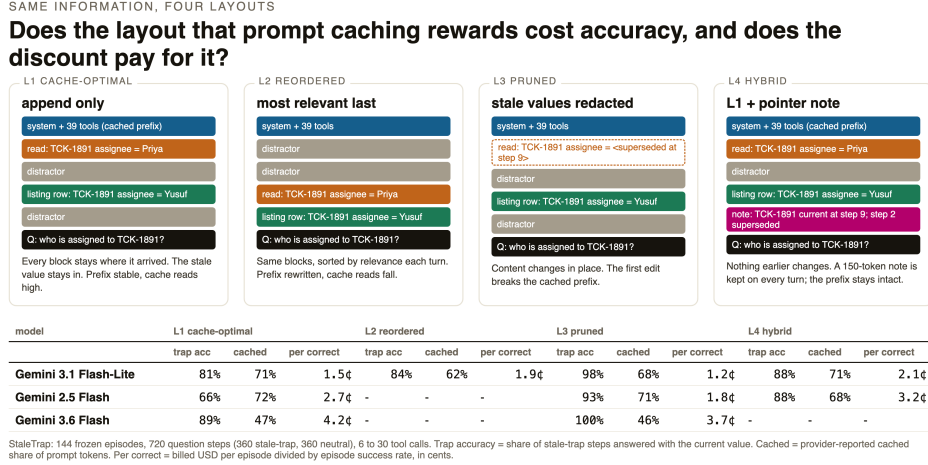


Figure 1: Study design and headline numbers. One scripted agent episode is replayed under four context layouts that carry the same information. L1 keeps every observation in arrival order (the layout that prompt caching rewards). L2 reorders earlier blocks by relevance. L3 redacts values that a later result has replaced. L4 keeps L1 and appends a short note saying which earlier observations are current. Each question step is scored against the live state of the environment, and every call is billed and cache-metered by the provider.

observations that the world has since overtaken (Liu et al., 2026b), and removing or masking old observations changes accuracy in a model-dependent way (Zhang et al., 2026; Lodha et al., 2026; Lindenbauer et al., 2025). Prompt caching cuts the cost of long agent sessions by 40 to 80 percent, and the discount is lost when earlier bytes change (Lumer et al., 2026; Song, 2026; Liu et al., 2026a).

What was not known is whether the layout discipline that caching imposes costs accuracy on its own, with the information content held fixed. Table 3 (Appendix B) lists the nearest work. The closest paper measures provider-billed cost split into cache creation and reads against task success and shows that compressing tool outputs can raise the bill by changing trajectories (Weinberger and Hozes, 2026); it varies compression, which changes content, not layout. The one evaluation of caching strategies for long agent tasks reports cost and latency and no accuracy (Lumer et al., 2026). The context-management papers that report accuracy price tokens at list rates with no cache term, so every published “compression saves X percent” is biased against append-only layouts by the cache-read discount (Lodha et al., 2026; Chintalapati et al., 2026; Erol et al., 2025; Kapoor et al., 2025). Relevance reordering was no better than shuffling in realistic retrieval (Cuconasu et al., 2025), but without a history or a cache. Ours alone: a content-fixed manipulation of layout (order only, content only, recitation only) in a multi-turn tool-using agent, scored against provider-reported cached tokens and billed dollars, with every question step annotated as a stale-trap step or a neutral step, and a per-model answer to whether a cache-read price exists at which the cheapest layout per call stops being the cheapest per correct answer. Who is affected: anyone who followed the 2025 guidance to design around the KV cache (Ji, 2025; Anthropic, 2025). If the append-only layout loses accuracy only where the world changed behind the agent, the cost of caching discipline becomes a known quantity with a known fix; if it loses none, the pruning literature’s cost numbers need a cache term.

3 Method

The question in one sentence. Hold the information in an agent’s context fixed, change only how it is laid out, and measure what happens to accuracy, cache reads and the bill.

Why a self-built testbed. To attribute an error to a stale value, we must know, for every question the agent answers, whether a superseded value for the queried field is sitting in its context. Public agent benchmarks do not annotate this. We therefore built StaleTrap, a deterministic tool environment with scripted episodes in which that annotation is known by construction and the testbed was frozen, with a recorded hash, before any scored episode ran.

Environment. Three back-office domains (warehouse inventory, a shared meeting calendar, a support-ticket queue) are exposed as 29 JSON-schema tools: 10 reads, 19 writes. A fourth domain of 10 messaging and HR tools is present in every call and never required by any task, so the catalogue is masked by the agent’s choice rather than trimmed per step, as production stacks do. A local executor holds the state and returns realistic JSON. Every result declares which (record, field) values it states: a dedicated read states the fields it returns, a write states the field it changed and reports the new value, and a listing states the fields of every row it contains. The shared prefix (system prompt plus tool declarations) measures 5,102 tokens on Google’s tokenizer, above every provider’s minimum for implicit caching. Appendix D prints it verbatim.

Episodes. An episode is a scripted session of user instructions. Most instructions ask for one tool call (“Record that we received 22 units of SKU-2249”). Some are no-tool questions (“Based only on the information already gathered in this conversation, and without calling any tools, what is the current on-hand quantity of SKU-2249?”), served with tool calling disabled, answered as a one-key JSON object and scored by exact match against the executor’s live state after canonicalisation. A trap thread reads a record’s field, later updates it one to three times, and then asks for its current value, so that a superseded value is present in context when the question arrives. A neutral thread reads a field once and asks for it later, so that nothing in context competes. Trap threads come in two kinds. In an own update, the agent’s own write returns the new value explicitly. In an external update, someone else changes the record and the agent only ever sees the new value as one row inside an unrelated listing it was asked to fetch (a bin inventory, a room schedule, a ticket queue). The second kind is the realistic source of stale observations (Liu et al., 2026b); five of nine task templates use it and four use own updates. The remaining calls are distractors. At the low level they are compact reads and writes on unrelated records. At the high level they are verbose listings, reads of near-miss identifiers that differ by one transposed digit, and writes to other fields of the very records the questions are about. Episode length is 6, 12, 18 or 30 tool calls, with 2, 4, 6 or 8 questions, and the distance from the last update to its question grows with length (2 to 3, 3 to 6, 3 to 7, 4 to 9 calls). The grid is 3 domains by 3 templates by 4 lengths by 2 distractor levels by 2 seeds: 144 episodes, 720 questions (360 trap, 360 neutral), 2,376 tool calls. Every task has two phrasings. A script verifies that every stale value in every trap differs from the final answer (360 of 360).

Layouts. Each turn, the runner rebuilds the message list from the transcript so far under one of four rules; the agent’s own replies and tool results are never altered except as the rule states, and the information content is identical for L1, L2 and L4. L1 is the cache-optimal layout: system prompt and tools first, every block in arrival order, newest observation and user turn last, tools masked by tool_choice rather than removed. L2 keeps the same blocks but stable-sorts the earlier ones by how many record identifiers they share with the current turn, most relevant last; the prefix is rewritten every turn. L3 keeps arrival order and redacts, in place, every value that a later result has restated (“<superseded at step 7>”); a result whose stated values are all superseded becomes one placeholder line. The rule is schema-level and transcript-derived, which makes it an upper bound for any real pruner, and it edits earlier bytes, so the cached prefix breaks at the first edit. L4 is L1 plus a pointer-only session note prepended to every user turn and kept there, so the history stays append-only: the session goal, the step number, and for every record with more than one observation which step is current and which steps are superseded. The note never contains a value, so it cannot leak an answer. It adds about 150 tokens per turn, and because earlier notes are never rewritten the prefix stays intact; the price is that every earlier note stays in the history, so an L4 episode sends more prompt tokens than an L1 episode (Section 4 reports how many). Appendix ?? shows one episode serialised under each layout.

Models and protocol. The plan froze three model families (Google, Anthropic, OpenAI). The OpenRouter workspace used for this study carries a guardrail that returns “no endpoints match your guardrail restrictions” for every non-Google model; it was probed before the plan was frozen and again after the main grid, and the plan’s stated fallback applies: a one-family capability ladder. Gemini 3.1 Flash-Lite runs the full grid (144 tasks by 4 layouts, 576 episodes). Gemini 2.5 Flash, an older generation with thinking enabled, runs L1, L3 and L4 on the seed-0 half of the grid (72 tasks, 216 episodes). Gemini 3.6 Flash, the newest generation, runs L1 and L3 on a 36-task subset balanced over domain, template, length and distractor (72 episodes). Gemini 3.5 Flash-Lite was smoke-tested and dropped because it returns malformed function calls on about a third of tool-calling turns with our 39-tool catalogue. The Anthropic cache on/off control could not run; Section 4 reports the Google substitute. All calls go through OpenRouter with temperature 0, reasoning effort minimal, 700 output tokens of headroom, a per-episode session identifier for sticky routing, and turns strictly sequential inside one process so that the cache never expires between turns. On Anthropic we place explicit cache breakpoints on the system block and on the newest user turn; on OpenAI caching is automatic; on Google it is implicit, because an explicit breakpoint on Gemini creates a billed cache on every call (we measured 3.5 to 5.8 times the cost of an implicit hit). Every call logs the provider’s prompt, cached, cache-write, completion and reasoning token counts and the billed dollar amount (usage.cost); no cost in this paper is modelled from a price sheet except where the text says counterfactual.

