LLM AGENTS ARE LATENT CONTEXT MANAGERS: ELICITING SELF-MANAGED CONTEXT VIA STATE PROPRIOCEPTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon tool agents are bottlenecked by how their context grows toward the limits of the context window. Existing systems delegate context management to runtime rules or train agents to compress their working memory, often leaving runtime state implicit or discarding evidence through summarization. We argue that a basic obstacle is context-state blindness: agents must decide what to keep or archive without an explicit view of block costs and remaining capacity. We introduce VISTA (Visible Internal State for Tool Agents), a training-free, model-agnostic layer that represents working memory as typed, addressable blocks, exposes token usage, creation age, archive status, and remaining budget through a runtime dashboard, and archives blocks as recoverable full-fidelity payloads. Across LOCA-Bench, BrowseComp-Plus, and GAIA, the same untrained interface transfers across trajectory scales. On LOCA-Bench, it raises Gemini-3-Flash accuracy from 22.7% to 50.7%; it reaches 58.0% on BrowseComp-Plus and remains competitive on GAIA. Pressure sweeps characterize its operating range, gains transfer across four backbones, and ablations show that the dashboard contributes beyond archive and recovery tools.

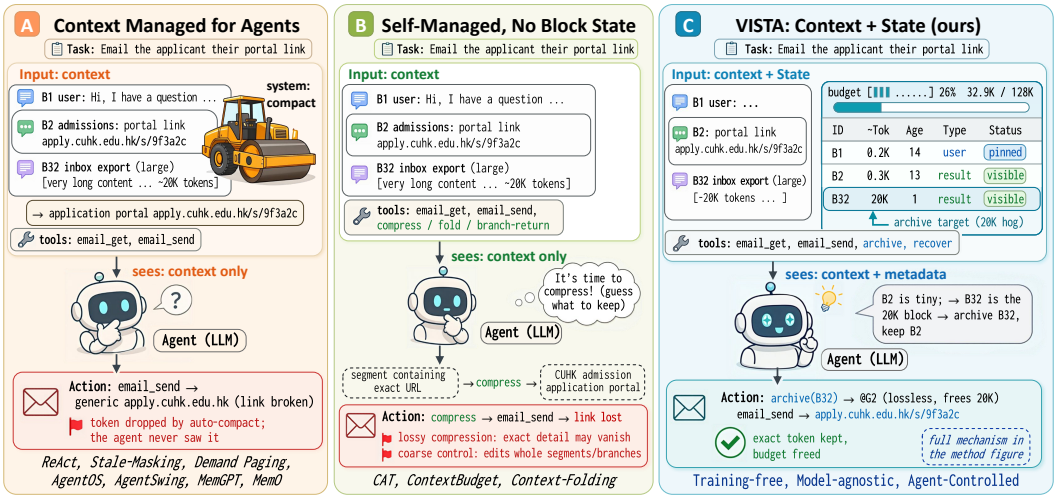


Figure 1: Context control with visible runtime state. External policies choose context edits; self-managed agents choose their own actions. VISTA adds an actionable block ledger and recoverable payloads, giving the policy the addresses, sizes, and status needed to manage its working set.

1 INTRODUCTION

Language agents operate over stateful tasks such as filling spreadsheets from web and email evidence, modifying databases, debugging code, and coordinating business workflows (Zeng et al., 2026; Liu et al., 2025; Xu et al., 2026a). Their context is working memory: it accumulates tool evidence, failed attempts, constraints, hypotheses, and identifiers that may remain essential many

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.

steps later (Packer et al., 2023; Verma, 2026; Wu et al., 2026b). As this history crowds the context window, the agent must decide what to keep visible, what to set aside, and what to recover. These decisions determine which evidence remains available for subsequent reasoning and action.

Existing approaches differ in who makes these decisions. External policies mask stale observations or page context by runtime rules (Zhang et al., 2026a; Packer et al., 2023). These policies act on runtime statistics, while the agent reasons about which evidence matters for the task. Learned approaches instead train agents to compress or reorganize working memory (Liu et al., 2025; Verma, 2026; Wu et al., 2026b; Sun et al., 2025), but summarization or deletion can remove evidence needed later. Across these approaches, a basic interface question remains: what context state can the decision-making policy actually observe? Block contents alone do not explicitly expose token costs, creation age, archive status, or remaining budget. Providing this state alongside addressable context actions connects resource constraints to the agent's keep-or-archive decisions (Figure 1).

We ask whether explicit context state can enable capable models to manage their own working memory. Context management is a meta-tool decision under partial observability: the agent acts on its own working memory using the state in its prompt. Figure 2 motivates the connection between perception and management. During context-tool RL, task performance and dashboard-free perception accuracy improve together, although probe accuracy is excluded from the reward. Supplying the dashboard yields high perception accuracy even before adaptation. We therefore expose runtime state explicitly and express management criteria as a rubric. This *context proprioception* enables a frozen policy to apply its existing capabilities to working-memory decisions.

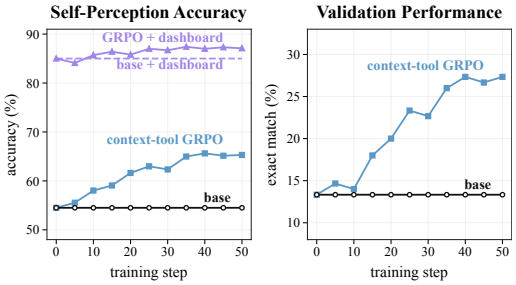


Figure 2: Perception improves during management training. Qwen3-8B context-tool GRPO checkpoints show rising task performance and dashboard-free perception accuracy, although probe accuracy is excluded from the reward. Dashboard scores stay high.

This view motivates three interface requirements: *observable state*, including per-block cost, creation age, archive status, and remaining budget; *reversible archiving*, so evidence can be recovered when its relevance changes; and *model-agnostic operation*, allowing the same interface to elicit management behavior across backbones without additional training.

We introduce VISTA (Visible Internal State for Tool Agents), which implements these requirements with typed, addressable blocks, a runtime dashboard, and recoverable storage. The agent archives selected blocks behind stable handles and reads exact payloads when needed. Two idealized task families formalize how retained information limits exact delayed recall and how block-size information constrains evidence selection (Proposition 1 and Theorem 1). These bounds motivate recoverable storage and observable state; Section 3 specifies their assumptions.

Across LOCA-Bench, BrowseComp-Plus, and GAIA, VISTA reaches 50.7%, 58.0%, and 73.3% accuracy (Table 1). Removing the dashboard while retaining archive and recovery lowers accuracy by 13.4, 8.0, and 4.8 percentage points. Pressure sweeps characterize the operating range; the same interface improves all four tested backbones and transfers to offline AMA-Bench trajectory memory. The Qwen3-8B adaptation study further reports ID and OOD task gains.

Our contributions are threefold.

- **Eliciting context-management capabilities.** We frame management as a meta-tool decision under partial observability and study how runtime state enables frozen-model memory management.
- **A proprioceptive, recoverable context interface.** VISTA combines an actionable block ledger with exact-payload recovery. Two resource bounds formalize how retained information and block-size information constrain recall and evidence selection.
- **Evidence across task scales and model families.** Three online benchmarks, four backbones, component ablations, pressure sweeps, and perception probes test the interface; offline trajectory-memory evaluation and a small RL study examine transfer and adaptation.

2 Related Work

Runtime management and recoverable memory. Stale-observation masking and paging-oriented runtimes manage context movement through external policies (Zhang et al., 2026a; Mason, 2026; Li et al., 2026; Feng et al., 2026). Structured eviction and cache-aware systems address pruning and ingestion cost (Semenov & Dorofeev, 2026; Xu et al., 2026d). MemGPT gives the model memory editing and retrieval operations, pressure warnings, and recall storage (Packer et al., 2023). PACE selects historical context at different granularities and exposes `glimpse` for full chunk details (Wei et al., 2026); HiAgent organizes working memory around subgoals and retrieves earlier action–observation trajectories (Hu et al., 2025). VISTA exposes the active working set as an actionable per-block runtime ledger, pairing direct archive control with recoverable original payloads.

Persistent-memory and retrieval systems organize experience in external stores (Chhikara et al., 2025; Liu et al., 2026; Ji et al., 2026; Yang et al., 2026; Zhang et al., 2026b; Mi et al., 2026). Our AMA-Bench adaptation applies the VISTA workspace to completed trajectories and evaluates subsequent question answering (Zhao et al., 2026).

Context actions and learned management. Context as a Tool integrates context maintenance into the agent's action space and trains a compressor on annotated trajectories (Liu et al., 2025). ContextBudget learns budget-aware compression (Wu et al., 2026b), while Context-Folding and LongSeeker expose structured context-control operations (Sun et al., 2025; Lu et al., 2026b). Active Context Compression uses agent-written knowledge blocks (Verma, 2026). Other systems develop compression guidance, learned memory operations, specialized controllers, and compact reasoning summaries (Kang et al., 2025; Yi et al., 2026; Liang et al., 2026; Yu et al., 2026; Zhang et al., 2026c; Kontonis et al., 2026; Wu et al., 2026a; Aghajohari et al., 2025; Wu et al., 2025; Wang et al., 2026). SUPO jointly optimizes tool-use and summarization policies through agentic RL (Lu et al., 2026a). VISTA's main evaluation holds model weights fixed and studies explicit block metadata alongside lossless storage. Appendix L studies RL adaptation with the same workspace.

Self-state awareness. Work on budget awareness, temporal blindness, and externalization examines models' judgments about their own state (Ackerman, 2026; Lin et al., 2026; Cheng et al., 2025; Zhou et al., 2026). Probing studies also connect the retention of agent plans to visible context (Mehta & Datta, 2026). VISTA supplies a runtime description of the working set: which blocks exist, their token estimates and creation ages, and their visible or archived status. Size probes evaluate the readability of this description; task-level ablations evaluate the contribution of the dashboard in long-horizon tool use (Zeng et al., 2026).

3 Methodology

VISTA exposes context as a workspace $W_t = (V_t, A_t)$, consisting of visible blocks V_t and externally archived payloads A_t. The frozen model selects both task actions and context actions from the current rendering. Figure 3 shows the interface; Appendix G specifies its full contract.

3.1 Workspace and Context Stream

At step t, the agent has a goal g, interaction history $H_t = (a_1, o_1, \ldots, a_{t-1}, o_{t-1})$, environment tools $\mathcal{T}_{\text{env}}$, and budget B. VISTA registers messages and tool outputs as blocks before assembling the next model input. The raw history is logged for evaluation; the workspace determines which evidence the model sees. Three invariants connect storage and execution: admitted inputs fit the harness budget, each actionable unit has a stable ID or handle, and archived bytes remain available unless explicitly deleted. The dashboard uses the same token estimator as the harness budget checks.

Context stream and dashboard. Each observation receives a stable block ID. Visible blocks retain their exact content; archived blocks leave compact summaries and handles in the prompt. The dashboard is regenerated after each registered tool result and reports block IDs, token estimates, creation age, type, archive level, status, and remaining budget. These fields connect a context action to a specific object and its expected space saving. Pinned blocks preserve required task and protocol content. Tool-call/result links remain valid after archiving.

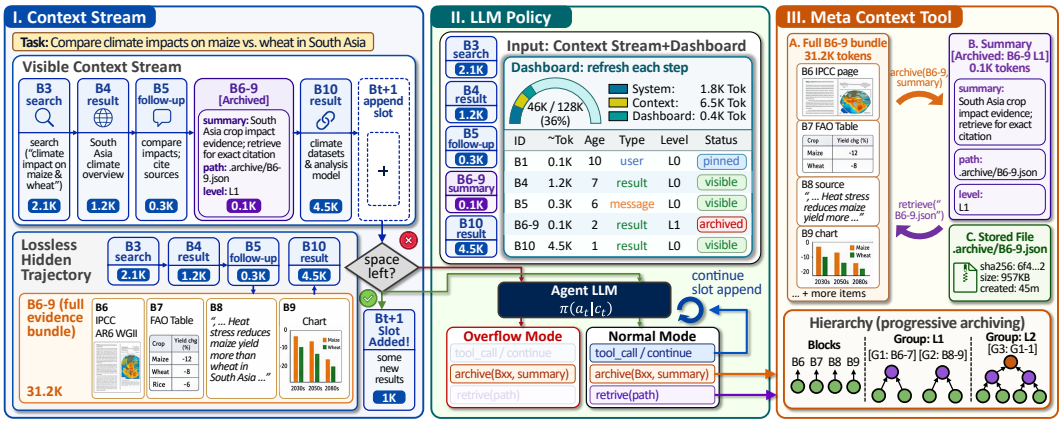


Figure 3: The VISTA workspace. Tool observations become addressable blocks; a refreshed ledger exposes their size, creation age, and archive status. The agent selects blocks to archive and uses their stable paths to recover original evidence, coupling a bounded working set with persistent payloads.