Analysis. The unit is the task. For each model we form within-task paired differences between layouts on five quantities: accuracy on stale-trap steps, accuracy on neutral steps, episode success (all questions right), billed dollars, and the cached share of prompt tokens. We report two-sided sign-flip permutation tests over tasks (20,000 flips), task-bootstrap 95 percent intervals (10,000 resamples), paired d_z , and, as a robustness column, an exact sign-flip over the nine task templates. Interactions are tested directly by label permutation: the trap gap by length (slope per call), by distractor level, trap against neutral (a difference of differences), and by model family. Benjamini-Hochberg runs over the full family of primary tests from the first run, and only survivors are bolded. For the cost side we compute dollars per correct task from billed cost, then recompute it under counterfactual cache-read prices from 0 to 1 times the input price to find where the cheapest layout changes. Forty wrong answers are hand-coded from their observation history as stale value, reveal not executed, ambiguous, arithmetic, wrong record or abstention, and shipped with the release. Two sensitivity checks guard the main contrast: trap accuracy restricted to questions whose scripted thread steps the agent actually executed (an agent that skips a listing call never sees the fresh value), and, in place of the cache on/off control, trap accuracy on question turns that Google’s best-effort cache served from cache against those it did not, paired within question on the repeat battery.

Pre-registration. Five directional predictions were written into the project log before any model call (Appendix A). We report each as confirmed, reversed or not supported in Section 4.3.

4 Results

Table 1 gives the per-layout means for every model and Table 2 the paired contrasts for the model that ran the full grid. Bold entries survive Benjamini-Hochberg over the family of The family holds 65 tests; 29 survive Benjamini-Hochberg at $q = 0.05$ (effective threshold $p \leq 0.0207$).

The cost of the cache-optimal layout is confined to external updates. On Gemini 3.1 Flash-Lite, L1 answers 80.6 percent of trap questions [75.0, 85.8] and L3 answers 98.3 percent [96.9, 99.4]; the paired difference is -0.177 $[-0.231, -0.127]$, sign-flip $p < 10^{-4}$, $d_z = -0.55$ (Table 2). Neutral questions are answered perfectly under every layout, so the trap-minus-neutral difference of differences equals the trap gap and is as significant. Splitting the trap questions by kind (Figure 2, Table 7) locates the entire effect: the 160 own-update traps are answered perfectly under every layout, and on the 200 external-update traps L1 scores

Model	Layout	n	Trap acc.	Neutral acc.	Success	Cents/episode	Cents/correct task	Cached share
Gemini 2.5 Flash	L1 append-only	72	0.660 [0.565, 0.751]	1.000	0.528	1.43	2.71	
	L3 pruned	72	0.933 [0.889, 0.971]	1.000	0.861	1.54	1.78	
	L4 hybrid	72	0.875 [0.811, 0.932]	1.000	0.778	2.45	3.15	
Gemini 3.1 Flash-Lite	L1 append-only	144	0.806 [0.752, 0.857]	1.000	0.694	1.05	1.52	
	L2 reordered	144	0.841 [0.792, 0.887]	1.000	0.729	1.35	1.85	
	L3 pruned	144	0.983 [0.969, 0.994]	1.000	0.944	1.16	1.22	
	L4 hybrid	144	0.879 [0.837, 0.917]	1.000	0.771	1.60	2.07	
Gemini 3.6 Flash	L1 append-only	36	0.889 [0.787, 0.972]	1.000	0.833	3.51	4.21	
	L3 pruned	36	1.000 [1.000, 1.000]	0.972	0.972	3.57	3.67	

Table 1: Per-layout means. Trap accuracy is the per-episode mean with a task-bootstrap 95 percent interval; neutral accuracy is the per-episode mean; success is the share of episodes with every question right; cost is the provider-billed amount per episode; cached share is cached prompt tokens over prompt tokens, summed over the episode. Gemini 2.5 Flash ran 72 tasks under L1, L3 and L4; Gemini 3.6 Flash ran 36 tasks under L1 and L3.

Contrast	Trap acc.	Neutral acc.	Success	USD/episode	Cached share
L1 - L2 (order)	-0.036 [-0.079, 0.006], $p=0.116$	+0.000 [0.000, 0.000], $p=1.000$	-0.035 [-0.090, 0.021], $p=0.334$	-0.30c [-0.36, -0.24], $p=<0.0001$	+0.089 [0.074, 0.103], $p=<0.0001$
L1 - L3 (content)	-0.177 [-0.230, -0.128], $p=<0.0001$	+0.000 [0.000, 0.000], $p=1.000$	-0.250 [-0.326, -0.180], $p=<0.0001$	-0.10c [-0.15, -0.06], $p=<0.0001$	+0.032 [0.018, 0.045], $p=<0.0001$
L4 - L3 (note)	+0.073 [0.031, 0.120], $p=0.0013$	+0.000 [0.000, 0.000], $p=1.000$	+0.076 [0.014, 0.139], $p=0.034$	+0.54c [0.44, 0.65], $p=<0.0001$	+0.003 [0.010, 0.015], $p=0.642$
L4 - L3	-0.104 [-0.145, -0.066], $p=<0.0001$	+0.000 [0.000, 0.000], $p=1.000$	-0.174 [-0.243, -0.111], $p=<0.0001$	+0.44c [0.35, 0.53], $p=<0.0001$	+0.035 [0.019, 0.050], $p=<0.0001$
L2 - L3	-0.141 [-0.188, -0.098], $p=<0.0001$	+0.000 [0.000, 0.000], $p=1.000$	-0.215 [-0.285, -0.146], $p=<0.0001$	+0.19c [0.15, 0.24], $p=<0.0001$	-0.057 [-0.069, -0.044], $p=<0.0001$

Table 2: Paired within-task contrasts on Gemini 3.1 Flash-Lite (144 tasks): mean difference, task-bootstrap 95 percent interval, two-sided sign-flip p . Bold survives BH. Cost differences are in cents.

65.5 percent, L2 70.5, L3 96.0 and L4 73.5. When the agent’s own tool reply states the current value, the stale copy beside it does no harm. When the current value arrived as one row of a listing the agent fetched for another reason, the append-only agent answers from its own earlier write about one time in three. The gap does not grow with episode length (slope +0.0006 per call, label-permutation $p = 0.85$; the gap is -0.22 at six calls and -0.22 at thirty, Table 6, Figure 6) and does not depend on the distractor level (high minus low -0.035 , $p = 0.54$). The stale value costs what it costs as soon as it exists.

Order alone does nothing; the note recovers a quarter. Reordering earlier blocks by relevance (L2) does not improve on L1: -0.036 $[-0.080, +0.006]$, $p = 0.11$, and it rewrites the prefix every turn. The pointer-only note (L4) improves on L1 by $+0.073$ $[+0.031, +0.120]$, $p = 0.001$, which is 42 percent of the L1-to-L3 gap over all traps [21, 60] and 26 percent on external traps, while keeping L1’s cached share exactly (ratio 1.00). L4 remains far from L3 (-0.104 , $p < 10^{-4}$). The note is also the layout whose gap to L1 shrinks with length (slope -0.0065 per call, $p = 0.012$): it helps most in short episodes and least at thirty calls, where the accumulated notes are longest.

What the wrong answers are. The hand audit (Appendix H) coded 40 wrong trap answers, including every wrong L3 answer: 34 are a superseded value read back, almost always the agent’s own earlier write with the overtaking listing row ignored; 3 are threshold questions where the fresh quantity equals the threshold, so a stale read and a strict-inequality misreading coincide; 2, both under L3, are questions whose scripted listing call the agent had skipped, so the fresh value never entered its context; 1 is a derived number. Reveal steps were executed 99.0 to 100 percent of the time under every layout, and restricting to fully executed threads moves no layout’s trap accuracy by more than half a point (Table 8). The repeat battery finds a cell’s trap outcome flipping between identical temperature-0 runs in 11 percent of cells (within-cell standard deviation of question accuracy 0.017); the phrasing battery finds -0.007 $[-0.042, +0.032]$ between the two phrasings (Appendix G).

The bill reverses between per episode and per correct episode. L1 is the cheapest layout per episode at 1.05 cents [0.97, 1.14]; L3 costs 1.16 cents [1.06, 1.26], a paired difference of +0.10 cents [0.06, 0.15], $p < 10^{-4}$; L2 costs 1.35 and L4 1.60. Per correct episode the order is L3 1.22 cents, L1 1.52, L2 1.85, L4 2.07 (Table 1, Figure 3). L1 and L3 are the two points on the Pareto frontier; L2 and L4 are dominated. Repricing every cached token at multiples of the input price from 0 to 1 (Table 10, Figure 5 in Appendix C) never changes the ranking:

THE SAME INFORMATION, FOUR LAYOUTS: ONLY EXTERNAL-UPDATE TRAPS SEPARATE THEM

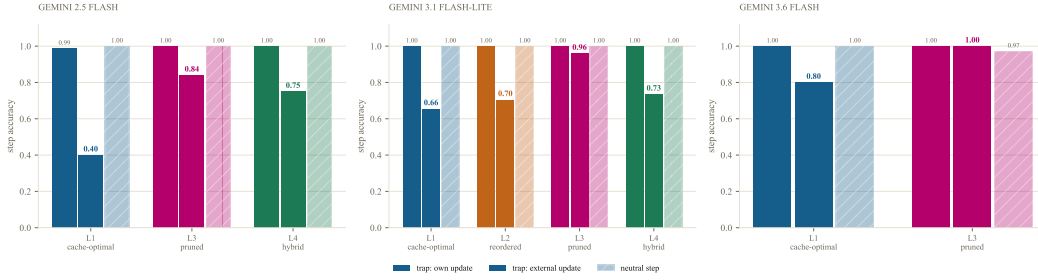


Figure 2: Step accuracy by layout on own-update traps, external-update traps and neutral steps, per model. Only the external-update bars move.

THE FRONTIER: OUTLINED POINTS ARE PARETO-OPTIMAL

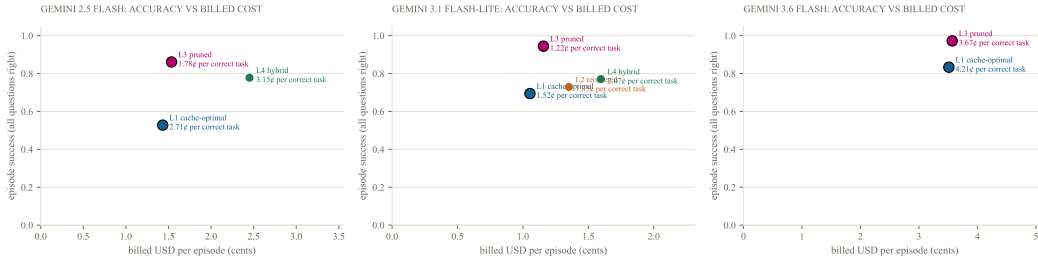


Figure 3: Episode success against billed cents per episode, per model; outlined points are Pareto optimal; cents per correct episode annotated.

L3 is the cheapest per correct episode at every multiplier. There is no cache-read price on this provider at which the cache-optimal layout pays for its errors. The modelled cost at list prices reproduces the billed cost to the fourth decimal, which confirms that the provider bills exactly what it reports.

Why: the history is rarely cached under any layout. Figure 4 (Appendix C) shows the cached share of prompt tokens by turn. Over an episode the share is 0.71 under L1 and 0.68 under L3 (paired +0.032 [0.019, 0.046]); under L2 it is 0.62 and under L4 0.71. Most of that share is the 5,100-token static prefix. Counting only the history, the cached share is 0.24 under L1, 0.13 under L2, 0.15 under L3 and 0.45 under L4 (Table 1). Google’s implicit cache serves the growing history on a minority of turns even when the bytes are unchanged, so the layout that caching rewards is rewarded for its history by nine percentage points over L3, and L3’s shorter prompts (118k against 122k tokens per episode) recover most of that. Two further facts shape the bill. No-tool question turns are served from cache far less often than tool turns (hit rate 0.51 against 0.93 under L1; 0.15 under L2; 0.37 under L3; 0.69 under L4), because disabling tool choice changes the request. And the L4 note, which must stay in the history to keep the prefix stable, raises prompt tokens per episode by 69 percent over L1 and the bill by 51 percent: the quadratic cost of a note repeated on every turn is what puts L4 off the frontier, not a cache loss.

No evidence that a cache hit changes an answer. The on/off control could not run (Section 3). In its place, the best-effort cache itself provides a comparison at identical bytes: the same question, same layout, served from cache in some runs of the repeat battery and not in others. Paired within question, the difference in trap accuracy is 0.000 over the 11 questions that saw both outcomes ($p = 1$). The unpaired comparison over the main grid is confounded, because cache hits concentrate in the later turns of long episodes where traps are harder; it shows -0.03 ($p = 0.56$) under L1 and -0.12 ($p = 0.001$) under L4 before stratification and -0.004 and -0.063 after stratifying by length and chain. We read this as

no evidence that a cache hit changes an answer, with the caveat that the paired sample is small.

4.1 Three Gemini generations

Gemini 2.5 Flash (older generation, thinking on, 72 tasks under L1, L3 and L4) and Gemini 3.6 Flash (newest generation, 36 tasks under L1 and L3) bracket the main model (Table 1, Figure 2). External-trap accuracy under L1 rises with generation: 0.40 on 2.5 Flash (100 questions), 0.655 on 3.1 Flash-Lite (200) and 0.80 on 3.6 Flash (40); under L3 it is 0.84, 0.96 and 1.00. Own-update traps and neutral steps stay at or near 1.00 everywhere (one own-update miss on 2.5 Flash under L1, one neutral miss on 3.6 Flash under L3). The paired L1 minus L3 trap gap is -0.273 $[-0.367, -0.186]$ on 2.5 Flash, -0.177 on 3.1 Flash-Lite and -0.111 $[-0.213, -0.028]$ on 3.6 Flash ($p = 0.031$ on 36 tasks). Permuting model labels over the task-level gaps gives a between-model variance with $p = 0.035$ over all tasks each model ran and $p = 0.092$ on the 36 tasks common to all three (gaps $-0.296, -0.176, -0.111$ there). The note does more for the older model: on 2.5 Flash it recovers 79 percent of the gap [58, 97], raising external-trap accuracy from 0.40 to 0.75, against 42 percent on 3.1 Flash-Lite; it is still dominated on the frontier (cost per episode +71 percent over L1). The cost ordering is the same on every rung: per episode L1 is cheapest or tied (2.5 Flash 1.43 against 1.54 cents, $p = 0.021$; 3.6 Flash 3.51 against 3.57, $p = 0.20$), per correct episode L3 is cheapest (1.78 against 2.71 cents; 3.67 against 4.21), and L3 remains cheapest at every counterfactual cache-read multiplier on all three models (Table 10). On 3.6 Flash the implicit cache serves the history essentially never (history cached share 0.006 under L1, 0.000 under L3; the overall share of 0.47 is the prefix alone), which is why its two layouts cost the same.

4.2 The external arm

BFCL v3 multi-turn (Appendix G and the released logs) was run through the official harness with a per-call message hook on Gemini 3.1 Flash-Lite, 25 base and 25 long-context tasks, under L1, L3 (age-based observation masking, since BFCL has no schema oracle) and L4. The result is a null: accuracy 0.44 / 0.44 (L1), 0.44 / 0.48 (L3), 0.44 / 0.40 (L4) on base / long-context; no layout moves more than three of fifty tasks; cached share 0.78 to 0.80 under all three; cents per correct task 2.10, 1.82 and 2.19. The failures there are state mismatches and twenty-step loops, that is capability failures rather than stale reads, so BFCL neither confirms nor contradicts the mechanism.

4.3 Pre-registered directions, scored

- D1, stale-retention cost: confirmed in two parts of three. L1 is below L3 on trap steps (-0.177 , $p < 10^{-4}$) and there is no gap on neutral steps (exactly zero). The gap does not grow with length (slope $+0.0006$ per call, $p = 0.85$); that part is not supported.
- D2, order-only effect is small, cost is large: half confirmed. L2 does not beat L1 (-0.036 , $p = 0.11$) and is dominated on the frontier, as predicted. Its cached share falls to 0.62, not below 0.20, and its cost per episode rises by 28 percent, not by three times; the magnitude predictions assumed a cache that serves the history, and this provider's does not.
- D3, the hybrid dominates: not supported. L4 recovers 42 percent of the gap [21, 60] over all traps and 26 percent on external traps, below the predicted 50; it keeps 100 percent of L1's cached share, as predicted; it is dominated on the frontier, the opposite of the prediction, because the persistent note raises prompt tokens by 69 percent.
- D4, model dependence: confirmed in direction, with weak evidence for the interaction. The L1-to-L3 gap shrinks monotonically with model generation ($-0.27, -0.18, -0.11$; label permutation $p = 0.035$ over all tasks, 0.092 on the 36 common tasks). The clause that the cost is largest for the cheapest model is not supported: the cheapest model by list price, 3.1 Flash-Lite, sits in the middle.

- D5, price threshold: reversed. L3 is the cheapest layout per correct episode at every counterfactual cache-read price from 0 to 1 times the input price. No threshold exists on this provider; the prediction assumed the cache discipline buys a large history discount, and the measured history cached share under L1 is 0.24.

5 Limitations, narrated

One provider family. The plan froze three families and the workspace key allowed one, so every number here is a Google number and the ladder is three Gemini generations, not three vendors. The cost conclusion depends on Google’s implicit cache rarely serving the history; a provider with explicit breakpoints and a documented 90 percent intra-turn hit rate (Liu et al., 2026a) would give the append-only layout a larger cost advantage, and the D5 threshold could then exist. The cache on/off control assigned to Anthropic could not run; its substitute, the same question served from cache or not at identical bytes, is paired within question but rests on 11 questions.

An oracle pruner. L3 redacts exactly the values that a later result restates, using the executor’s schema annotations. No deployed pruner has that oracle. L3 is therefore an upper bound on what content-only intervention can achieve, not a recommendation to ship. The right reading of the L1-to-L3 gap is the size of the prize, and the right reading of L4 is how much of that prize a pointer-only note can collect without an oracle: about a quarter on external traps.

A synthetic testbed. StaleTrap episodes are scripted, the domains are small, the distractors are designed, and the questions have one exact answer. This is what makes the stale annotation possible, and it is also why the absolute accuracies should not be read as deployment numbers. The external arm on BFCL v3 multi-turn (Appendix G) found no layout effect at all on 50 tasks, which we read as a property of that benchmark (its failures are capability failures, not stale reads) rather than as evidence against the mechanism; it also means we have not shown the mechanism on a public benchmark.

Temperature-0 noise and phrasing. The repeat battery shows that a trap-accuracy outcome flips between identical runs in about one cell in nine, so individual cells carry noise that the paired tests over 144 tasks absorb and single-cell readings do not. The phrasing battery (Appendix G) covers one alternative phrasing of every instruction and question, not free-form user language.

The audit is not blind. The forty wrong answers were coded by the agent that ran the study from the observation history of each question, with the layout visible. The codes are mechanical (does the answer equal an earlier observation of the same field) and ship with the release, but a blind second coder would be better.

What the action-fidelity check does and does not rule out. Agents occasionally skip a scripted listing call (about one percent of reveal steps), and a skipped reveal makes the question unanswerable. Restricting to fully executed threads moves every accuracy by less than a point, so the layout gap is not an artefact of layouts changing tool-calling behaviour. It does not rule out subtler effects of layout on the agent’s attention that we have not measured.