3.2 Policy and Recoverable Context Actions

A shared policy for task and context actions. The same policy receives the current rendering C_t and chooses an action:

$$a_t \sim \pi_\theta(\cdot \mid C_t), \qquad a_t \in \mathcal{T}_{\text{env}} \cup \mathcal{T}_{\text{ctx}} \cup \{\text{answer}\}, \qquad \mathcal{T}_{\text{ctx}} = \{\text{archive}, \text{delete}\}.$$

Recovery is an ordinary file or terminal read in $\mathcal{T}_{\text{env}}$. After a tool result is registered, the harness refreshes the ledger before the next decision. The rubric directs the agent to consider task relevance together with block cost, archive status, and remaining budget when choosing a working set. The model chooses both the task operation and the evidence available for later decisions.

Archive and recovery. The dedicated tools are $\text{archive}(\mathcal{S}, \rho)$ and $\text{delete}(\mathcal{S})$, where $\mathcal{S}$ is a set of block IDs and ρ a replacement summary. Archiving stores the original transcript bytes and returns a handle h with a payload path; an ordinary file or terminal read of h recovers those bytes. The model can read an exact range, search a payload, or follow a coarser handle to a nested archive. Summaries guide navigation; exact payloads supply evidence. Deletion permanently removes selected content. Source pagination or truncation remains part of the stored observation, and the agent can re-query the source when more evidence is needed.

Hierarchical access. An archive can group several raw blocks into one bundle; subsequent archives can collect those handles into a coarser group. Recovery traverses this hierarchy in reverse: the agent opens a handle, locates the relevant payload, and reads the needed range or sub-block. This supports returning an exact row or identifier to active context without reloading an entire transcript. Handles preserve the navigation path while summaries help select which payload to inspect.

3.3 Budgeted Execution and Resource Bounds

Budgeted execution. The candidate input combines visible blocks, handles, and the dashboard:

$$\widetilde{C}_t = \text{render}(V_t) \oplus \text{handles}(A_t) \oplus D_t, \qquad C_t = \text{preflight}(\widetilde{C}_t, B).$$

A feasible admitted input satisfies $|C_t| \le B$ under the harness token metric, including instructions and tool schemas. Overflow enables only context actions in a compact management rendering; ordinary task tools resume after cleanup. The minimum management input must fit the cap, otherwise preflight returns a budget-infeasible outcome. A separate runtime guard can offload visible tool results and reject newly returned results that cannot fit. Provider admission also requires $\text{tokens}_{\text{provider}}(C_t) + r_{\text{out}} \le H$, with provider limit H and output reserve r_{out}. The experiments evaluate this combined agent–runtime loop (Appendix H.1); guard-trigger and tokenizer-calibration records remain implementation checks.

The candidate/admitted-input distinction also determines the interaction loop. The overflow decision uses the candidate size; the model always receives the admitted, budget-safe rendering. After

cleanup, the harness rebuilds the candidate and rechecks feasibility before enabling task tools. This keeps state reporting, available actions, and request admission synchronized.

Why state information and recovery matter. Two idealized task families isolate these resources (full statements and proofs in Appendices A–E). In delayed exact recall, N independent k-bit blocks are compressed into a state of entropy at most b bits before a uniform independent query index is revealed. Proposition 1 bounds non-recovering success by $\min\{1, b/(Nk) + 1/k\}$. A reliable archive permits perfect recall whenever the instructions, handles, and requested block fit. Here b is a bit budget; a token cap implies such a bound only through an encoding argument.

In the make-room family, one unneeded block frees L tokens, whereas each of $n-1$ future-relevant blocks frees ℓ tokens. With integer $\kappa = L/\ell$, $2 \le \kappa < n/2$, overflow of L, and only binary overflow feedback between distinct archive actions, Theorem 1 gives

$$\mathbb{E}[Z] \ge \kappa \left[1 - \frac{I+1}{\log_2((n-\kappa)/\kappa)}\right]_+ ,$$

where Z counts archived future-relevant blocks and $I = I(J^\star; Y)$ includes *all* initial information about the large-block index. A full size ledger identifies that block and achieves $Z = 0$. Under atomic paging, serving all future requests requires restoring the Z archived blocks. The assumptions specify when state information saves useful resident evidence; the experiments measure task success and archive/read events in general tool workflows.

4 Experiments

4.1 Evaluation Setup

Benchmarks and budgets. We evaluate three online regimes: the public 75-configuration LOCA-Bench suite (Zeng et al., 2026) with million-token trajectories; 150 BrowseComp-Plus questions (Chen et al., 2025) with 100K-scale retrieval; and a fixed 165-question GAIA validation subset (Mialon et al., 2024). The main LOCA-Bench runs use Gemini-3-Flash at 128K. BrowseComp-Plus and GAIA use DeepSeek-V4-Pro with a 12K active window; their declared cumulative budgets are 160K and 80K. BrowseComp-Plus deliberately stresses retention: 12K is below the reported median first retrieval of a gold document ($\approx$ 17K tokens). We retain each benchmark's scorer and count errors and timeouts as failures. Most cells use one trajectory per task. Appendix I gives protocols; Appendix M lists open budget and accounting checks.

Comparisons. Table 1 includes fixed masking/clearing policies, agent-mediated compression, and recoverable-storage systems. Task instances, backbone, native tools, context budget, and scoring are shared within a benchmark. Harness-native baselines share the agent loop and prompt assembly. SLIM and Active Context Compression use inference-time reproductions; Skeleton Compression implements structured compression inspired by context-as-a-tool methods (Yen et al., 2025; Verma, 2026; Liu et al., 2025). Claude Code retains its released CLI harness (May 6, 2026) with benchmark-native MCP task tools, providing a complete-system comparison. The capability matrix and prompts appear in Appendices H and I. The evaluations in this section hold model weights fixed; Section 4.4 studies Qwen3-8B adaptation.

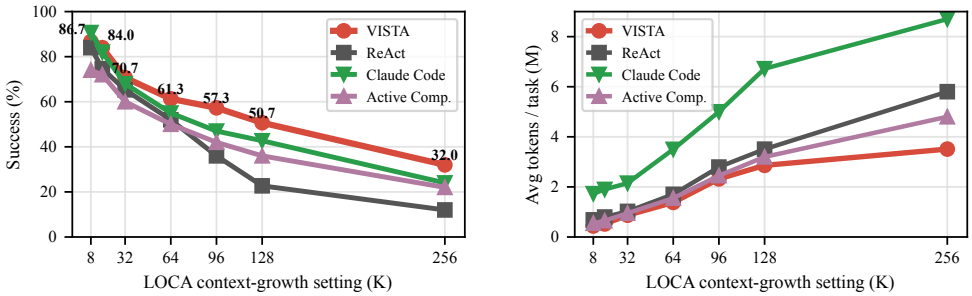
4.2 Task Success and Trajectory Length

VISTA solves 38/75 LOCA-Bench tasks, compared with 17/75 for ReAct, 33/75 for Auto-Archive + Recover, and 32/75 for Claude Code. VISTA uses 2.86M trajectory tokens and 36.4 steps per LOCA task, versus 6.72M and 171.5 for Claude Code (Table 5). It achieves 58.0% on BrowseComp-Plus and 73.3% on GAIA. The highest observed baseline scores are 52.0% and 73.9%, respectively. Removing the dashboard while retaining archive and recovery lowers accuracy by 13.4, 8.0, and 4.8 percentage points across the three benchmarks; removing recovery yields 45.3%, 43.3%, and 72.1%, respectively.

The trajectory-length comparison varies by benchmark. On BrowseComp-Plus, VISTA improves accuracy from ReAct's 39.3% to 58.0% while reducing mean trajectory tokens from 163K to 135K.

Table 1: Task success and context cost. Accuracy (%) and mean trajectory tokens across three benchmarks. State/Ctrl/Recov indicate runtime metadata, agent control, and recovery (✓/✗/∼: yes/no/partial). VISTA leads on LOCA-Bench and BrowseComp-Plus. Bold/underline mark the top two accuracies.

	Mechanism			LOCA-Bench		BrowseComp-Plus		GAIA	
Method	State	Ctrl	Recov	Acc↑	Traj↓	Acc↑	Traj↓	Acc↑	Traj↓
Fixed external policy									
ReAct	✗	✗	✗	22.7	3.51M	39.3	163K	61.2	23K
Tool-result Clearing	✗	✗	✗	26.7	2.60M	42.7	161K	65.5	24K
Stale-obs. Masking	✗	✗	✗	28.0	3.32M	38.0	112K	61.8	24K
Skeleton Compression	✗	✗	✗	33.3	2.84M	40.0	139K	70.3	28K
Agent-mediated / lossy									
SLIM (summary)	✗	∼	✗	29.3	3.76M	49.3	162K	67.9	30K
Active Ctx. Compression	✗	✓	✗	36.0	3.20M	42.7	162K	71.5	39K
Context-Folding	✗	✓	✗	34.7	3.41M	43.3	166K	64.8	39K
Lossless external store									
Auto-Archive + Recover	✗	✗	✓	44.0	2.73M	45.3	133K	63.6	20K
Claude Code	✗	✓	∼	42.7	6.72M	<u>52.0</u>	247K	**73.9**	44K
Ours and ablations									
VISTA w/o dashboard	✗	✓	✓	37.3	5.25M	50.0	423K	68.5	24K
VISTA w/o recovery	✓	✓	✗	<u>45.3</u>	2.99M	43.3	161K	72.1	28K
VISTA (full)	✓	✓	✓	**50.7**	2.86M	**58.0**	135K	<u>73.3</u>	33K



(a) Success vs. context-growth setting **(b) Token cost vs. context-growth setting**

Figure 4: Performance under increasing context growth. On the same 75 LOCA-Bench tasks, growth settings range from 8K to 256K. Left: success; right: average API tokens per task. VISTA's advantage over ReAct expands from 2.7 points at 8K to 28.0 at 128K, with lower reported token use throughout.

On GAIA, accuracy rises from 61.2% to 73.3% with a longer mean trajectory, 33K versus 23K. Each comparison uses the same task set, backbone, and harness budget within its benchmark, allowing accuracy and token consumption to be read together in Table 1.

The VISTA–Claude Code difference on LOCA is 8.0 points (task-pair bootstrap 95% interval $[-4.0, 20.0]$; exact McNemar $p = 0.263$). Appendix I.1 reports paired uncertainty and marginal-only bounds for the other comparisons.

4.3 Context Pressure and Backbone Transfer

LOCA-Bench varies the volume of accumulated context on a fixed task suite (Figure 4; exact values in Table 8). At 8K, VISTA and ReAct score 86.7% and 84.0%; at 128K, they score 50.7% and 22.7%; at 256K, 32.0% and 12.0%. Both degrade as context growth increases, with VISTA retaining a larger fraction of solved tasks.

The BrowseComp-Plus sweep varies active-window capacity on a fixed task set (Figure 5). Both methods improve with larger windows; VISTA's margin peaks at intermediate windows. At 12K, its cumulative trajectory is 135K tokens versus ReAct's 163K. Figure 6 keeps the interface and instruc-

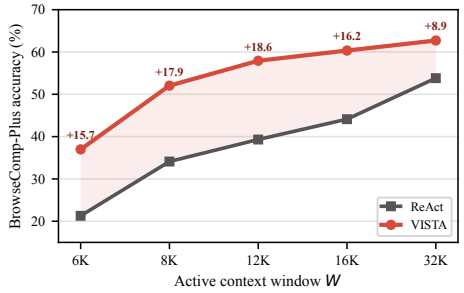


Figure 5: Active-window sensitivity. On a fixed BrowseComp-Plus task set, larger windows improve both methods. VISTA's margin over ReAct is largest at intermediate windows, where evidence retention remains consequential.

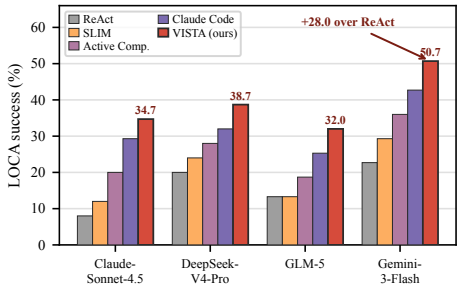


Figure 6: Transfer across four backbones. At the 128K LOCA-Bench setting, the same untrained VISTA interface has the highest observed success on each backbone among the plotted methods, using the same task and scoring protocol.

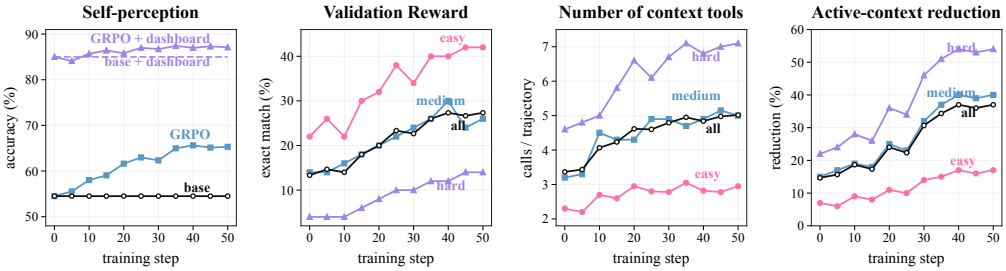


Figure 7: Training dynamics of context management. Across Qwen3-8B checkpoints, panels track perception, validation reward, context-tool calls per trajectory, and active-context reduction. Task and perception scores rise together; the hard group makes more calls and shows greater reduction than the easy group.

tions fixed across Claude-Sonnet-4.5, DeepSeek-V4-Pro, GLM-5, and Gemini-3-Flash, showing gains across all four backbones at the 128K LOCA-Bench setting.

4.4 Implicit Competence and RL Adaptation

Qwen3-8B receives task reward and penalties for context pressure, invalid actions, and archive–recovery reversals; probe accuracy is excluded. Evaluation uses BrowseComp-Plus; training uses BrowseComp-Plus (ID) or GAIA (OOD). VISTA RL gains 4.0 points over context-tool GRPO in both settings; zero-shot VISTA gains 6.7 over Base (Table 2). Pilot screening uses training prompts only. Appendix L gives the objective and protocols; Appendix M lists outstanding reproduction records.

The adaptation gain varies with the training regime: VISTA rises from 20.0% zero-shot to 31.3% with ID training and 21.3% with OOD training. The interface therefore yields a substantial gain before adaptation; the reported post-training benefit is larger when training and evaluation share a benchmark. All four variants use Qwen3-8B, keeping the backbone fixed in this comparison.

Perception and validation performance rise together during management training (Figures 2 and 7). The hard group has more calls per trajectory and greater context reduction; comparing management rates requires episode-length and occupancy controls.

4.5 Offline Trajectory-Memory Transfer

The offline setting asks the model to answer questions about completed agent histories. We replay actions and observations into the VISTA workspace before QA, organizing the history into the same addressable blocks used online. The resulting memory is then queried without continuing the original task. This tests whether the interface also supports retaining evidence across a completed trajectory and locating it for later questions.

We replay 208 completed AMA-Bench trajectories and answer 12 questions per episode (2496 pairs) (Zhao et al., 2026). VISTA leads in F1 at 0.382; the AMA agent has higher judge accu-

Method	ID ↑	OOD ↑
Zero-shot baselines		
Base	13.3	13.3
Ours (zero-shot)	20.0 (+6.7)	20.0 (+6.7)
RL adaptation		
Context-tool GRPO	27.3 (+14.0)	17.3 (+4.0)
Ours (RL)	**31.3** (+18.0)	**21.3** (+8.0)

Table 2: Post-training and transfer. Qwen3-8B accuracy on BrowseComp-Plus with ID/OOD training on BrowseComp-Plus/GAIA. VISTA RL leads in both settings. Green: gains over Base.

Method	F1	Acc.	Time/ep	Tokens/ep
BM25	0.335	0.575	**35.5s**	303.43K
EMem	0.363	0.651	166.2s	470.30K
Mem0	0.329	0.536	108.0s	**30.63K**
AMA	0.368	**0.753**	176.5s	268.98K
VISTA	**0.382**	0.731	43.7s	148.49K

Table 3: Transfer to trajectory memory. On 208 AMA episodes, VISTA leads in F1; AMA has higher judge accuracy. VISTA uses less generation time and fewer tokens.

racy (Table 3). VISTA also uses less time and fewer tokens than AMA. Domain results show higher F1 in five of six domains (Table 9); Appendix I specifies the timing scope and adapters.

4.6 Component and Interface Ablations

The variants separate context capabilities from their presentation. No-dashboard removes the state display while keeping archive and recovery; no-archive removes recoverable externalization; no-recovery disables access to archived payloads. Auto-Archive supplies a fixed archive policy. Minimal wording and the expanded status display change how the interface is described or shown. The task-success and event panels report the outcomes of these changes.

Figure 8 compares full VISTA (38 solved tasks), no-dashboard (28), no-archive (27), no-recovery (34), minimal archive wording (37), and an expanded status display (36). Auto-Archive uses a fixed archive policy and solves 33; its dashboard flag remains a configuration-verification item. The largest observed losses follow removal of archive capability or the dashboard. The wording and status variants differ from full VISTA by at most two solved tasks.

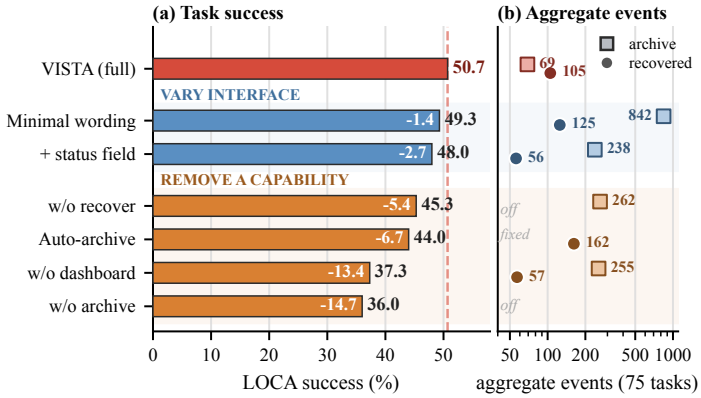


Figure 8: Component ablations. 75 LOCA tasks: accuracy (left), archive calls and successful recoveries (right, log scale). Dashboard removal lowers accuracy; *off*: disabled.

Management frequency varies much more: minimal wording produces 842 archive calls at 49.3% accuracy, versus 69 calls at 50.7% for full VISTA. No-dashboard records 255 archives and 57 successful recoveries. Thus call frequency alone does not determine task accuracy or selection quality. Events may involve multiple blocks and repeated reads. In no-recovery, the 57 recorded attempts all failed; the disabled capability is shown as *off*.

Across four backbones, dashboard-free total-size estimates have median relative error 0.43–0.84; the factual ledger reduces it to zero (Appendix K, Table 12).

5 Context Management in Action

Archive and later access. Figure 9 follows one 128K LOCA email-triage run. The dashboard identifies a large inbox-export block; the agent archives it, reduces active occupancy, and later reads the exact payload for the final action. The stable handle connects the archive event to later evidence access. A matched baseline summarizes or clears the block and loses the verbatim value. Across rescued tasks, archive-then-recover occurs in 8 of the 16 runs containing an archive. In 13 other archive-containing runs outside that subset, archiving frees space without a later read (Table 11).

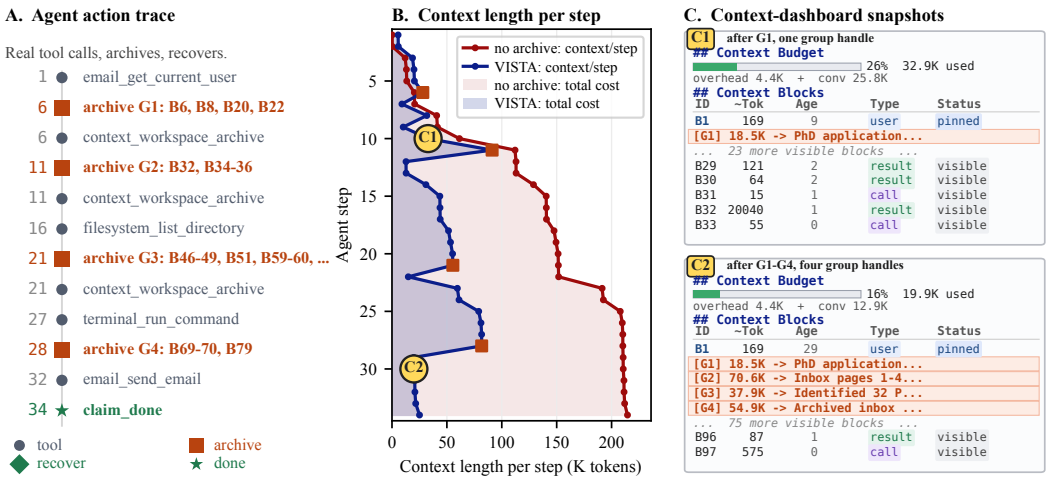


Figure 9: Context management along an email-triage task. At 128K on LOCA, four archives reduce active context (center) and retain recovery handles (right). The completed trace illustrates deferred evidence access; the no-archive curve reconstructs occupancy.

These traces exhibit two uses of the same workspace: keeping active context small and preserving evidence for delayed use.

Dashboard footprint. Figure 10 uses three 30-query BrowseComp-Plus runs sampled from the 150-question evaluation, with DeepSeek-V4-Pro, $W = 12{,}288$, and $B = 163{,}840$. The traces use this subset; the reported 58.0% task accuracy uses all 150 questions. The horizontal axis counts cumulatively registered blocks, while the right panel counts blocks still visible to the model.

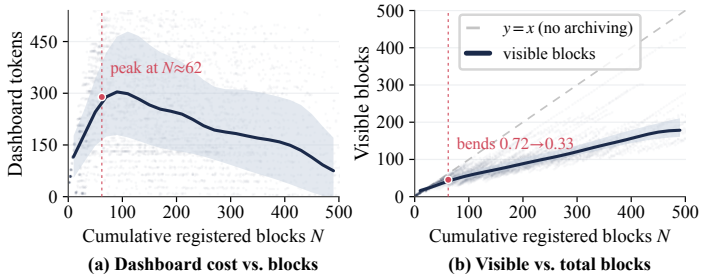


Figure 10: Dashboard and working-set traces. Three 30-query BrowseComp-Plus runs at 12K. Net dashboard tokens fall as visible blocks grow. Curves: binned medians; bands: interquartile ranges.

Per-turn net counts subtract a fixed 492-token system/schema component measured with `cl100k_base`. The plotted median peaks near 62 registered blocks at roughly 340 tokens, then declines while the visible working set continues to grow. The visible-block median stays below the no-archiving diagonal as registered history grows. This gap summarizes how many registered blocks are absent from the visible set, complementing the left panel's measurement of the dashboard's rendered token footprint. Curves and bands summarize binned medians and interquartile ranges. Interpreting this decline requires matching each plotted point to the same-turn rendered ledger, displayed rows/groups, paging decisions, and token counts. Appendix M records this outstanding accounting check.

6 Conclusion

VISTA exposes working memory through an actionable ledger and recoverable payloads. Three online benchmarks, four backbones, and component ablations support the value of explicit runtime state for frozen-model context management. Size probes reveal a consistent estimation gap across four backbones; archive/read traces demonstrate deferred access to exact evidence. The resource bounds formalize how retained information and observable block sizes constrain recall and evidence selection. RL adaptation improves task performance alongside state-estimation gains, further connecting context perception and management.

AI USE STATEMENT

OpenAI Codex assisted with manuscript revision and translation, critical review of mathematical arguments, interpretation of reported results, experimental-design feedback, statistical-checking code, figure editing, and LaTeX organization and formatting. These uses include refining the exposition of assumptions and comparing empirical claims with the available evidence.

REPRODUCIBILITY STATEMENT

Appendices A–E specify the theoretical assumptions and proofs. Appendices G and H describe the workspace contract, context loop, prompts, and tool schemas; Appendices I and I.1 give evaluation protocols and paired-outcome analyses. Appendix L contains the RL extension. The remaining implementation and accounting records are enumerated in Appendix M.

REFERENCES

Christopher Ackerman. Evidence for limited metacognition in LLMs, 2026. URL https://arxiv.org/abs/2509.21545.

Milad Aghajohari, Kamran Chitsaz, Amirhossein Kazemnejad, Sarath Chandar, Alessandro Sordoni, Aaron Courville, and Siva Reddy. The markovian thinker: Architecture-agnostic linear scaling of reasoning. *arXiv preprint arXiv:2510.06557*, 2025.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent, 2025. URL https://arxiv.org/abs/2508.06600.

Yize Cheng, Arshia Soltani Moakhar, Chenrui Fan, Parsa Hosseini, Kazem Faghih, Zahra Sodagar, Wenxiao Wang, and Soheil Feizi. Your LLM agents are temporally blind: The misalignment between tool use decisions and human time perception. *arXiv preprint arXiv:2510.23853*, 2025.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Zhaopeng Feng, Liangcai Su, Zhen Zhang, Xinyu Wang, Xiaotian Zhang, Xiaobin Wang, Runnan Fang, Qi Zhang, Baixuan Li, Shihao Cai, et al. AgentSwing: Adaptive parallel context management routing for long-horizon web agents. *arXiv preprint arXiv:2603.27490*, 2026.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32779–32798, 2025.

Shuo Ji, Yibo Li, and Bryan Hooi. Memory is reconstructed, not retrieved: Graph memory for llm agents, 2026. URL https://arxiv.org/abs/2606.06036.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. Acon: Optimizing context compression for long-horizon llm agents. *arXiv preprint arXiv:2510.00615*, 2025.