6 Related work

Position and staleness. Accuracy is U-shaped in the position of the relevant passage (Liu et al., 2023a), an attention bias that calibration or training can reduce (Hsieh et al., 2024; An et al., 2024); as inputs grow, primacy fades and recency stays (Veseli et al., 2025); repeated keys are retrieved at their first and last occurrence (Bajaj et al., 2025), and the latest of several revisions less reliably as revisions accumulate (Qiao et al., 2026). Related distractors hurt most (Cuconasu et al., 2024); placing top-ranked passages at the extremes helps in

retrieval (Jin et al., 2024), but reordering is no better than shuffling in realistic retrieval (Cuconasu et al., 2025). Repeating the instruction near the evidence helps (Agrawal et al., 2024; Liu et al., 2023b; Pal et al., 2023); multi-turn conversation anchors on early turns and on the model’s own outputs (Laban et al., 2025). In agents, tool-list position biases tool choice (Faghih et al., 2025), observations go stale (Liu et al., 2026b), masking old observations helps mid-capacity models only (Zhang et al., 2026), restating goal and state every step helps (Rozanov and Rei, 2024; Luo et al., 2026), and time stamps alone do not tell a model which version is current (Hu et al., 2026).

Prompt caching and its discipline. Exact-prefix reuse is the production default (Zheng et al., 2023; Gim et al., 2023). Reusing KV entries at shifted positions degrades outputs, and systems work recomputes, re-embeds or fine-tunes to recover it (Yao et al., 2024; Hu et al., 2024; Ma et al., 2024; Yang et al., 2025b;a; Agarwal et al., 2025; Bai and Eisner, 2025); agent serving work edits cached content in place (Ma et al., 2026a;b) and shows request-history dependence under selective reuse (Gu et al., 2026). On the API side, caching strategies have been costed (Lumer et al., 2026), compaction shown to break caches (Song, 2026; Xu et al., 2026a;b), production hit rates measured near 90 percent within a turn and 55 across turns (Liu et al., 2026a), warm-cache economics modelled (Khailo, 2026), tool routing reshaped to keep the prefix fixed (Zha et al., 2026), and cache-aware ordering studied for retrieved evidence (Tan et al., 2026; He, 2026) but not for an agent’s own history. Prefix caching alone can change open-source agent trajectories (Patodiya, 2026); it is also observable from outside through timing, which is how its presence at API providers was first audited (Gu et al., 2025). Serving systems schedule multi-agent workflows to keep prefixes warm (Pan et al., 2025) and cache whole plans at test time (Zhang et al., 2025); KV compression has its own accuracy pitfalls on reasoning-heavy inputs (Chen et al., 2025).

Context management and cost-aware evaluation. Pruning the history to the last few tool pairs raises completion and cuts tokens (Lodha et al., 2026); observation masking matches LLM summaries at a fraction of the cost (Lindenbauer et al., 2025; Chintalapati et al., 2026); dropping content hides its cost in extra calls (Liu, 2026); recurrent compression weakens recency (Min et al., 2026); whether a past interaction is safe to replace is a learned judgement (Wang et al., 2026; Nguyen et al., 2026), and what an agent keeps in memory changes what it does next (Xiong et al., 2025). Cost-controlled evaluation asks for a frontier of accuracy against dollars (Kapoor et al., 2024; Erol et al., 2025; Kapoor et al., 2025), shows harness choices move cost by an order of magnitude (Ali et al., 2026; Vats and Golev, 2026; Liu and Han, 2026), and shows tool-output compression can raise a bill once cache behaviour is counted (Weinberger and Hozes, 2026). None isolates layout. Surveys frame the field (Hua et al., 2025; Mei et al., 2025); practitioner guides prescribe the layouts we test (Ji, 2025; Anthropic, 2025; Bouchard, 2026).

7 Conclusion

The cache-optimal layout is free when the agent’s own tool replies carry the current value and costs about a third of the answers when the world changes behind the agent and the new value arrives inside a listing. That cost is paid at six tool calls as surely as at thirty, it is not reduced by reordering, and a pointer-only note recovers about a quarter of it at a price in tokens that puts the note off the cost frontier. Redacting the stale value in place removes the cost almost entirely, and on the provider we could test it is cheaper per correct episode than the cache-optimal layout at every cache-read price, because the history is rarely served from cache under any layout. Across three Gemini generations the gap shrinks from 27 to 11 points and does not close, and the cost ordering does not change. The practical reading is narrow and usable: keep the prefix, but when a later observation overtakes an earlier value for the same field, overwrite the earlier value even if that breaks the cache; the accuracy is worth more than the discount. The open question is the one our key could not reach: on a provider whose explicit cache serves the history nine turns in ten, the discount is larger and the threshold the plan predicted may exist. The testbed, logs and analysis are released so that the question can be answered there with the same episodes.

Reproducibility statement

The frozen testbed with its SHA-256 manifest, the environment, the four layout serialisers, the runner, every raw episode log with per-call provider usage and billed cost, the single analysis script that produces every number in this paper, the audited sample with codes, and the spend log accompany this submission and are released publicly. Total metered spend for the study, including pilots, calibration, smoke tests, discarded runs and the external arm, was USD 18.42 .

Ethics statement

The testbed is synthetic: fictional records, fictional names drawn from a fixed list, no personal data. No deployed system was probed beyond public model APIs. The work measures a cost of an engineering practice; it introduces no new capability.

LLM usage statement

Large language models are the subjects of this study and served as engineering assistants for code, figures and manuscript editing. All design choices, statistical decisions, interpretations and claims are the author’s; every reported number was checked against the scripted analysis output before entering the text.

References

- Shubham Agarwal, Sai Sundaresan, Subrata Mitra, Debabrata Mahapatra, Archit Gupta, Rounak Sharma, Nirmal Joshua Kapu, Tong Yu, et al. Cache-craft: Managing chunk-caches for efficient retrieval-augmented generation. In SIGMOD 2025, 2025. arXiv preprint arXiv:2502.15734.
- Devanshu Agrawal, Shang Gao, and Martin Gajek. Can’t remember details in long documents? you need some r&r. In Findings of EMNLP 2024, 2024. arXiv preprint arXiv:2403.05004.
- Muayad Sayed Ali, Aliaksandra Novik, Anji Boddupally, Artem Yavorskyi, Chris Nickerson, Daniel Rica, Emily DuGranrut, Felix Leung, et al. The harness effect: How orchestration design sets the token economics of enterprise agentic ai. arXiv preprint arXiv:2607.06906, 2026.
- Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, and Jian-Guang Lou. Make your llm fully utilize the context. In NeurIPS 2024, 2024. arXiv preprint arXiv:2404.16811.
- Anthropic. Effective context engineering for ai agents. <https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents>, 2025. Engineering blog, 29 September 2025; accessed 2026-10-10.
- TJ Bai and Jason Eisner. Accelerating language model workflows with prompt choreography. In Transactions of the ACL, 2025. arXiv preprint arXiv:2512.23049.
- Anooshka Bajaj, Deven Mahesh Mistry, Sahaj Singh Maini, Yash Aggarwal, and Zoran Tiganj. Beyond semantics: How temporal biases shape retrieval in transformer and state-space models. arXiv preprint arXiv:2510.22752, 2025.
- Louis Bouchard. Context engineering in 2026: Why we stopped compacting our agent’s context. <https://www.louisbouchard.ai/context-engineering-2026/>, 2026. Blog post, 18 August 2026; accessed 2026-10-10.
- Alex Chen, Renato Geh, Aditya Grover, Guy Van den Broeck, and Daniel Israel. The pitfalls of kv cache compression. arXiv preprint arXiv:2510.00231, 2025.

- Renuka Chintalapati, Sid Raskar, Anurag Acharya, Jared Willard, Patrick Emami, and Sameera Horawalavithana. Evaluating memory condensation strategies for coding agents in data-driven scientific discovery. arXiv preprint arXiv:2605.18854, 2026.
- Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. The power of noise: Redefining retrieval for rag systems. In SIGIR 2024, 2024. arXiv preprint arXiv:2401.14887.
- Florin Cuconasu, Simone Filice, Guy Horowitz, Yoelle Maarek, and Fabrizio Silvestri. Do rag systems really suffer from positional bias? In EMNLP 2025, 2025. arXiv preprint arXiv:2505.15561.
- Mehmet Hamza Erol, Batu El, Mirac Suzgun, Mert Yuksekgonul, and James Zou. Cost-of-pass: An economic framework for evaluating language models. In ICLR 2026, 2025. arXiv preprint arXiv:2504.13359.
- Kazem Faghieh, Wenxiao Wang, Yize Cheng, Siddhant Bharti, Gaurang Sriraman, Sriram Balasubramanian, Parsa Hosseini, and Soheil Feizi. Tool preferences in agentic llms are unreliable. arXiv preprint arXiv:2505.18135, 2025.
- In Gim, Guojun Chen, Seung seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. In MLSys 2024, 2023. arXiv preprint arXiv:2311.04934.
- Chenchen Gu, Xiang Lisa Li, Rohith Kuditipudi, Percy Liang, and Tatsunori Hashimoto. Auditing prompt caching in language model apis. arXiv preprint arXiv:2502.07776, 2025.
- Tiffany Gu, Annie Guan, Manshu Huang, Nitin Rao, Siddhant Shah, and Margaret Capetz. Request order matters: Cache-history sensitivity in selective kv-cache reuse for rolling agents. arXiv preprint arXiv:2610.05833, 2026.
- Rong He. Prefix sharing is a sorting problem. arXiv preprint arXiv:2609.13692, 2026.
- Cheng-Yu Hsieh, Yung-Sung Chuang, Chun-Liang Li, Zifeng Wang, Long T. Le, Abhishek Kumar, James Glass, Alexander Ratner, et al. Found in the middle: Calibrating positional attention bias improves long context utilization. In Findings of ACL 2024, 2024. arXiv preprint arXiv:2406.16008.
- Chuanrui Hu, Tong Li, Xingze Gao, Hongda Chen, Yi Bai, Dannong Xu, Tianwei Lin, Xiaohong Li, et al. Evaluating long-horizon memory for multi-party collaborative dialogues. arXiv preprint arXiv:2602.01313, 2026.
- Junhao Hu, Wenrui Huang, Weidong Wang, Haoyi Wang, Tiancheng Hu, Qin Zhang, Hao Feng, Xusheng Chen, et al. Epic: Efficient position-independent caching for serving large language models. In ICML 2025, 2024. arXiv preprint arXiv:2410.15332.
- Qishuo Hua, Lyumanshan Ye, Dayuan Fu, Yang Xiao, Xiaojie Cai, Yunze Wu, Jifan Lin, Junfei Wang, et al. Context engineering 2.0: The context of context engineering. arXiv preprint arXiv:2510.26493, 2025.
- Yichao Ji. Context engineering for ai agents: Lessons from building manus. <https://manus.im/blog/Context-Engineering-for-AI-Agents-Lessons-from-Building-Manus>, 2025. Blog post, 18 July 2025; accessed 2026-10-10.
- Bowen Jin, Jinsung Yoon, Jiawei Han, and Serkan O. Arik. Long-context llms meet rag: Overcoming challenges for long inputs in rag. arXiv preprint arXiv:2410.05983, 2024.
- Sayash Kapoor, Benedikt Stroebel, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. Ai agents that matter. In TMLR 2025, 2024. arXiv preprint arXiv:2407.01502.
- Sayash Kapoor, Benedikt Stroebel, Peter Kirgis, Nitya Nadgir, Zachary S Siegel, Boyi Wei, Tianci Xue, Ziru Chen, et al. Holistic agent leaderboard: The missing infrastructure for ai agent evaluation. In ICLR 2026, 2025. arXiv preprint arXiv:2510.11977.