Vasilis Kontonis, Yuchen Zeng, Shivam Garg, Lingjiao Chen, Hao Tang, Ziyan Wang, Ahmed Awadallah, Eric Horvitz, John Langford, and Dimitris Papailiopoulos. Memento: Teaching llms to manage their own context. *arXiv preprint arXiv:2604.09852*, 2026.

ChengYou Li, XiaoDong Liu, XiangBao Meng, and XinYu Zhao. Architecting agentos: From token-level context to emergent system-level intelligence, 2026. URL https://arxiv.org/abs/2602.20934.

Jiaqing Liang, Jinyi Han, Weijia Li, Xinyi Wang, Zhoujia Zhang, Zishang Jiang, Ying Liao, Tingyun Li, Ying Huang, Hao Shen, et al. GenericAgent: A token-efficient self-evolving llm agent via contextual information density maximization. *arXiv preprint arXiv:2604.17091*, 2026.

Yuxiang Lin, Zihan Wang, Mengyang Liu, Yuxuan Shan, Longju Bai, Junyao Zhang, Xing Jin, Boshan Chen, Jinyan Su, Xingyao Wang, Jiaxin Pei, and Manling Li. Bagen: Are llm agents budget-aware?, 2026. URL https://arxiv.org/abs/2606.00198.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. SimpleMem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*, 2026.

Shukai Liu, Jian Yang, Bo Jiang, Yizhi Li, Jinyang Guo, Xianglong Liu, and Bryan Dai. Context as a tool: Context management for long-horizon swe-agents. *arXiv preprint arXiv:2512.22087*, 2025.

Miao Lu, Weiwei Sun, Weihua Du, Zhan Ling, Xuesong Yao, Kang Liu, and Jiecao Chen. Beyond the context window: Scaling agentic rl via end-to-end optimized context compression. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21074–21125, 2026a. URL https://aclanthology.org/2026.acl-long.966/.

Yijun Lu, Rui Ye, Yuwen Du, Jiajun Wang, Songhua Liu, and Siheng Chen. Longseeker: Elastic context orchestration for long-horizon search agents. *arXiv preprint arXiv:2605.05191*, 2026b.

Tony Mason. The missing memory hierarchy: Demand paging for llm context windows, 2026. URL https://arxiv.org/abs/2603.09023.

Aman Mehta and Anupam Datta. Plans don't persist: Why context management is load bearing for llm agents. *arXiv preprint arXiv:2606.22953*, 2026.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-pro: Learning reusable skills from experience via non-parametric ppo for llm agents, 2026. URL https://arxiv.org/abs/2602.01869.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general ai assistants. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. MemGPT: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Yuta Sakai. Generalizations of Fano's inequality for conditional information measures via majorization theory. *Entropy*, 22(3):288, 2020. doi: 10.3390/e22030288.

Andrew Semenov and Svyatoslav Dorofeev. Beyond compaction: Structured context eviction for long-horizon agents. *arXiv preprint arXiv:2606.11213*, 2026.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon llm agent via context-folding. *arXiv preprint arXiv:2510.11967*, 2025.

Nikhil Verma. Active context compression: Autonomous memory management in llm agents. *arXiv preprint arXiv:2601.07190*, 2026.

Zhenting Wang, Huancheng Chen, Jiayun Wang, and Wei Wei. Memex (rl): Scaling long-horizon llm agents via indexed experience memory. *arXiv preprint arXiv:2603.04257*, 2026.

Lei Wei, Xiao Peng, Tt, Guannan Zhang, Chenhao Jiang, Hongyu Li, Lanbo Lin, Yuanwu Xu, Jiayao Liu, Kesu Wang, and Bin Wang. Pace: Predictive adaptive context extraction for long-horizon llm agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27184–27199, 2026.

Ian Wu, Yuxiao Qu, Amrith Setlur, and Aviral Kumar. Reasoning cache: Continual improvement over long horizons via short-horizon rl. *arXiv preprint arXiv:2602.03773*, 2026a.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, et al. Resum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2025.

Yong Wu, YanZhao Zheng, TianZe Xu, ZhenTao Zhang, YuanQiang Yu, JiHuai Zhu, Chao Ma, BinBin Lin, BaoHua Dong, HangCheng Zhu, et al. Contextbudget: Budget-aware context management for long-horizon search agents. *arXiv preprint arXiv:2604.01664*, 2026b.

Binyan Xu, Xilin Dai, Fan Yang, and Kehuan Zhang. When agent automation becomes profitable: Quantifying and insuring autonomous ai risk through trace-economic underwriting. *arXiv preprint arXiv:2606.16465*, 2026a.

Binyan Xu, Xilin Dai, and Kehuan Zhang. Contextual agentic memory is a memo, not true memory. *arXiv preprint arXiv:2604.27707*, 2026b.

Binyan Xu, Fan Yang, Xilin Dai, Di Tang, and Kehuan Zhang. From internal diagnosis to external auditing: A vlm-driven paradigm for data-free online backdoor defense. In *Forty-third International Conference on Machine Learning*, 2026c.

Buqiang Xu, Zirui Xue, Dianmou Chen, Chenyang Fu, Chiyu Wu, Caiying Huang, Chen Jiang, Jizhan Fang, Xinle Deng, Yijun Chen, et al. Tokenpilot: Cache-efficient context management for llm agents. *arXiv preprint arXiv:2606.17016*, 2026d.

Ke Yang, Zixi Chen, Xuan He, Jize Jiang, Michel Galley, Chenglong Wang, Jianfeng Gao, Jiawei Han, and ChengXiang Zhai. PlugMem: A task-agnostic plugin memory module for llm agents, 2026. URL https://arxiv.org/abs/2603.03296.

Howard Yen, Ashwin Paranjape, Mengzhou Xia, Thejas Venkatesh, Jack Hessel, Danqi Chen, and Yuhao Zhang. Lost in the maze: Overcoming context limitations in long-horizon agentic search. *arXiv preprint arXiv:2510.18939*, 2025.

Lu Yi, Runlin Lei, Liuyi Yao, Yuexiang Xie, Yuyang Li, Wenhao Zhang, Zhewei Wei, Yaliang Li, and Jian-Yun Nie. Learning agent-compatible context management for long-horizon tasks. *arXiv preprint arXiv:2605.30785*, 2026.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. *arXiv preprint arXiv:2601.01885*, 2026.

Weihao Zeng, Yuzhen Huang, and Junxian He. LOCA-bench: Benchmarking language agents under controllable and extreme context growth. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

Haoxiang 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*, 2026a.

Haozhen Zhang, Haodong Yue, Tao Feng, Quanyu Long, Jianzhu Bao, Bowen Jin, Weizhi Zhang, Xiao Li, Jiaxuan You, Chengwei Qin, and Wenya Wang. Learning query-aware budget-tier routing for runtime agent memory, 2026b. URL https://arxiv.org/abs/2602.06025.

Yuxiang Zhang, Jiangming Shu, Ye Ma, Xueyuan Lin, Shangxi Wu, and Jitao Sang. Memory as action: Autonomous context curation for long-horizon agentic tasks. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 19149–19164, 2026c.

Yujie Zhao, Boqin Yuan, Junbo Huang, Haocheng Yuan, Zhongming Yu, Haozhou Xu, Lanxiang Hu, Abhilash Shankarampeta, Zimeng Huang, Wentao Ni, et al. AMA-bench: Evaluating long-horizon memory for agentic applications. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, Congming Zheng, Jiachen Zhu, Zeyu Zheng, Zhuosheng Zhang, Xingyu Lou, Changwang Zhang, Zhihui Fu, Jun Wang, Weiwen Liu, Jianghao Lin, and Weinan Zhang. Externalization in llm agents: A unified review of memory, skills, protocols and harness engineering, 2026. URL https://arxiv.org/abs/2604.08224.

A Recovery Capacity and Information-Limited Archiving

VISTA couples lossless recovery with an informative dashboard. The two resources solve different problems: recovery preserves evidence after eviction, whereas proprioception helps select what to evict. We formalize both with Fano's inequality (Sakai, 2020): reconstruction gives the recovery separation, and list decoding gives an information–archiving tradeoff. Each result uses a separate worst-case task family with explicit information and access assumptions. Channel calculations, auxiliary lemmas, and full proofs follow in Appendices B–E.

Recovery is necessary under budget pressure. In $\mathcal{T}_{N,k}$, the history contains N independent, uniformly random k-bit evidence blocks, but the retained state has entropy at most $B < Nk$ bits. A query index, uniform and independent of the blocks and retained state, is revealed after compression; the corresponding block must be reproduced exactly. A non-recovering decoder may use only its retained representation R and this index, with $H(R) \le B$ and no additional evidence channel. A recovering policy may reload the selected block after the reveal. In this proposition B is an information budget in bits, distinct from the token budget elsewhere in the method; a token constraint requires a separate encoding argument to imply an entropy constraint.

Proposition 1 (Recovery is necessary under budget pressure)**.** *For any non-recovering method whose pre-reveal in-prompt state R satisfies $H(R) \le B$,*

$$\Pr[\textit{correct on } \mathcal{T}_{N,k}] \le \min\left\{1, \frac{B}{Nk} + \frac{1}{k}\right\}.$$

A reliable lossless archive interface admits an ideal policy with success probability 1 whenever the instruction, the N handles, and one recovered block fit within B. There are feasible scaling regimes in which the non-recovering bound vanishes while this policy remains feasible. This is an existence result, not a guarantee for the stochastic LLM policy used by VISTA.

The bound captures delayed evidence demand: a lossy state cannot know which exact block will later matter, whereas archive handles preserve every block until the target is known.

Information can reduce future-relevant archiving. Recovery removes the losslessness barrier but not the *control* problem: the agent must still decide *which* blocks to move. We give both agents the same lossless archive and vary only how much state information identifies the best eviction target.

Definition 1 (Make-room instance $\mathcal{M}_{n,L,\ell}$)**.** The working set holds n blocks. A hidden index $J^\star$, uniform on $\{1, \dots, n\}$, marks one *bulky* block of size L; the other $n-1$ *load-bearing* blocks each have size $\ell < L$, with $\kappa := L/\ell \ge 2$ an integer. The prompt is over budget by L tokens. The bulky block is not needed again, but every load-bearing block is queried later; after each archive the agent observes only whether overflow remains. During this make-room phase it archives distinct blocks and makes no other queries or workspace modifications. Block content and metadata are available only through the observation Y in Definition 2; any additional side information must be included in Y. Sizes are net prompt tokens freed after inserting replacement handles. The definition does not require future answers to be inaccessible without a read of the original block.

To exit overflow the agent archives blocks in some order until the freed size reaches L. Let τ be the position of $J^\star$ in the complete counterfactual archive order (including its extension after stopping): archiving $J^\star$ frees L at once, while each load-bearing block frees only ℓ, so κ of them are needed otherwise. Hence the number of load-bearing blocks archived before overflow clears is

$$Z \;=\; \min(\tau - 1,\; \kappa). \tag{1}$$

Making room efficiently is therefore a localization problem: rank $J^\star$ early and avoid archiving blocks that will be queried later.

Definition 2 (Proprioceptive interface of rate I)**.** Before acting, the agent receives an observation Y containing all available information about the instance other than subsequent binary overflow feedback. It uses private randomness $U \perp (J^\star, Y)$; the counterfactual all-over construction in Appendix E represents a sequential policy by a (Y, U)-measurable archive order. The *rate* of the interface is the information it carries about the bulky block, $I := I(J^\star; Y)$ bits. A full size ledger identifies $J^\star$ and has $I = \log_2 n$. One restricted endpoint has $Y = W$, a specified noisy size channel, with no additional informative side information; Proposition 2 in Appendix B bounds its rate. The theorem otherwise applies to arbitrary Y through $I(J^\star; Y)$. A small rate for W alone does not

bound the rate of (W, C) when content C is also visible. Complete block contents, boundaries, and an exact tokenizer can reveal sizes even without a dashboard.

Theorem 1 (Information–archiving tradeoff). *Assume $2 \le \kappa < n/2$ and let $\nu := \log_2 \frac{n-\kappa}{\kappa} > 0$. On $\mathcal{M}_{n,L,\ell}$ under an interface of rate I (Def. 2), the expected number Z of archived load-bearing blocks obeys*

$$\mathbb{E}[Z] \;\ge\; \kappa \left[1 - \frac{I+1}{\nu}\right]_+ .$$

Here $[x]_+ := \max\{x, 0\}$. The bound concerns archived future-relevant blocks; it does not by itself lower-bound subsequent reads or answer errors. The full ledger identifies $J^\star$ and attains $Z = 0$. At the other endpoint, any low-rate interface satisfying $(I+1)/\nu \to 0$ incurs $\mathbb{E}[Z]/\kappa \to 1$. For any fixed $\delta \in (0,1)$, $\mathbb{E}[Z] \le \delta\kappa$ requires $I \ge (1-\delta)\nu - 1$ bits, and the balanced-bucket construction in Appendix E gives a sufficient rate of the same order when $n/\kappa \to \infty$. This is an existence result for a designed interface, not a guarantee for every interface with that rate.

A recovery interpretation is available under an additional atomic paging assumption: a future request is served only by accessing its original block; an archived block can be accessed only by restoring it, and summaries, replicas, guessing, recomputation, and other tools cannot substitute for that access. Under this assumption, serving all later requests requires restoring all Z archived future-relevant blocks. One grouped read may restore several blocks. This assumption is an idealized access model and is not asserted for VISTA's general question-answering tasks, where summaries or partial reads may suffice. The observed archive/retrieve event counts are therefore only qualitatively consistent with the proposed mechanism, not a direct measurement of this theorem's block-level quantity. The appendix gives content-channel calculations, finite-sample corollaries, and an achieving policy.

B Technical Setup for the Information–Archiving Analysis

The main text states the two separations and their operational interpretation. Here we record the content-derived endpoint and the auxiliary list-decoding inequality used by Theorem 1.

For the restricted noisy-observation endpoint of Definition 2, let the agent's entire initial observation be $Y = W$, where it observes $W_j = \ln s_j + \varepsilon_j$ for every block, with independent $\varepsilon_j \sim \mathcal{N}(0, \sigma^2)$ and $\sigma > 0$. This is an analyzable model of noisy size perception, not an estimate fitted from the empirical probe in Appendix K. No raw-content or other informative side channel is available in this endpoint. More generally, additional state S is harmless for this bound only if $I(J^\star; S \mid W) = 0$; otherwise the relevant rate is $I(J^\star; W, S)$, which need not obey the stated upper bound.

Proposition 2 (Information in the specified Gaussian size channel). *For the channel above,*

$$I_{\text{content}} = I(J^\star; W) \le \frac{(\log_2 \kappa)^2}{\sigma^2} \ln 2 \quad \textit{bits}.$$

Thus for fixed (κ, σ), information in this channel is bounded independently of the number of blocks n, whereas an exact size ledger can identify one block among n.

Lemma 1 (Fano, list-decoding form). *Let $J^\star$ be uniform on $\{1, \dots, n\}$ and let $\mathcal{L}(Y, U)$ be any (Y, U)-measurable list of size $1 \le m \le n-1$, where $U \perp (J^\star, Y)$. With $I = I(J^\star; Y)$ and $P_c = \Pr[J^\star \in \mathcal{L}]$,*

$$\log_2 n - I \le 1 + P_c \log_2 m + (1 - P_c) \log_2 (n - m).$$

C Proof of Proposition 1

This is Fano's inequality in *reconstruction* mode: a budget-limited state cannot reproduce more blocks than it has bits for. We restate the setting. The history holds N blocks $X_1, \dots, X_N$, each an independent string of k uniformly random bits, so $H(X_i) = k$ and the X_i are mutually independent. A non-recovering method holds a pre-reveal in-prompt state R with $H(R) \le B$. The state R is a function of the blocks and the method's internal randomness, formed before the query index $i^\star$ is revealed, and $i^\star$ is drawn uniformly on $\{1, \dots, N\}$ independently of everything else. After the reveal the method outputs a guess $g(R, i^\star)$, and it is correct when $g(R, i^\star) = X_{i^\star}$.

Let $P_e^{(i)} = \Pr[g(R, i) \ne X_i]$ and let the reported success probability be $1 - P_e$ with $P_e = \frac{1}{N} \sum_{i=1}^{N} P_e^{(i)}$, the average over the uniform $i^\star$.

Step 1: per-block Fano bound. Fix a block i. Since X_i is uniform on an alphabet of size 2^k, Fano's inequality applied to the estimator $g(R, i)$ gives

$$H(X_i \mid R) \;\le\; H_b\Big(P_e^{(i)}\Big) + P_e^{(i)} \log_2(2^k - 1) \;\le\; 1 + P_e^{(i)}\, k,$$

where H_b is the binary entropy function, bounded by 1.

Step 2: independence couples the blocks to a budget. Because the X_i are mutually independent, $H(X_{1:N}) = \sum_i H(X_i)$, and subadditivity of conditional entropy gives $H(X_{1:N} \mid R) \le \sum_i H(X_i \mid R)$. Hence

$$\begin{aligned}\sum_{i=1}^{N} I(X_i; R) &= \sum_i \big(H(X_i) - H(X_i \mid R)\big) \\ &\le H(X_{1:N}) - H(X_{1:N} \mid R) \\ &= I(X_{1:N}; R) \;\le\; H(R) \;\le\; B.\end{aligned}$$

Step 3: combine. Using $I(X_i; R) = k - H(X_i \mid R) \ge k - 1 - P_e^{(i)} k$ from Step 1 and summing,

$$\begin{aligned}B &\ge \sum_{i=1}^{N} I(X_i; R) \\ &\ge \sum_{i=1}^{N} \big(k - 1 - P_e^{(i)} k\big) = Nk - N - k \sum_{i=1}^{N} P_e^{(i)}.\end{aligned}$$

Dividing by Nk and using $P_e = \frac{1}{N}\sum_i P_e^{(i)}$,

$$\begin{aligned}P_e &\ge 1 - \frac{1}{k} - \frac{B}{Nk}, \\ \Pr[\text{correct}] = 1 - P_e &\le \frac{B}{Nk} + \frac{1}{k}.\end{aligned}$$

Existence of a probability-one recovery policy. Assume reliable external storage and an ideal controller. The controller writes each block to storage exactly and retains an indexed handle table using $O(N \log N)$ bits. Once $i^\star$ is revealed, it reads the indexed payload and copies $X_{i^\star}$ exactly. If the instruction, handles, and recovered block fit within B, this policy succeeds with probability 1. The interface thus admits such a policy; an actual stochastic LLM need not execute it correctly.

Asymptotic separation. The gap statement requires a regime in which VISTA *stays feasible* while the lossy bound vanishes. Here feasibility and probability-one success refer to the ideal controller just constructed. Feasibility needs the N handles plus one recovered block to fit, i.e. $B \ge c_0 N \log_2 N + k$ for the constant c_0 set by the handle encoding; the lossy bound $\frac{B}{Nk} + \frac{1}{k}$ vanishes when $Nk/B \to \infty$ and $k \to \infty$. Both hold, for example, at $k = \lceil \sqrt{N} \rceil$ and $B = cN \log_2 N$ with $c > c_0$: then B dominates the handle cost, so the ideal recovery policy is feasible and succeeds with probability 1, while $\frac{B}{Nk} + \frac{1}{k} = \Theta\big(\frac{\log N}{\sqrt{N}}\big) \to 0$, so the lossy success probability tends to 0 and the gap tends to 1 as $N \to \infty$. A *fixed* B does not exhibit this separation, because then VISTA's own $O(N \log N)$ handle table eventually violates the budget. The separation concerns the growth of raw evidence Nk relative to a budget B that grows only fast enough to index it. □

D Proof of Proposition 2

We analyze the restricted noisy-observation endpoint of Def. 2 for $\sigma > 0$. Conditioned on $J^\star = j$, the observation $W = (W_1, \ldots, W_n)$ is Gaussian with independent coordinates, $W_i \sim \mathcal{N}(m_i^{(j)}, \sigma^2)$, where the mean vector $m^{(j)}$ has $m_j^{(j)} = \ln L$ (the bulky block) and $m_i^{(j)} = \ln \ell$ for $i \ne j$. Let P_j denote this conditional law and $\bar{P} = \frac{1}{n}\sum_k P_k$ the mixture.

Step 1: mutual information as mixture KL. Writing mutual information in bits and KL with natural logarithms, for $J^\star$ uniform a standard identity gives

$$I(J^\star; W) \;=\; \frac{1}{n \ln 2} \sum_{j=1}^{n} \mathrm{KL}\big(P_j \,\|\, \bar{P}\big)\,.$$

Because $\mathrm{KL}(P \,\|\, \cdot)$ is convex in its second argument and $\bar{P} = \frac{1}{n}\sum_k P_k$, Jensen gives $\mathrm{KL}(P_j \| \bar{P}) \le \frac{1}{n}\sum_k \mathrm{KL}(P_j \| P_k)$, hence

$$I(J^\star; W) \;\le\; \frac{1}{n^2 \ln 2} \sum_{j=1}^{n} \sum_{k=1}^{n} \mathrm{KL}(P_j \,\|\, P_k)\,.$$

Step 2: pairwise Gaussian KL. P_j and P_k are Gaussians with the same covariance $\sigma^2\mathbf{I}$, so $\mathrm{KL}(P_j \| P_k) = \frac{1}{2\sigma^2}\|m^{(j)} - m^{(k)}\|_2^2$. For $j \neq k$ the mean difference is nonzero only in coordinates j and k: coordinate j contributes $\ln L - \ln \ell = \ln \kappa$ and coordinate k contributes $\ln \ell - \ln L = -\ln \kappa$, so $\|m^{(j)} - m^{(k)}\|_2^2 = 2(\ln \kappa)^2$ and

$$\mathrm{KL}(P_j \| P_k) = \frac{(\ln \kappa)^2}{\sigma^2} \quad (j \neq k), \qquad \mathrm{KL}(P_j \| P_j) = 0.$$

Step 3: combine. There are $n(n-1)$ ordered pairs with $j \neq k$, so

$$I(J^\star; W) \;\le\; \frac{1}{n^2 \ln 2}\, n(n-1)\, \frac{(\ln \kappa)^2}{\sigma^2} \;\le\; \frac{(\ln \kappa)^2}{\sigma^2 \ln 2} \text{ bits } = \; \frac{(\log_2 \kappa)^2}{\sigma^2}\, \ln 2 \text{ bits},$$

the last equality using $\ln \kappa = (\ln 2)\log_2 \kappa$. The bound depends only on the log size-ratio $\log_2 \kappa$ and the perception noise σ. For fixed κ and σ, it is constant in the workspace size n, while the full ledger carries $\log_2 n$ bits. More generally, Theorem 1 places this content channel on the costly side of the make-room threshold whenever $(I_{\text{content}} + 1)/\log_2(n/\kappa) \to 0$. □

E Proof of Theorem 1

The proof has five parts: (A) the list-Fano lemma; (B) a reduction that turns the make-room archiving cost into a list-decoding error, converting the information budget I into a lower bound on $\mathbb{E}[Z]$; (C) the closed-form tradeoff and its corollaries (endpoints, threshold, price); (D) an order-wise achievability construction; and (E) the reduction from adaptive archive policies to ordered probing. Throughout, $J^\star$ is uniform on $[n] := \{1, \dots, n\}$, $\kappa = L/\ell$ is an integer satisfying $2 \le \kappa < n/2$, and $\nu := \log_2 \frac{n-\kappa}{\kappa} > 0$.

(A) List-Fano lemma (proof of Lemma 1). Let $\mathcal{L} = \mathcal{L}(Y, U)$ be a list of size m and $E := \mathbf{1}[J^\star \notin \mathcal{L}]$, so $\Pr[E{=}0] = P_c$. Expand $H(J^\star, E \mid Y, U)$ two ways. Since E is a function of $(J^\star, Y, U)$, $H(J^\star, E \mid Y, U) = H(J^\star \mid Y, U)$. Also

$$H(J^\star, E \mid Y, U) = H(E \mid Y, U) + H(J^\star \mid E, Y, U) \le 1 + H(J^\star \mid E, Y, U).$$

Condition on E: given $E{=}0$, $J^\star$ lies in a set of size $\le m$, so $H(J^\star \mid E{=}0, Y, U) \le \log_2 m$; given $E{=}1$, $J^\star$ lies in the complement of size $\le n - m$ (recall $\mathcal{L}$ is (Y, U)-measurable), so $H(J^\star \mid E{=}1, Y, U) \le \log_2(n - m)$. Averaging with weights $P_c, 1 - P_c$ and using $H(J^\star \mid Y, U) = H(J^\star) - I(J^\star; Y, U) = \log_2 n - I$ (as $U \perp (J^\star, Y)$, $I(J^\star; Y, U) = I(J^\star; Y) = I$),

$$\log_2 n - I \;\le\; 1 + P_c \log_2 m + (1 - P_c)\log_2(n - m),$$

which is Lemma 1. □

(B) Reduction: archiving cost lower-bounds via list decoding. By the "still-over" argument (part (E) below), it suffices to prove the lower bound for the more powerful class of sequential policies. Such a policy archives blocks in an order $\pi = (\pi_1, \pi_2, \dots)$ measurable in (Y, U) and stops at the first time the freed size reaches L. Because the bulky block frees L by itself while each load-bearing block frees only ℓ, the process stops exactly at step $T = \min(\tau, \kappa)$ where τ is the position of

$J^\star$ in π: if $\tau \le \kappa$ it captures $J^\star$ and frees L; if $\tau > \kappa$ it has archived κ load-bearing blocks, freeing $\kappa\ell = L$, before reaching $J^\star$. The number of archived load-bearing blocks is (1), $Z = \min(\tau - 1, \kappa)$.