- Maxim Khailo. Keeping the cache warm pays: Keepalive economics for agentic workloads. arXiv preprint arXiv:2607.19214, 2026.
- Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. Llms get lost in multi-turn conversation. arXiv preprint arXiv:2505.06120, 2025.
- Tobias Lindenbauer, Igor Slinko, Ludwig Felder, Egor Bogomolov, and Yaroslav Zharov. The complexity trap: Simple observation masking is as efficient as llm summarization for agent context management. In DL4C Workshop at NeurIPS 2025, 2025. arXiv preprint arXiv:2508.21433.
- Banruo Liu, Haoran Qiu, Íñigo Goiri, Rodrigo Fonseca, Ricardo Bianchini, and Esha Choukse. Agentic coding in the wild: Characterizing github copilot traces at production scale. arXiv preprint arXiv:2608.00101, 2026a.
- 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. In Transactions of the ACL, 2023a. arXiv preprint arXiv:2307.03172.
- Shuyu Liu. What does context compression cost an agent? interaction costs unrevealed by task-completion metrics. arXiv preprint arXiv:2608.16370, 2026.
- Yangze Liu and Zhongyi Han. What does a harness buy? tokens, mostly. arXiv preprint arXiv:2610.04433, 2026.
- Yijin Liu, Xianfeng Zeng, Fandong Meng, and Jie Zhou. Instruction position matters in sequence generation with large language models. In Findings of ACL 2024, 2023b. arXiv preprint arXiv:2308.12097.
- Yingying Liu, Junzhou Fang, and Chenxiong Qian. When agent context goes stale: Incoherence in volatile agent context. arXiv preprint arXiv:2610.05281, 2026b.
- Abhilasha Lodha, Mahsa Pahlavikah Varnosfaderani, Abir Chakraborty, and Abhinav Mithal. Less context, better agents: Efficient context engineering for long-horizon tool-using llm agents. arXiv preprint arXiv:2606.10209, 2026.
- Elias Lumer, Faheem Nizar, Akshaya Jangiti, Kevin Frank, Anmol Gulati, Mandar Phadate, and Vamse Kumar Subbiah. Don't break the cache: An evaluation of prompt caching for long-horizon agentic tasks. arXiv preprint arXiv:2601.06007, 2026.
- Yu Luo, Jiamin Jiang, Yimin Zuo, Xidao Wen, Rongchen Gao, Yongqian Sun, Shenglin Zhang, Guiyang Liu, et al. Beyond memory: Harnessing long-horizon agents with explicit belief states. arXiv preprint arXiv:2610.01415, 2026.
- Bole Ma, Jan Eitzinger, and Harald Koestler. Leyline: Kv cache directives for agentic inference. arXiv preprint arXiv:2606.01065, 2026a.
- Bole Ma, Jan Eitzinger, and Harald Köstler. Irmisul: Mla-native position-independent caching for agentic llm serving. arXiv preprint arXiv:2605.05696, 2026b.
- Dongyang Ma, Yan Wang, and Lan Tian. Block-attention for efficient prefilling. In ICLR 2025, 2024. arXiv preprint arXiv:2409.15355.
- Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, et al. A survey of context engineering for large language models. arXiv preprint arXiv:2507.13334, 2025.
- Guanghui Min, Liang Wu, Mayank Darbari, Chen Chen, and Liangjie Hong. Toward reliable context compression for long-horizon agents: An empirical study of execution instability. arXiv preprint arXiv:2608.06503, 2026.
- Trang Nguyen, Eulrang Cho, Bingqing Chen, and Tim Dettmers. Cliffcompaction: Cost-efficient compaction for long-horizon coding agents. arXiv preprint arXiv:2609.26779, 2026.

- Arka Pal, Deep Karkhanis, Manley Roberts, Samuel Dooley, Arvind Sundararajan, and Siddhartha Naidu. Giraffe: Adventures in expanding context lengths in llms. arXiv preprint arXiv:2308.10882, 2023.
- Zaifeng Pan, Ajjkumar Patel, Zhengding Hu, Yipeng Shen, Yue Guan, Wan-Lu Li, Lianhui Qin, Yida Wang, et al. Kvflow: Efficient prefix caching for accelerating llm-based multi-agent workflows. arXiv preprint arXiv:2507.07400, 2025.
- Aditi Patodiya. Same request, different answer: Quantization amplifies cache-induced divergence in llm serving. arXiv preprint arXiv:2609.04748, 2026.
- Boyu Qiao, Sean Guo, Xian Yang, Kun Li, Wei Zhou, Songlin Hu, and Yunya Song. Diagnosing retrieval bias under multiple in-context knowledge updates in large language models. arXiv preprint arXiv:2603.12271, 2026.
- Nikolai Rozanov and Marek Rei. Stateact: Enhancing llm base agents via self-prompting and state-tracking. arXiv preprint arXiv:2410.02810, 2024.
- Yan Song. Cache-aware prompt compression: a two-tier cost model for llm api caching. arXiv preprint arXiv:2607.15516, 2026.
- Kaizhen Tan, Rong Gu, and Mingyuan Li. Cacheweaver: Cache-aware evidence ordering for efficient grounded rag inference. arXiv preprint arXiv:2606.19667, 2026.
- Naman Vats and Oleg Golev. The scaffold effect in coding agents: Harness choice as a hidden variable in coding-agent evaluation. arXiv preprint arXiv:2607.22585, 2026.
- Blerta Veseli, Julian Chibane, Mariya Toneva, and Alexander Koller. Positional biases shift as inputs approach context window limits. In COLM 2025, 2025. arXiv preprint arXiv:2508.07479.
- Mingxuan Wang, Hongyue Chen, Yinglong Guo, Fei Luo, Chao Ning, Bo Wang, Guorun Yao, Yanbiao Ma, et al. Statecomp: Learning when to compress history in long horizon agents. arXiv preprint arXiv:2609.27298, 2026.
- Sarel Weinberger and Amir Hoziez. Token reduction is not cost reduction. arXiv preprint arXiv:2607.12161, 2026.
- Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How memory management impacts llm agents: An empirical study of experience-following behavior. arXiv preprint arXiv:2505.16067, 2025.
- Buqiang Xu, Zirui Xue, Dianmou Chen, Chenyang Fu, Chiyu Wu, Caiying Huang, Chen Jiang, Jizhan Fang, et al. Tokenpilot: Cache-efficient context management for llm agents. In Findings of EMNLP 2026, 2026a. arXiv preprint arXiv:2606.17016.
- Peng Xu, Zuyu Zhang, Yuze Sun, Feng Tian, Long Wang, and Chen Zhang. Contextpipe: Database-inspired context assembly for long-horizon agents. arXiv preprint arXiv:2609.00749, 2026b.
- Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. Kvlink: Accelerating large language models via efficient kv cache reuse. In NeurIPS 2025, 2025a. arXiv preprint arXiv:2502.16002.
- Xinyu Yang, Tianqi Chen, and Beidi Chen. Ape: Faster and longer context-augmented generation via adaptive parallel encoding. In ICLR 2025, 2025b. arXiv preprint arXiv:2502.05431.
- Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, et al. Cacheblend: Fast large language model serving for rag with cached knowledge fusion. In EuroSys 2025, 2024. arXiv preprint arXiv:2405.16444.
- Donghui Zha, Lingwei Xu, Linxiao Wu, Yixue Dong, and Haochen Li. Cacherouter: A dual-path tool routing architecture with cache-preserving main-model isolation for long-tail tool discovery. arXiv preprint arXiv:2608.22708, 2026.

Haoliang Zhang, Qixin Xu, Zhuofeng Li, Lei Zhang, Pengcheng Jiang, Yu Zhang, and Julian McAuley. Masking stale observations helps search agents – until it doesn’t: A regime map and its mechanism. arXiv preprint arXiv:2606.00408, 2026.

Qizheng Zhang, Michael Wornow, Gerry Wan, and Kunle Olukotun. Agentic plan caching: Test-time memory for fast and cost-efficient llm agents. arXiv preprint arXiv:2506.14852, 2025.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, et al. Sglang: Efficient execution of structured language model programs. In NeurIPS 2024, 2023. arXiv preprint arXiv:2312.07104.