Fix any integer $1 \le m \le \kappa$ and let $\mathcal{L}_m := \{\pi_1, \dots, \pi_m\}$, the first m probed blocks; this is a size-m, (Y, U)-measurable list. The key observation is

$$Z \ge m \iff \tau > m \iff J^\star \notin \mathcal{L}_m, \tag{2}$$

because $Z = \min(\tau - 1, \kappa) \ge m$ iff $\tau - 1 \ge m$ (using $m \le \kappa$) iff $J^\star$ is not among the first m probes. Hence, writing $P_c(m) := \Pr[J^\star \in \mathcal{L}_m]$, we have $\Pr[Z \ge m] = 1 - P_c(m)$.

Expectation via tail sum. Since $Z \in \{0, 1, \dots, \kappa\}$,

$$\mathbb{E}[Z] = \sum_{m=1}^{\kappa} \Pr[Z \ge m] = \sum_{m=1}^{\kappa} \big(1 - P_c(m)\big) = \kappa - \sum_{m=1}^{\kappa} P_c(m). \tag{3}$$

One-shot list bound. Apply Lemma 1 to the size-κ list $\mathcal{L}_\kappa$. With $m = \kappa$ and $n - \kappa > \kappa$ (so $\nu = \log_2 \frac{n-\kappa}{\kappa} > 0$),

$$\log_2 n - I \le 1 + P_c(\kappa) \log_2 \kappa + (1 - P_c(\kappa)) \log_2 (n - \kappa).$$

Rearranging, and using $\log_2 n \ge \log_2 (n - \kappa)$,

$$-(I + 1) \;\le\; \big(\log_2 (n - \kappa) - \log_2 n\big) + P_c(\kappa)\big(\log_2 \kappa - \log_2 (n - \kappa)\big) \;\le\; -P_c(\kappa)\, \nu,$$

so $P_c(\kappa) \le \frac{I+1}{\nu}$. Because $\mathcal{L}_m \subseteq \mathcal{L}_\kappa$ for $m \le \kappa$, $P_c(m) \le P_c(\kappa) \le \frac{I+1}{\nu}$, and plugging into (3),

$$\boxed{\mathbb{E}[Z] \;\ge\; \kappa \left[1 - \frac{I+1}{\nu}\right]_+ .} \tag{4}$$

Here the positive part combines the derived inequality with $Z \ge 0$. This is the bound in Theorem 1.

(C) Corollaries. *(i) Endpoints.* The size-aware ledger reveals the size vector, which identifies the unique block of size L, so $I = \log_2 n$; then the agent archives $\{J^\star\}$, freeing exactly L, giving $\tau = 1$ and $Z = 0$. This entails no later obligations induced by archiving load-bearing blocks; it is not a guarantee about an LLM's later answers. The restricted noisy-channel agent has $I = I_{\text{content}}$. Whenever $(I_{\text{content}} + 1)/\nu \to 0$, (4) gives $\mathbb{E}[Z]/\kappa \to 1$; fixed κ and σ with $n \to \infty$ is one such regime by Proposition 2. The pure size-blind case $I = 0$ is realized by an observation independent of $J^\star$ (and approached by the Gaussian channel as $\sigma \to \infty$ for fixed n, κ).

(ii) Threshold. $\mathbb{E}[Z] \le \delta\kappa$ forces, via (4), $\kappa(1 - \frac{I+1}{\nu}) \le \delta\kappa$, i.e. $I \ge (1 - \delta)\nu - 1$.

(iii) Price. Before taking the positive part, the right-hand floor $\kappa(1 - \frac{I+1}{\nu})$ is affine in I with slope $-\kappa/\nu$. Thus the lower-bound floor changes at rate κ/ν per bit. In particular, requiring $\mathbb{E}[Z] \le \kappa - \Delta$ forces $I \ge \Delta\nu/\kappa - 1$ bits.

Conditional recovery interpretation. Under the additional atomic paging assumption stated after Theorem 1, each of the Z archived load-bearing blocks must be restored to serve its later request. If at most r distinct such blocks can be restored, at least $(Z - r)_+$ requests, one per unserved block, remain unserved; hence their expected number is at least $\mathbb{E}[(Z - r)_+] \ge [\mathbb{E}Z - r]_+$. This is an access-service statement under the paging model, not a deterministic exact-answer error bound for an unrestricted decoder. In particular, queries answerable from summaries or known strings need not require any restoration. No answer-error lower bound is claimed without an additional conditional unpredictability assumption.

(D) Achievability (order-wise upper bound). Given a rate budget I, let $M := \min\{n, \lfloor 2^I \rfloor\}$ and consider a balanced M-bucket ledger: partition $[n]$ into M buckets, each of size at most $b := \lceil n/M \rceil$, and report the bucket Y containing $J^\star$. Since Y is a deterministic function of $J^\star$, $I(J^\star; Y) = H(Y) \le \log_2 M \le I$, so this is a rate-$\le I$ interface. The agent probes blocks inside the reported bucket in uniformly random order until the flag clears. Conditional on a bucket of size $b' \le b$, the bulky block sits at a uniform position $P \in \{1, \dots, b'\}$. As in part (E), it frees L upon capture, so the number of load-bearing blocks archived is $\min(P - 1, \kappa) \le P - 1$, giving

$$\mathbb{E}[Z \mid Y] \;\le\; \mathbb{E}[P - 1 \mid Y] = \frac{b' - 1}{2} \;\le\; \frac{\lceil n/M \rceil - 1}{2}.$$

The last bound holds for every bucket, and therefore also holds after averaging over Y. Because $\lceil x \rceil - 1 < x$, $\mathbb{E}[Z] < n/(2M)$. Therefore $I \geq \min\{\log_2 n, \log_2 \lceil n/(2\delta\kappa) \rceil\}$ suffices for $\mathbb{E}[Z] \leq \delta\kappa$. At $I = \log_2 n$, we have $M = n$, hence singleton buckets and $Z = 0$. Define the minimum required rate by optimizing over both observation channels and admissible policies. For fixed $\delta \in (0, 1)$ in the regime $n/\kappa \to \infty$, the necessary and sufficient bounds show that this minimum is $\Theta(\log_2(n/\kappa))$. They do not determine a sharp critical constant for fixed δ, and equal mutual information does not ensure equal performance for different channels. If additionally $\delta_n \to 0$ and $\log_2(1/\delta_n) = o(\nu)$, both bounds are $(1 + o(1))\nu$.

The divergence condition matters. With $I = 0$, the position τ is uniform, so the tail-sum identity gives

$$\mathbb{E}Z = \sum_{m=1}^{\kappa} (1 - m/n) = \kappa - \frac{\kappa(\kappa + 1)}{2n}.$$

For $n = 2\kappa + 1$ and $\kappa \to \infty$, $\mathbb{E}Z/\kappa \to 3/4$; thus a target such as 0.8κ already needs no information in that regime.

(E) Counterfactual all-over reduction for adaptive policies. For a sequential policy, fix (Y, U), including its entire random seed, and simulate its choices along a counterfactual history in which every archive is followed by "over." The policy selects distinct remaining blocks until this simulation has produced κ entries; define arbitrary measurable choices on unreachable histories if necessary, then append the remaining indices in increasing order. This produces a complete (Y, U)-measurable permutation π. In the actual run, every response before stopping is "over," so its choices coincide with this permutation up to $T = \min(\tau, \kappa)$. Thus $Z = \min(\tau - 1, \kappa)$ and all lists used in part (B) are (Y, U)-measurable. The feedback can exclude already selected indices; the argument does not assert that it is independent of $J^\star$. It establishes the prefix representation for adaptive choices.

For a batch policy, fix an order within each selected batch and simulate its blocks sequentially, stopping immediately when overflow clears. When an entire batch leaves overflow, pass that response to the original policy to choose the next batch. This refinement archives no more load-bearing blocks on any instance. Therefore the sequential lower bound also lower-bounds the actual batch cost, even though the equality defining Z need not hold for a batch that overshoots the first clearing point. □

Remark 1 (Interpretation and empirical scope). Theorem 1 gives an order-wise information requirement for reducing future-relevant archiving when $n/\kappa \to \infty$. Recovery obligations require the separate atomic paging assumption. The affine lower-bound floor has slope $-\kappa/\nu$; this is not the derivative of an established optimal tradeoff. For the restricted channel $Y = W$, Proposition 2 gives a rate bounded independently of n at fixed κ, σ. This does not establish low Shannon information for an LLM that also observes complete block contents. Difficulty computing a length differs from absence of information in the input.

The size probes in Appendix K measure performance of particular decoders, not an upper bound on $I(J^\star; Y)$. A median total-size error of zero does not imply every estimate is exact. The empirical pairwise probe compares blocks within a factor of two, whereas the theorem has one bulky block with ratio $\kappa \geq 2$ and equal-sized small blocks; these are different statistical experiments. Finally, the $255/57$ versus $69/105$ archive/successful-recovery figures count events, not distinct future-relevant blocks. The separate no-recovery condition records zero successful recoveries and 57 failed attempts; these attempts are not counted as recovered events. Batching and repeated reads prevent a direct conversion. The theorem also does not predict that a low-information policy will issue fewer actual reads: a policy may stop early, time out, or fail to serve later requests. Those data are qualitatively consistent with the proposed mechanism, but do not test the quantitative lower bound without block-level access and query logs.

F Method Capability Comparison

Table 4 summarizes the evaluated baselines and the learned-compression family against the design properties of Section 3. It isolates the two properties that distinguish VISTA: an agent-facing context dashboard and byte-exact recovery of externalized evidence. The marks are a coarse capability summary, not a performance claim, and partial marks reflect mechanisms that hold the property only in part, such as Claude Code, which keeps files on disk but still summarizes the active context.

Table 4: Context-management capabilities. The matrix compares the interfaces used by the listed methods: agent control, recoverable payloads, and explicit state metadata. VISTA combines all three in a training-free interface. A tilde denotes partial support; the learned-method rows describe their published configurations.

Method	Training-free	Model-agnostic	Agent-controlled	Exact recovery	Context dashboard
ReAct (append until truncation)	✓	✓	✗	✗	✗
Tool-result clearing	✓	✓	✗	✗	✗
Stale-observation masking	✓	✓	✗	✗	✗
Active Context Compression	✓	✓	✓	✗	✗
Skeleton compression	✓	✓	✗	✗	✗
Claude Code	✓	∼	✓	∼	✗
Learned compression (CAT, RL budget)	✗	✗	✓	✗	✗
Context-Folding	✗	✗	✓	✗	✗
LongSeeker	✗	✗	✓	✗	✗
GenericAgent	✗	✗	✓	✗	✗
VISTA (ours)	✓	✓	✓	✓	✓

G Workspace Contract and Implementation

VISTA treats context management as a meta-tool decision over the agent's own working memory. Figure 3 shows the three-stage loop: a context stream, a refreshed dashboard, and archive/recovery tools. The goal is to make context state perceptible to an unmodified model without fine-tuning, model-specific changes, or destroyed evidence.

G.1 Problem Setup

At step t, a tool agent has a task goal g, raw interaction history H_t, environment tools $\mathcal{T}_{\text{env}}$, and a context budget B. We write the history as action-observation pairs $H_t = (a_1, o_1, \ldots, a_{t-1}, o_{t-1})$. A standard ReAct-style harness serializes this append-only history into the next model input. Once the serialized history exceeds B, the harness must truncate, clear, mask, or summarize prior content. These interventions conflate what remains visible, what is preserved exactly, and what can be recovered later.

VISTA separates these choices through a workspace $W_t = (V_t, A_t)$ of visible blocks and recoverable archived payloads. The model acts on the workspace rendering

$$\begin{aligned} \widetilde{C}_t &= \text{render}(V_t) \oplus \text{handles}(A_t) \oplus D_t, \\ C_t &= \text{preflight}(\widetilde{C}_t, B), \qquad a_t \sim \pi_\theta(\cdot \mid C_t), \qquad |C_t| \le B. \end{aligned} \tag{5}$$

Here $|C_t|$ denotes the harness's token-length estimate, including required system instructions, tool schemas, and protocol overhead in the assembled input. The inequality holds when preflight returns a feasible input under this metric. The provider's physical context limit H imposes a separate request constraint, $\text{tokens}_{\text{provider}}(C_t) + r_{\text{out}} \le H$, where r_{out} is the reserved output allowance. Reproduction requires the provider tokenizer, output reserve, and rejection policy. Here D_t is the dashboard: a factual ledger of block IDs, token estimates, recency, block type, archive level, status, and remaining budget. The candidate $\widetilde{C}_t$ may exceed the budget. In that case, preflight attempts to construct a budget-safe management rendering C_t with the dashboard, actionable handles, and compact stubs while disabling ordinary task tools; otherwise it returns the ordinary rendering. This contract requires the minimum management input, including pinned content and required metadata, to fit within B. If no such input fits, preflight must return an explicit budget-infeasible outcome without issuing a model call. A growing ledger must be bounded, paged, or hierarchically represented within this input cap. The raw transcript is retained in the evaluation log.