Appendix

A Pre-registered directions, verbatim

The project log (MILESTONES.md, committed in the public repository) records the following five directional predictions, each dated 2026-10-10, before any model call:

1. D1 (stale-retention cost). L1 accuracy is below L3 on stale-trap steps, the gap grows with trajectory length, and there is no comparable gap on neutral steps.
2. D2 (order-only effect is small, cost is large). L2 does not significantly beat L1, while its cache-read fraction falls below 20 percent and its USD per task rises by at least 3x on 0.1x-read pricing; L2 is dominated on the frontier.
3. D3 (the hybrid dominates). L4 recovers at least 50 percent of the L1-to-L3 accuracy gap while keeping at least 80 percent of L1’s cache-read fraction, and sits on or inside the frontier for every model.
4. D4 (model dependence). The L1-vs-L3 gap shrinks with model capability; the confusion cost is largest for the cheapest models, which is where caching saves the least absolute money.
5. D5 (price threshold). Per correct task, L3 is cheapest when cache reads are expensive and L1/L4 become cheapest below a cache-read price threshold; the threshold is reported per model and compared with the published break-evens.

This is a project-log pre-registration, not a registry filing. Three design changes were made during calibration and before the testbed was frozen, all logged with their reason: the prefix was enlarged to clear the 4,096-token minimum of Gemini 3.x on Google’s tokenizer; a 30-call length and a wider update-to-question distance were added after a first pilot scored 12 of 12 trap steps; and the external-update trap kind was added after a second calibration scored 10 of 10 on own-update traps at 18 and 30 calls. Every calibration episode was discarded.

B Novelty delineation table

C Additional figures

D Verbatim prompts and layouts

D.1 System prompt (verbatim, byte-identical across layouts)

You are an operations assistant for Halvard Logistics, a mid-sized distribution company. You work inside an internal console that exposes three back-office systems as tools: the warehouse inventory system, the shared meeting calendar, and the customer-support ticketing system. The person you are talking to is an operations coordinator who gives you one instruction at a time during

How a session works

- The coordinator sends short instructions one at a time. Most instructions ask you to look something up or to make a change in o

Closest neighbour	What it established / what it does not cover
Token Reduction Is Not Cost Reduction (Weinberger and Hozes, 2026) Don't Break the Cache (Lumer et al., 2026)	Billed cost split into cache creation and reads, with task success, on one coding agent. Varies tool-output compression (content changes); no order-only or recitation arm; one family. Caching strategies on three providers, 500+ sessions: 41 to 80 percent cost savings. Accuracy is not an outcome; no layout manipulation.
What Does a Harness Buy? (Liu and Han, 2026)	Five models by three harnesses: cost per task at one price list; cached pricing scales the bill but does not reorder harnesses. Whole harnesses differ; no content-fixed arm.
TokenPilot (Xu et al., 2026a); Cache-Aware Prompt Compression (Song, 2026)	Name the text-sparsity versus cache-continuity trade-off; compaction methods with hit rates. Order and pruning confounded; accuracy reported as "competitive" or on 50 tasks; no stale-step annotation.
CacheWeaver (Tan et al., 2026); Prefix Sharing Is a Sorting Problem (He, 2026)	Cache-aware ordering of evidence or shared prefixes with unchanged quality. Single-turn retrieval or serving; no agent history, no pricing.
Do RAG Systems Really Suffer From Positional Bias? (Cuconasu et al., 2025)	Reordering no better than shuffling in realistic retrieval. No history, no caching; it is our D2 null.
Less Context, Better Agents (Lodha et al., 2026); Complexity Trap (Lindenbauer et al., 2025); Masking Stale Observations (Zhang et al., 2026); condenser study (Chintalapati et al., 2026)	Pruning or masking versus full history, with accuracy. Costs at list prices with no cache term; no order-only arm.
When Agent Context Goes Stale (Liu et al., 2026b); multi-revision retrieval (Qiao et al., 2026)	Agents answer from overtaken observations; latest-value accuracy falls with revisions. Runtime fix or single call; no layout factorial, no cost.
Leyline, Irmisul, Request Order Matters (Ma et al., 2026a;b; Gu et al., 2026)	Serving-side fixes for non-prefix KV reuse. Do not evaluate the API-level discipline that practitioners follow.
StateAct, R&R, belief-state block (Rozanov and Rei, 2024; Agrawal et al., 2024; Luo et al., 2026)	Restating the goal and the current state near the question helps. Not tested as a cache-preserving layout against pruning; no cost.

Table 3: Novelty delineation: the nearest work and what each does not cover. Full related work in Section 6.

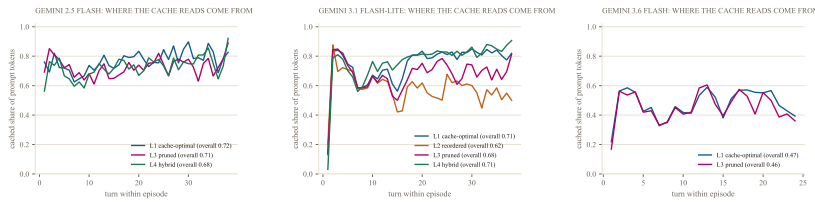


Figure 4: Cached-token share of the prompt by turn index, per layout, Gemini 3.1 Flash-Lite. The static prefix is cached from turn two under every layout; the history is cached on a minority of turns.

- Some instructions are questions that must be answered from the information already present in this conversation. These questions are answered by the agent.
- Tool results are returned to you as JSON. They are authoritative at the moment they were produced. A later tool result about the same thing may differ.
- Identifiers look like SKU-4471 (inventory items), EVT-2093 (calendar events), and TCK-5810 (support tickets). Identifiers that differ by a single digit refer to different records. People are referred to by full name. Times are in HH:MM format. Prices are in US dollars.

Inventory system

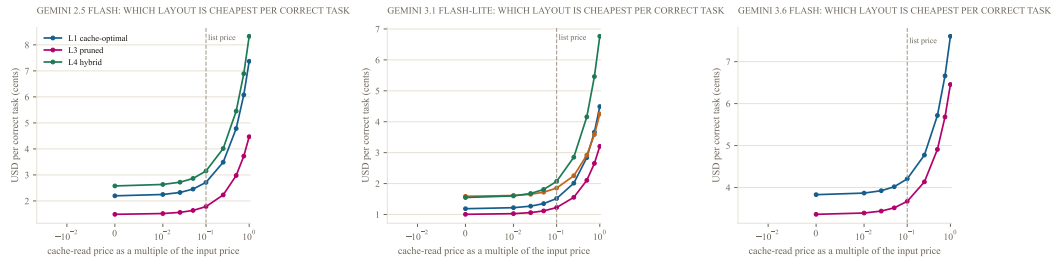


Figure 5: Cents per correct episode under counterfactual cache-read prices from free ($0\times$) to the full input price ($1\times$). The billed point is $0.1\times$. L3 is cheapest throughout.

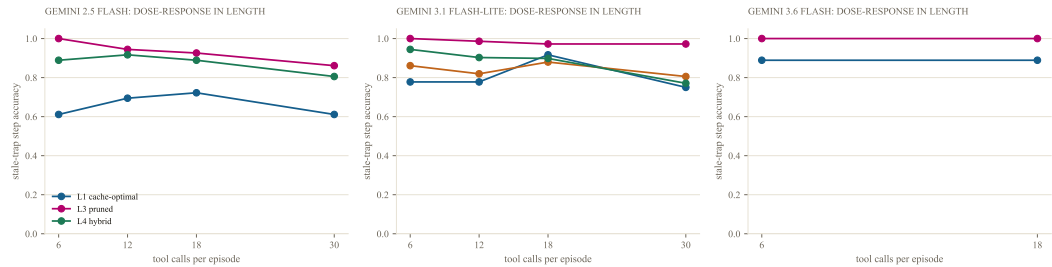


Figure 6: Trap accuracy by episode length and by distractor level, per layout and model. Under L1 the 18-call cell (0.92) stands out against 0.78, 0.78 and 0.75 at 6, 12 and 30 calls; on external traps alone L1 scores 0.60, 0.60, 0.85 and 0.55. We have no mechanism for the 18-call cell; the length interaction is null and the repeat battery shows that single cells carry noise.

- Each SKU has an on-hand quantity, a storage bin, a unit sale price, a reorder threshold, and a primary supplier. `receive_shipment` updates on-hand quantity and report the quantity that is now on hand. `set_price`, `move_stock`, and `set_reorder_threshold` change the corresponding values. `only` cycle-count aid.
- A SKU is considered below its reorder threshold when its on-hand quantity is less than or equal to the threshold.

Calendar system

- Each event has a title, a date, a start and end time, a room, an attendee list, and a status (confirmed or cancelled). `reschedule_event` only.
- When asked how many attendees an event has, count the names on the most recent attendee list you have seen for that event.

Ticketing system

- Each ticket has a subject, a priority from P1 (most urgent) to P4 (least urgent), an assignee, a status (open, pending, in_progress), and a due date. `only`.

Style

- When carrying out an instruction, call the tool directly; a short confirmation afterwards is fine but not required.
- When answering a no-tool question, output only the JSON object.
- Never fabricate tool results. If a tool returns an error, report the error briefly and wait for the next instruction.
- Keep every reply brief. The coordinator is working through a long list and values speed and precision over commentary.

Reference notes (read once; they do not change during a session)

- Answer formats for no-tool questions. Quantities, thresholds, counts and hours: a bare integer, for example `{"answer": 27}`. Prices: `{"answer": "04"}`, `{"answer": "Cedar"}`, `{"answer": "in_progress"}`, `{"answer": "P2"}`, `{"answer": "Lena Fischer"}`. Times: HH:MM in 24-hour format, for example `{"answer": "14:30"}`. Yes/no questions: `{"answer": "yes"}` or `{"answer": "no"}`.
- Deciding what is current. Scan every tool result in the conversation that mentions the record named in the question. The result is the most recent one.
- Error handling. The backend returns `{"error": "..."}` when an identifier is unknown, an argument is malformed, or stock is insufficient.
- Priorities and SLAs. P1 is the most urgent and P4 the least. The response SLA is returned by `get_sla_policy` for a given priority.
- Units and formats. Quantities and units are whole numbers. Prices are decimal dollars. Times are HH:MM office local time; dates are MM-DD. Room names are single capitalised words. Bin codes are a letter, a hyphen and two digits. Agent and attendee names are single capitalised words.
- Scope. Only the three systems above are available. Requests outside them (email, HR, finance) should be declined in one sentence.
- Confirmations. After a successful change, one short sentence such as "Done: SKU-4471 now has 27 units on hand." is acceptable. After a successful lookup, repeat only the fields that were requested. Never repeat the entire record.