Workspace invariants. The interface contract requires $|C_t| \le B$ after preflight and final assembly, stable IDs or handles for actionable blocks, and recoverable archived bytes until explicit deletion. The dashboard and guard use the same token estimates. Provider-level safety additionally requires the provider tokenizer or a conservative error bound. Step and time limits handle invalid or non-progressing actions. Implementation checks appear in Appendix M.

G.2 Context Stream

The left panel of Figure 3 shows how VISTA rewrites the transcript as a block stream. The agent-facing state has two storage regimes:

(i) VISIBLE blocks appear in the active prompt with exact content; pinned blocks are a protocol-required subset of this state.

(ii) ARCHIVED blocks are replaced by compact handles and summaries, while their original bytes are stored externally.

The delete action permanently removes selected content. A separate hard-budget guard may replace already-visible raw tool results with recoverable offloaded placeholders; a new tool result that still cannot fit is rejected with a compact notice and can be rerun after context cleanup or with a narrower query. These safeguards are distinct from the agent's archive decision.

This stream gives the agent an address space for context decisions. When a bundle of evidence becomes too large, the stream keeps a compact handle while the exact payload remains in the hidden trajectory. New observations are admitted only when the resulting request remains budget-safe.

Structure preservation. The visible stream is the next-call working set, while the hidden trajectory stores exact payloads that may be needed later. Tool results remain linked to their assistant tool calls, and archived results are rendered in protocol-valid form. If a parent call is also archived, the placeholder becomes ordinary context instead of an orphaned tool response.

G.3 LLM Policy

Dashboard input. The middle panel shows what the model receives: the visible context stream plus a dashboard. The dashboard is regenerated after every tool result is registered, so the agent acts on current workspace state. Its fields reflect registered blocks and prior workspace actions.

Unified action space. The same model policy chooses ordinary environment actions and context actions:

$$\begin{aligned} \mathcal{T}_{\mathrm{ctx}} &= \{\mathrm{archive}(\mathcal{S}, \rho),\ \mathrm{delete}(\mathcal{S})\}, \\ a_t &\in \mathcal{T}_{\mathrm{env}} \cup \mathcal{T}_{\mathrm{ctx}} \cup \{\mathrm{answer}\}. \end{aligned} \tag{6}$$

Here $\mathcal{T}_{\mathrm{ctx}}$ contains the two dedicated workspace tools. Recovery uses an ordinary file or terminal read from the archive path and is therefore an environment action, which we denote abstractly by $\mathrm{read_path}(h, q) \in \mathcal{T}_{\mathrm{env}}$. Context management is part of the model's action space, conditioned on the dashboard and task evidence visible in C_t.

Mode switch. The dashboard supplies the state used to select visible blocks and archived payloads. Block sizes follow the harness's declared token metric. In normal mode, the agent may call environment tools, continue the task, archive blocks, or read archived payloads. In overflow mode, ordinary tool calls are disabled until the agent reduces the visible context. The allowed action set is therefore

$$\mathcal{A}_t = \begin{cases} \mathcal{T}_{\mathrm{env}} \cup \mathcal{T}_{\mathrm{ctx}} \cup \{\mathrm{answer}\}, & |\widetilde{C}_t| \le B, \\ \mathcal{T}_{\mathrm{ctx}}, & |\widetilde{C}_t| > B. \end{cases} \tag{7}$$

Explicit archive calls are chosen by the agent; automatic guard actions are separate runtime interventions, so the evaluated system combines both. A final preflight guard can offload already-visible raw tool-result blocks near the hard limit; if a newly returned result still cannot be admitted, the harness rejects that result instead of silently truncating it. The complete loop is given in Appendix H.1. Task accuracy measures this combined agent–runtime system. Guard intervention statistics are pending in the implementation record (Appendix M).

G.4 Meta Context Tool

Archive interface. The right panel shows the meta-context tool. Archiving takes a selected block set $\mathcal{S}$ and a short replacement summary ρ:

$$\begin{aligned} h &= \mathrm{archive}(\mathcal{S}, \rho), \\ \mathrm{read_path}(h, \varnothing) &\equiv \mathrm{payload}(\mathcal{S}). \end{aligned} \tag{8}$$

Algorithm 1 Self-managed context loop. Each admitted request satisfies the workspace cap; archive handles preserve exact payload access across context edits.

Require: task block b_{task}, budget B, environment tools $\mathcal{T}_{\text{env}}$, context tools $\mathcal{T}_{\text{ctx}}$
Ensure: final answer or explicit failure status, and workspace trajectory $W_{1:T}$
$V_0 \leftarrow \{b_{\text{task}}\}$; $A_0 \leftarrow \varnothing$
while the task is unfinished **do**
  Apply the budget guard: admit fitting results; offload eligible visible raw results or reject an unadmittable new result
  $D_t \leftarrow \text{Dashboard}(V_t, A_t, B)$; $\widetilde{C}_t \leftarrow \text{Assemble}(V_t, A_t) \cup \{D_t\}$
  $C_t \leftarrow \text{preflight}(\widetilde{C}_t, B)$; if the minimum valid management input cannot fit, **return** budget-infeasible
  Check the final input length under the declared budget metric; if it exceeds B, **return** budget-infeasible before any model call
  Use $\mathcal{T}_{\text{ctx}}$ only if $|\widetilde{C}_t| > B$; otherwise use $\mathcal{T}_{\text{env}} \cup \mathcal{T}_{\text{ctx}} \cup \{\text{answer}\}$
  $a_t \leftarrow \text{LLM}(C_t)$
  If $a_t = \text{read_path}(h, q)$, read the requested exact bytes and admit the returned content subject to the same budget guard
  Else if $a_t = \text{archive}(\mathcal{S}, \rho)$, replace $\mathcal{S}$ by handles and store exact payloads in A_t
  Else if $a_t = \text{delete}(\mathcal{S})$, remove $\mathcal{S}$ and its stored payloads
  Else if $a_t \in \mathcal{T}_{\text{env}}$, execute it; if $a_t = \text{answer}$, **return** the answer
  $W_{t+1} \leftarrow (V_t, A_t)$
end while

The first line creates a compact handle in the visible stream; the second states the lossless contract using an ordinary environment read, with no third context tool. The exact payload moves to the external archive, while h keeps the path, level, size, and checksum metadata visible, leaving a natural boundary for external auditing (Xu et al., 2026c). The optional selector q requests an exact range or subset of that payload; $q = \varnothing$ requests the full payload. Recovery is performed through ordinary file or terminal access to the stored payload path. There is no task-specific retrieval oracle. For large payloads, the agent may read bounded chunks or rerun the source tool with narrower arguments.

The stored payload is a transcript of what the model saw, not a guarantee that the source was complete. If a source response was paginated or truncated, the archive preserves that result exactly and leaves re-querying to the agent.

Hierarchical recovery. Archiving is hierarchical. A first archive level may group several raw blocks into a bundle. Later, groups can themselves be archived into coarser handles when context pressure grows. The visible stream therefore stores a small summary and retrieval guide, while the hidden trajectory stores exact evidence. Summaries guide navigation; archives retain original evidence.

Recovery follows the hierarchy in reverse. The agent may inspect a coarse handle, recover the payload, and then decide whether a narrower piece of evidence should return to active context. It need not reload a long transcript when one row or identifier is needed. It can recover the file, search or read a bounded part, and continue with a smaller block.

H Implementation Details

This section reports the supplied run configuration, prompt, dashboard format, and context tool definitions. These describe the interface; full reproduction also requires the implementation details listed in Appendix M, especially for RL and infeasible budgets.

H.1 End-to-End Context Loop

Algorithm 1 expands the compact loop in Section 3. The recoverable store A_t includes agent-archived payloads and any raw tool result externalized by the final wire-payload guard; explicit deletion removes a payload from this store. The pseudocode states the safety contract, including an explicit infeasibility branch. Harness conformance checks are listed in Appendix M. Step and time limits ensure termination when actions are invalid or fail to reduce the working set.

Benchmark setting. LOCA-Bench evaluates online tool agents under controllable context growth: the agent must continue acting while earlier reasoning, tool calls, and observations remain in or are externalized from the working context. We evaluate 75 task configurations; unless otherwise stated, accuracy is solved tasks over all 75, with errors and timeouts counted as failures. AMA-Bench is used as a secondary generalization benchmark. Its episodes provide a completed trajectory and ask questions about past events, causal relations, and state changes. This is not the native online-control setting for VISTA; it tests whether the same context-management layer can be adapted into trajectory memory. We use the benchmark's two-stage memory interface. During memory construction, the completed trajectory is replayed step by step as a growing conversation: each action and observation becomes a workspace block, the future questions are hidden, and the replayed agent may archive exact payloads when the workspace budget becomes tight. During retrieval, VISTA assembles the resulting workspace, dashboard, construction events, and recoverable archive handles for the current question. Thus the AMA-Bench result evaluates a replayed VISTA workspace as offline memory (Xu et al., 2026b), instead of simply placing the full trajectory in the model prompt.

Run configuration. VISTA is integrated into the LOCA-Bench harness and invoked as a strategy (`loca run -s self_managed`) with no training and no per-model tuning. The main results use `gemini-3-flash` at a 128K budget (`max-context-size` = 128,000); the cross-backbone runs reuse the same strategy unchanged on `claude-sonnet-4-5`, `deepseek-v4-pro` (open-weight), and `glm-5` (open-weight). Two flags define the full method: `SM_STRICT_LONG_CONTEXT=1` enforces a hard budget instead of issuing a soft warning, and `SM_BETTER_DASHBOARD=1` selects the factual ledger dashboard below. The ablations toggle single flags from this base, for example `SM_DISABLE_ARCHIVE`, `SM_DISABLE_AGENT_ARCHIVE` (fixed archive policy), and `SM_ENABLE_STATE_BOARD` (status-board variant). All backbones use a 1800-second task timeout and medium reasoning effort.

Baseline definitions. We organize the LOCA-Bench baselines by who makes the keep-or-drop decision. Fixed external policies include ReAct, which appends until truncation; Tool-result Clearing, which removes old tool-result/tool-call pairs after the prompt crosses a threshold; and fixed stale masking, which masks old tool observations while preserving the assistant reasoning and tool-call skeleton. Agent-mediated baselines still reduce context irreversibly. SLIM (Yen et al., 2025), reproduced from its public release, periodically summarizes older context once the budget is exceeded. Active Context Compression (Verma, 2026) asks the agent to write and prune its own knowledge blocks. A structured-compression baseline preserves a compact skeleton of prior context, following the design of context-as-a-tool compressors (Liu et al., 2025). Learned members of this family generally do not release trained checkpoints and, in many cases, do not release code, so we reproduce the training-free methods directly and follow the published inference-time design for the rest without introducing a trained policy. Claude Code is the Claude Code command-line agent at the CLI release of May 6, 2026, included as a strong practical agent with mature tool-use and context-handling heuristics. We retain its complete released CLI harness and replace only the MCP task tools with the benchmark-native equivalents. These baselines cover deletion, masking, summarization, self-compression, and structured compression. None combines agent-facing context-state metadata with exact evidence recovery. On AMA-Bench, the EMem-style and Mem0-style rows are local adapters implemented for this harness and should be read as engineering baselines, not official reproductions.

Baseline reproduction details. SLIM and Active Context Compression are faithful reproductions of training-free published methods, run with the procedure described by their authors and triggered at the same 128K budget used for every method. SLIM periodically summarizes older context once the budget is exceeded, and Active Context Compression runs the explore, write a knowledge block, then prune the raw history loop. The structured-compression baseline is inspired by context-as-a-tool compressors but is not a faithful reimplementation, since that method is learned and we run no trained policy. For harness-native baselines we vary only the context-management mechanism and hold the agent loop, tools, budget, backbone, and scoring fixed. For Claude Code, the released CLI harness is held intact apart from substituting the benchmark-native task tools for the MCP task tools; task instances and scoring remain unchanged.

Context-management protocol. The agent receives the following instruction block appended to the task prompt, together with a budget notice. It is identical across backbones.

Context-management protocol (verbatim)

```
CONTEXT MANAGEMENT PROTOCOL:
A <context_workspace_status> dashboard is shown every turn as a compact map of
context blocks. Use context tools only when clearly needed. Do not archive,
delete, or offload content solely because it is old, large, or listed in context
metadata; leave content visible when the context budget is sufficient. Large
payloads may be represented by placeholders; inspect originals only when needed.
Use ordinary file/terminal/python tools, source metadata, and any in-context
payload placeholders to inspect external evidence when details are needed. For
structured data or calculations, use the source file, source tool, or query
directly. Do not copy table, CSV, or JSON rows from the conversation into code.

-------------------------------------------------------------------------------

You need to complete the task within the following context window size:
<budget:token_budget>128000</budget:token_budget>
```

Dashboard format. Each turn the harness injects a `<context_workspace_status>` block. It is a budget bar followed by one ledger row per block, with columns ID, approximate tokens, age (root-turn distance, where `0r` is newest), type, compression level, parent, and status (`visible`, `pinned`, `archived`, or `offloaded_placeholder`). The instance below shows the same compact column subset used in Figures 1 and 3; the full renderer additionally prints the compression Level and Parent columns described above. Age measures creation recency. The displayed schema includes no read-frequency or last-access field; additional renderer fields require saved prompts and update rules for verification.