Messaging and HR tools

- The console also exposes internal messaging (`send_message`, `post_channel_update`, `schedule_message`, `cancel_scheduled_message`) and service tools (`get_employee`, `get_time_off_balance`, `request_time_off`, `submit_expense`, `post_job_opening`). They are available in the console.

Standard procedures

- Stock receipt: when a delivery is logged with `receive_shipment`, the on-hand quantity rises by the received units and the result reports the new on-hand quantity. No separate confirmation call is needed.
 - Order fulfilment: `ship_order` fails with an insufficient-stock error when the requested units exceed the on-hand quantity; report the error and do not partially ship.
 - Price changes: `set_price` takes effect immediately in the storefront. Quote prices with two decimals when the coordinator asks for prices.
 - Relocation: `move_stock` relocates every unit of a SKU; there are no partial moves. `list_bin_contents` reflects the bin's contents.
 - Rescheduling: `reschedule_event` keeps the event on the same date; a move to another date requires cancelling and recreating, with `cancel_event` and `schedule_event`.
 - Attendance: `add_attendee` and `remove_attendee` return the complete attendee list after the change; the number of attendees is in the result.
 - Triage: `assign_ticket` replaces the previous assignee outright. `set_priority` sets an absolute level; `escalate_ticket` raises by exact level.
 - Reporting: when a question asks for a current value, the most recent result for that record and field in this conversation is the answer.
- #### Glossary
- On-hand quantity: units physically present in the main warehouse, excluding inbound shipments not yet received and outbound shipments not yet shipped.
 - Reorder threshold: the on-hand level at or below which purchasing raises an order; it is set per SKU and changes only through `set_reorder_threshold`.
 - Bin: a shelf location code (letter, hyphen, two digits). A SKU occupies exactly one bin at a time.
 - Lead time: supplier-quoted days from purchase order to delivery; informational only.
 - Event status: confirmed events appear on room schedules; cancelled events do not, but their record remains readable with `get_event`.
 - Attendee list: the people invited to an event, in invitation order; the organiser is included as an attendee.
 - Ticket status: open (not yet picked up), pending (waiting on the customer), in progress (being worked), resolved (closed; excluded from the queue).
 - Agent load: the number of unresolved tickets assigned to one agent, as returned by `get_agent_load` at the time of the call.
 - Response SLA: the policy response time in hours for a priority level; it is a property of the priority, not of the ticket.
 - Superseded observation: an earlier tool result whose value for a record and field has been replaced by a later tool result; it must not be used for reasoning.
 - Consistency. Treat the coordinator's instructions as authoritative about what to do and the tool results as authoritative about what is.

D.2 Tool catalogue

The 39 tool declarations (names, descriptions and JSON schemas) ship in `corpus/tools.json`; the names are: `get_stock`, `get_price`, `get_reorder_threshold`, `get_supplier`, `list_bin_contents`, `receive_shipment`, `ship_order`, `set_price`, `move_stock`, `set_reorder_threshold`, `get_event`, `get_room_schedule`, `get_availability`, `reschedule_event`, `move_event_room`, `add_attendee`, `remove_attendee`, `cancel_event`, `set_event_title`, `get_ticket`, `get_agent_load`, `get_sla_policy`, `list_tickets`, `assign_ticket`, `set_priority`, `set_status`, `add_tag`, `add_comment`, `escalate_ticket`, `send_message`, `post_channel_update`, `schedule_message`, `cancel_scheduled_message`, `get_channel_members`, `get_employee`, `get_time_off_balance`, `request_time_off`, `submit_expense`, `post_job_opening`.

D.3 User-turn phrasings

Every action instruction and every question has two phrasings; phrasing 1 is the main grid. Examples (phrasing 1 / phrasing 2):

`get_stock`: Check the current stock level of {sku}. / How many {sku} do we have on hand right now? Look it up.
`receive_shipment`: Record that we received {units} units of {sku}. / A delivery of {units} units of {sku} just arrived. Log the receipt.
`list_tickets`: List the tickets currently in status {status}. / Show me everything that's {status} right now.
`reschedule_event`: Reschedule event {event_id} to {new_start}-{new_end}. / Move {event_id} so it runs {new_start} to {new_end}.

question wrapper 1: Based only on the information already gathered in this conversation, and without calling any tools (tool calls are disabled here): {question}
question wrapper 2: No tool calls for this one, just use what we have already seen in this conversation (tools are disabled here): {question}

`qty`: what is the current on-hand quantity of {record}? (format <number>)
`start`: at what time does event {record} currently start? (format <HH:MM>)
`assignee`: who is currently assigned to ticket {record}? (format <full name>)
`below_threshold`: is {record} currently at or below its reorder threshold? (format yes or no)

D.4 L4 session note (template)

[Session note before step 9]

Goal: <goal sentence>

Rule: for every record use the latest observation listed as current; observations listed as superseded are no longer valid.

Records with updated observations:

- <record>: current <field> at step <k>; superseded: steps <i>, <j>

- <n> other records observed once; those observations are current.

[End of note]

E Statistical test family

The family holds 65 tests; 29 survive Benjamini-Hochberg at $q = 0.05$ (effective threshold $p \leq 0.0207$). Every test below is a two-sided sign-flip permutation over tasks (20,000 flips) unless its name says int (label permutation, 10,000) or battery. Table 5 gives the interaction estimates.

Model	Test	p	BH
Gemini 2.5 Flash	cache_frac:L1-L3	0.581	
Gemini 2.5 Flash	cache_frac:L4-L1	0.030	
Gemini 2.5 Flash	cache_frac:L4-L3	0.112	
Gemini 2.5 Flash	cost:L1-L3	0.021	survives
Gemini 2.5 Flash	cost:L4-L1	<0.0001	survives
Gemini 2.5 Flash	cost:L4-L3	<0.0001	survives
Gemini 2.5 Flash	int:L1-L3:distractor	0.377	
Gemini 2.5 Flash	int:L1-L3:length	0.355	
Gemini 2.5 Flash	int:L1-L3:trap_vs_neutral	<0.0001	survives
Gemini 2.5 Flash	int:L4-L1:distractor	0.196	
Gemini 2.5 Flash	int:L4-L1:length	0.559	
Gemini 2.5 Flash	int:L4-L1:trap_vs_neutral	<0.0001	survives
Gemini 2.5 Flash	neutral_acc:L1-L3	1.000	
Gemini 2.5 Flash	neutral_acc:L4-L1	1.000	
Gemini 2.5 Flash	neutral_acc:L4-L3	1.000	
Gemini 2.5 Flash	success:L1-L3	<0.0001	survives
Gemini 2.5 Flash	success:L4-L1	<0.0001	survives
Gemini 2.5 Flash	success:L4-L3	0.147	
Gemini 2.5 Flash	trap_acc:L1-L3	<0.0001	survives
Gemini 2.5 Flash	trap_acc:L4-L1	<0.0001	survives
Gemini 2.5 Flash	trap_acc:L4-L3	0.043	
Gemini 3.1 Flash-Lite	cache_frac:L1-L2	<0.0001	survives
Gemini 3.1 Flash-Lite	cache_frac:L1-L3	<0.0001	survives
Gemini 3.1 Flash-Lite	cache_frac:L2-L3	<0.0001	survives
Gemini 3.1 Flash-Lite	cache_frac:L4-L1	0.642	
Gemini 3.1 Flash-Lite	cache_frac:L4-L3	<0.0001	survives
Gemini 3.1 Flash-Lite	cost:L1-L2	<0.0001	survives
Gemini 3.1 Flash-Lite	cost:L1-L3	<0.0001	survives
Gemini 3.1 Flash-Lite	cost:L2-L3	<0.0001	survives
Gemini 3.1 Flash-Lite	cost:L4-L1	<0.0001	survives
Gemini 3.1 Flash-Lite	cost:L4-L3	<0.0001	survives
Gemini 3.1 Flash-Lite	int:L1-L2:distractor	1.000	
Gemini 3.1 Flash-Lite	int:L1-L2:length	0.643	
Gemini 3.1 Flash-Lite	int:L1-L2:trap_vs_neutral	0.111	
Gemini 3.1 Flash-Lite	int:L1-L3:distractor	0.538	
Gemini 3.1 Flash-Lite	int:L1-L3:length	0.847	
Gemini 3.1 Flash-Lite	int:L1-L3:trap_vs_neutral	<0.0001	survives
Gemini 3.1 Flash-Lite	int:L4-L1:distractor	0.719	
Gemini 3.1 Flash-Lite	int:L4-L1:length	0.011	survives
Gemini 3.1 Flash-Lite	int:L4-L1:trap_vs_neutral	0.0014	survives
Gemini 3.1 Flash-Lite	neutral_acc:L1-L2	1.000	
Gemini 3.1 Flash-Lite	neutral_acc:L1-L3	1.000	
Gemini 3.1 Flash-Lite	neutral_acc:L2-L3	1.000	

Model	Test	p	BH
Gemini 3.1 Flash-Lite	neutral_acc:L4-L1	1.000	
Gemini 3.1 Flash-Lite	neutral_acc:L4-L3	1.000	
Gemini 3.1 Flash-Lite	success:L1-L2	0.334	
Gemini 3.1 Flash-Lite	success:L1-L3	<0.0001	survives
Gemini 3.1 Flash-Lite	success:L2-L3	<0.0001	survives
Gemini 3.1 Flash-Lite	success:L4-L1	0.034	
Gemini 3.1 Flash-Lite	success:L4-L3	<0.0001	survives
Gemini 3.1 Flash-Lite	trap_acc:L1-L2	0.116	
Gemini 3.1 Flash-Lite	trap_acc:L1-L3	<0.0001	survives
Gemini 3.1 Flash-Lite	trap_acc:L2-L3	<0.0001	survives
Gemini 3.1 Flash-Lite	trap_acc:L4-L1	0.0013	survives
Gemini 3.1 Flash-Lite	trap_acc:L4-L3	<0.0001	survives
Gemini 3.6 Flash	cache_frac:L1-L3	0.351	
Gemini 3.6 Flash	cost:L1-L3	0.196	
Gemini 3.6 Flash	int:L1-L3:distractor	0.597	
Gemini 3.6 Flash	int:L1-L3:length	1.000	
Gemini 3.6 Flash	int:L1-L3:trap_vs_neutral	0.016	survives
Gemini 3.6 Flash	neutral_acc:L1-L3	1.000	
Gemini 3.6 Flash	success:L1-L3	0.126	
Gemini 3.6 Flash	trap_acc:L1-L3	0.030	
family	int:L1-L3	0.035	
family	int:L1-L3:common	0.092	

Model	Contrast	Gap slope per call (p)	High – low distractor (p)	Trap – neutral gap (p)
Gemini 2.5 Flash	L1 – L3 (content)	+0.0049 (0.355)	+0.083 (0.377)	-0.273 (<0.0001)
	L4 – L1 (note)	-0.0033 (0.559)	-0.125 (0.196)	+0.215 (<0.0001)
Gemini 3.1 Flash-Lite	L1 – L2 (order)	+0.0012 (0.643)	+0.000 (1.000)	-0.036 (0.111)
	L1 – L3 (content)	+0.0006 (0.847)	-0.035 (0.538)	-0.177 (<0.0001)
	L4 – L1 (note)	-0.0065 (0.011)	+0.017 (0.719)	+0.073 (0.0014)
Gemini 3.6 Flash	L1 – L3 (content)	-0.0000 (1.000)	+0.074 (0.597)	-0.139 (0.016)

Table 5: Interaction tests by label permutation. The gap slope is the change in the trap-accuracy contrast per additional tool call in the episode. The trap-minus-neutral column is the difference of differences; bold survives BH.

F Per-cell tables

Model	Layout	6 calls	12	18	30	low distr.	high distr.
Gemini 2.5 Flash	L1 append-only	0.611	0.694	0.722	0.611	0.630	0.690
	L3 pruned	1.000	0.944	0.926	0.861	0.944	0.921
	L4 hybrid	0.889	0.917	0.889	0.806	0.907	0.843
Gemini 3.1 Flash-Lite	L1 append-only	0.778	0.778	0.917	0.750	0.817	0.794
	L2 reordered	0.861	0.819	0.880	0.806	0.853	0.830
	L3 pruned	1.000	0.986	0.972	0.972	0.977	0.988
	L4 hybrid	0.944	0.903	0.898	0.771	0.882	0.876
Gemini 3.6 Flash	L1 append-only	0.889	–	0.889	–	0.852	0.926
	L3 pruned	1.000	–	1.000	–	1.000	1.000

Table 6: Trap accuracy by episode length (tool calls) and by distractor level, per model and layout. Each length cell holds 36 episodes for the full grid.

Model	Layout	own	external	chain 1	chain 2	chain 3	inventory	calendar	tickets
Gemini 2.5 Flash	L1 append-only	0.988	0.400	0.733	0.597	0.556	0.691	0.677	0.611
	L3 pruned	1.000	0.840	0.956	0.889	0.778	0.910	0.969	0.920
	L4 hybrid	1.000	0.750	0.889	0.861	0.722	0.778	0.944	0.903
Gemini 3.1 Flash-Lite	L1 append-only	1.000	0.655	0.850	0.771	0.750	0.804	0.788	0.825
	L2 reordered	1.000	0.705	0.867	0.806	0.806	0.861	0.844	0.819
	L3 pruned	1.000	0.960	1.000	0.951	0.972	0.995	0.976	0.977
	L4 hybrid	1.000	0.735	0.922	0.792	0.750	0.887	0.899	0.851
Gemini 3.6 Flash	L1 append-only	1.000	0.800	0.889	0.889	–	0.972	0.722	0.972
	L3 pruned	1.000	1.000	1.000	1.000	–	1.000	1.000	1.000

Table 7: Trap accuracy by trap kind (own update versus external update revealed in a listing), by chain length (number of updates before the question) and by domain.

Model	Layout	Action exec.	Reveal exec.	Trap acc. (all)	Trap acc. (thread fully executed)
Gemini 2.5 Flash	L1 append-only	0.997	1.000	0.661	0.66
	L3 pruned	0.997	1.000	0.911	0.91
	L4 hybrid	1.000	1.000	0.861	0.86
Gemini 3.1 Flash-Lite	L1 append-only	0.998	0.995	0.808	0.81
	L2 reordered	0.998	1.000	0.836	0.83
	L3 pruned	0.996	0.990	0.978	0.98
	L4 hybrid	0.995	0.990	0.853	0.85
Gemini 3.6 Flash	L1 append-only	0.984	0.975	0.889	0.88
	L3 pruned	0.988	1.000	1.000	1.00

Table 8: Action-fidelity sensitivity. “Reveal exec.” is the share of scripted reveal steps the agent actually executed with the right arguments. The last two columns restrict trap accuracy to questions whose thread steps were all executed; “skipped” counts questions where a thread step was not executed, so the fresh value never entered the context.

G Batteries

Repeat battery. 36 task cells on L1 were re-run (five runs each, temperature 0). The trap-accuracy outcome differed between runs in 11.1 percent of cells and episode success in 8.3 percent; the mean within-cell standard deviation of question accuracy is 0.017.

Phrasing battery (Gemini 3.1 Flash-Lite). 36 episodes were re-run with the second phrasing of every instruction and question. Question accuracy under phrasing 2 minus phrasing 1: -0.007 [-0.042, 0.032], sign-flip $p = 0.879$.

Cache on/off. Not run: the only provider that documents output-neutral explicit caching (Anthropic) was unreachable on this workspace’s OpenRouter key (a guardrail that blocks every non-Google model; probed before the plan was frozen and again after the main grid). The implicit-cache hit-versus-miss comparison within layout on Google is reported in its place.

H Hand audit

Sample: `analyze.py –audit-export 40`: wrong trap answers from the main gemini-3.1-flash-lite grid, stratified across layouts (L1 11, L2 11, L3 8 = all L3 wrong answers, L4 10), seeded. Wrong trap answers in the main grid: 189. Coder: Claude Fable 5.1 (the agent that ran the study), coded from `results/_audit_view_0_40.txt` which lists every observation of the queried field before the question; not blind to layout. Date: 2026-10-10.

`stale_value` the answer equals an earlier observation of the same (record, field) that a later tool result had superseded

Model	L3 – L1 gap	L4 – L1 gap	Recovery [CI]	L4/L1 cached share	Cheapest per correct (bill)
Gemini 2.5 Flash	+0.273	+0.215	0.79 [0.58, 0.97]		0.95
Gemini 3.1 Flash-Lite	+0.177	+0.073	0.42 [0.21, 0.60]		1.00

Table 9: D3 and D5 summary per model. Recovery is $(L4 - L1)/(L3 - L1)$ on trap accuracy with a task-bootstrap interval.

Model	0.0×	0.01×	0.025×	0.05×	0.1×	0.25×	0.5×	0.75×	1.0×
Gemini 2.5 Flash	L3	L3	L3	L3	L3	L3	L3	L3	L3
Gemini 3.1 Flash-Lite	L3	L3	L3	L3	L3	L3	L3	L3	L3
Gemini 3.6 Flash	L3	L3	L3	L3	L3	L3	L3	L3	L3

Table 10: Cheapest layout per correct task as the cache-read price moves from free (0×) to the full input price (1×); the billed point for Google sits at 0.1×.

reveal_not_executed the agent skipped the scripted tool call that would have revealed the fresh value, so the fresh value never entered the context (an action-fidelity failure, not a stale read)

ambiguous_stale_or_boundary below-threshold question where the fresh quantity equals the threshold; the wrong answer fits both a stale read and a strict-inequality misreading

arithmetic_or_count the answer matches no observed value and looks derived

distractor_record the answer equals another record’s value for the same field (none found as the sole explanation)

abstain_or_unparsed no parseable answer (none in sample)

Code	L1	L2	L3	L4	All
stale_value	11	9	5	9	34
ambiguous_stale_or_boundary	0	2	0	1	3
reveal_not_executed	0	0	2	0	2
arithmetic_or_count	0	0	1	0	1
total	11	11	8	10	40

Table 11: Hand-coded wrong trap answers by layout. The per-item codes and notes ship in results/audit_sample.json.

I Cost and compute

Every call was billed through OpenRouter and logged by a locked cost tracker with a hard stop at USD 24. The total below includes pilots, calibration episodes, smoke tests, discarded runs and the external arm; the main grids are the gemini-* rows. No GPU was used; the executor, layouts and analysis run on a laptop in under a minute.

Item (OpenRouter billed)	USD
google/gemini-3.1-flash-lite	11.82
google/gemini-2.5-flash	3.90
google/gemini-3.6-flash	2.55
google/gemini-2.5-flash-lite	0.12
google/gemini-3.5-flash-lite	0.03
_smoke_and_diag_2026-10-10	0.01
total (24817 calls)	18.42

Table 12: Metered spend by model and arm (USD, OpenRouter usage.cost).