Dashboard instance shown to the agent

```
## Context Budget
[#############--------] 62%  (~79,400 / 128,000 tokens)
  overhead ~6,200 | conversation ~71,900 | dashboard ~1,300

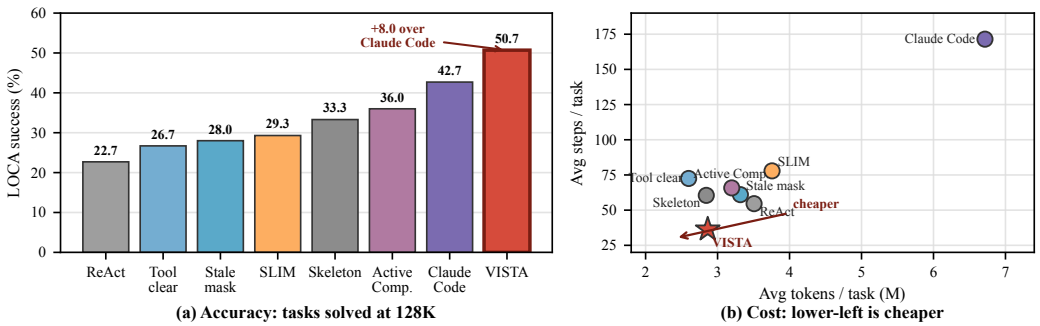
## Context Blocks  (Age = root-turn distance; 0r newest)
ID     ~Tok   Age  Type              Status
-------------------------------------------------
B1        120   8r  user_message      pinned
B2     18,400   7r  tool_call         archived
B3      2,150   6r  assistant_message visible
B5     14,800   3r  tool_call         visible
B6      9,300   2r  tool_call         visible
B9      1,070   0r  assistant_message visible
```

Context tool definitions. The agent acts on the workspace with two tools. Archiving replaces a block with a compact handle and returns the payload file path; the agent recovers byte-exact content by reading that path with ordinary file or terminal tools, so recovery is a normal read, with no dedicated decompressor.

Context tool definitions (docstrings)

```
context_workspace_archive(block_id: str, replacement: str = "") -> str
  Replace one or more blocks with compact indexes. The original content is
  stored externally as a payload file. Operations are block-level: only listed
  block IDs are archived. Returns the payload file path for later recovery.
    block_id:    Block IDs, ranges, or group IDs, e.g. "B3", "B3,B4",
                 "B10-B20", or "G2".
    replacement: Short index text for the archived block(s).

context_workspace_delete(block_id: str, reason: str) -> str
  Permanently remove one or more blocks. Deleted content cannot be recovered.
    block_id: Block IDs, ranges, or group IDs.
    reason:   Short reason why the content has no future task value.
```



(a) Accuracy: tasks solved at 128K (b) Cost: lower-left is cheaper

Figure 11: Expanded execution results. At the 128K LOCA setting, the panels compare task success, average token use, and average steps. VISTA has the highest observed accuracy and completes trajectories in fewer steps than the plotted baselines; Table 5 preserves the numerical results.

Table 5: Accuracy and execution outcomes at 128K. Each method runs all 75 LOCA tasks; errors and timeouts count as incorrect. VISTA has the highest observed accuracy and the fewest average steps. Dashes indicate unreported breakdowns; token and step means cover the available task rows.

Method	Correct	Acc.	Timeout	Error	Steps	Tokens
ReAct	17	22.7	0	0	54.6	3.51M
Tool-result clearing	20	26.7	0	0	72.9	2.60M
Stale masking	21	28.0	–	–	61.0	3.32M
SLIM	22	29.3	–	–	77.9	3.76M
Active compression	27	36.0	–	–	65.8	3.20M
Skeleton compression	25	33.3	–	–	60.5	2.84M
Claude Code	32	42.7	0	22	171.5	6.72M
VISTA	38	50.7	20	0	36.4	2.86M

Table 6: Reported management-event totals. Counts aggregate over the 75-task 128K LOCA runs. ReAct trims and clearing events alter visible history; VISTA records 69 archive calls and 105 successful payload recoveries. The operations differ in payload scope and can affect multiple blocks.

Method	Recorded operation	Events
ReAct	Trims	636
Tool-result clearing	Clears	1968
VISTA	Archive calls	69
VISTA	Successful payload recoveries	105

Large tool results are stored as external transcript payloads with compact placeholders. These payloads record what a tool returned to the model, not a complete source database; if a transcript is truncated or paginated, the agent must query the original source tool for complete data.

I Evaluation Details

For LOCA-Bench, we use the independently released public task suite and evaluation protocol without modification. It contains 75 online tool-task configurations with controllable context growth, where prior reasoning, tool calls, and observations accumulate until context management becomes central. We report task success, count errors and timeouts as incorrect, and compute average steps and tokens over task rows present in each run log. These cost values therefore describe observed execution cost, not cost conditioned on success. Archive/recovery figures count events aggregated over the 75 tasks, not per-task rates. An archive event is a workspace archive call, which can cover multiple blocks; a successful recovery event is a read that returns archived payload content. Ordinary environment retrieval is not an archive recovery, and failed attempts are separate outcomes. "Read" denotes the file/terminal operation; "recovery" denotes its successful archive-payload effect. The no-recovery condition has 57 failed attempts and no successful recoveries.

Figure 11 and Table 5 expand the 128K LOCA-Bench comparison, Table 7 gives the pairwise rescued-task split, and Table 8 gives exact counts for the context-growth sweep in Figure 4.

Table 7: Which tasks change outcome. Each row partitions the same 75 LOCA tasks at 128K into both correct, baseline only, VISTA only, and neither. VISTA-only successes exceed baseline-only successes in all four comparisons; these joint counts support Table 10.

Comparison	Both	Base.	VISTA	Neither
Fixed stale masking	18	3	20	34
SLIM	17	5	21	32
Active Context Compression	20	7	18	30
Claude Code	25	7	13	30

Table 8: Context-growth sweep in numbers. Both methods use the same 75 LOCA tasks at each setting. VISTA has higher success and lower average steps and tokens at every reported growth level; the accuracy margin is largest at 128K. Success is a percentage.

Method	Setting	Success	Avg. steps	Avg. tok.	Notes
VISTA	8K	86.7	16.7	0.44M	complete
VISTA	16K	84.0	16.2	0.52M	complete
VISTA	32K	70.7	17.4	0.87M	complete
VISTA	64K	61.3	21.2	1.38M	complete
VISTA	96K	57.3	29.7	2.32M	complete
VISTA	128K	50.7	36.4	2.86M	complete
VISTA	256K	32.0	43.1	3.51M	complete
ReAct	8K	84.0	27.9	0.68M	complete
ReAct	16K	74.7	24.9	0.80M	complete
ReAct	32K	65.3	25.3	1.02M	complete
ReAct	64K	52.0	29.6	1.70M	complete
ReAct	96K	36.0	39.5	2.79M	complete
ReAct	128K	22.7	54.6	3.51M	complete
ReAct	256K	12.0	82.6	5.80M	complete

Table 9: Transfer across six domains. In the same AMA replay evaluation, VISTA has higher F1 in five domains and higher judge accuracy in three. Bold marks the higher value within each pair.

	Accuracy		Token F1	
Domain	AMA	VISTA	AMA	VISTA
Embodied	0.678	**0.683**	0.489	**0.525**
Game	**0.814**	0.747	0.407	**0.443**
OpenQA	**0.817**	0.731	0.249	**0.386**
Software	0.565	**0.567**	0.150	**0.238**
Text2SQL	0.838	**0.846**	**0.395**	0.384
Web	**0.782**	0.763	0.272	**0.341**

For BrowseComp-Plus, we evaluate deep-research retrieval with DeepSeek-V4-Pro on an N=150 subset and report judged Pass@1 with one sampled answer. The agent searches the benchmark's fixed corpus (Chen et al., 2025), so evidence is scattered across retrieved passages and the transcript grows through repeated retrieval. To expose context management, we use a deliberately tight active window (W=12K tokens per call) and total budget (B=160K), chosen so early evidence can be evicted before synthesis.

For GAIA, we use a fixed random 165-question subset of the public validation split. We preserve the original question text and attached files, require the official `FINAL ANSWER:` format, and score with quasi-exact match. All methods use DeepSeek-V4-Pro, real web/search/file tools, W=12K, and B=80K.

For AMA-Bench, each of the 208 episodes contains 12 open-ended questions, giving 2496 judged QA pairs. We report judge accuracy and token-level F1 following the benchmark harness. Runtime per episode is measured for generation; judge time is reported separately in the analysis files.

I.1 Uncertainty from reported paired outcomes

This post-hoc analysis uses the joint outcome counts in Table 7. Let b be baseline-only successes and c VISTA-only successes among $n = 75$ task pairs. The observed accuracy difference is $(c - b)/n$.

Table 10: Paired task outcomes and uncertainty. On the 75 LOCA tasks at 128K, b/c counts baseline-only/VISTA-only successes, and Δ is VISTA's accuracy gain. Intervals use a task-pair percentile bootstrap; exact p values use McNemar's test. Holm adjusts these four comparisons. Stale masking and SLIM remain below $p = 0.05$ after adjustment.

Baseline	b/c	Δ (pp)	95% interval (pp)	Exact p	Holm p
Stale masking	3/20	22.7	$[12.0, 34.7]$	0.00049	0.00195
SLIM	5/21	21.3	$[9.3, 33.3]$	0.00249	0.00748
Active compression	7/18	14.7	$[2.7, 26.7]$	0.04329	0.08657
Claude Code	7/13	8.0	$[-4.0, 20.0]$	0.26318	0.26318

Table 11: Exact-evidence use in rescued tasks. Four 128K LOCA examples succeed with VISTA after the listed baselines fail. Archive calls, subsequent payload uses, and benchmark acceptance signals connect working-set edits to completed task artifacts; several payload reads can follow one archive.

Case	Baselines failed	VISTA steps	Archive/read evidence	Evaluation signal
NHL B2B schedule analysis	ReAct, Tool-clear, SLIM	64	5 archive calls; 17 payload reads/uses	CSV and Google Sheet correct; HA/AH/HH/AA counts verified
WooCommerce low-selling products	ReAct, Claude-style, Tool-clear, SLIM	38	4 archive calls; 3 payload reads/uses	Correct products moved; subscriber emails sent
Canvas final-exam schedule	ReAct, Claude-style, Tool-clear, SLIM	17	1 archive call for course announcements	Final Excel schedule accepted
NLP course reminders	ReAct, Claude-style, Tool-clear, SLIM	31	1 archive call for large roster table	Correct students emailed; dropped/submitted students excluded

Under the conditional null that either method is equally likely to win each discordant pair, the two-sided exact McNemar p value is

$$p = \min\left\{1,\ 2\sum_{j=0}^{\min(b,c)}\binom{b+c}{j}2^{-(b+c)}\right\}.$$

For the paired percentile bootstrap, sample n pairs with replacement from the empirical difference distribution: -1 with probability b/n, $+1$ with probability c/n, and 0 otherwise. We obtain the 2.5% and 97.5% quantiles of the mean by discrete convolution. The joint counts suffice for this binary-outcome statistic. `scripts/paired_uncertainty.py` reproduces the calculation.

Both procedures treat task pairs as independent sampling units. Task families and decoding-seed variation require clustering and repeated runs. The Claude Code interval spans zero. Under the four-comparison Holm correction, stale masking and SLIM have $p < 0.05$; Active Context Compression and Claude Code have $p > 0.05$. Bootstrap intervals are unadjusted and pointwise.

What marginal counts alone imply. For a baseline solving s tasks, let a be the unknown number both solve. Feasible pairings satisfy $\max(0, 38+s-75) \le a \le \min(38, s)$, with $b = s-a$ and $c = 38-a$. Enumerating these integer possibilities yields ranges of the exact p value; the range reflects the unknown joint pairing. For Auto-Archive + Recover ($s = 33$), the range is $[0.0625, 0.6353]$. Even five VISTA-only wins with no losses give $2/2^5 = 0.0625$. For the no-dashboard ablation ($s = 28$), the range is $[0.0020, 0.2678]$; for no recovery ($s = 34$), it is $[0.1250, 0.7239]$. The no-dashboard range straddles the 5% threshold; the no-recovery range lies above it. For ReAct ($s = 17$), the full range lies below 5%, from 9.54×10^{-7} to 0.00646, under the same nominal assumptions. Actual paired logs determine the realized p values within these ranges; task-family identifiers and repeated runs support cluster-aware and decoding-variance analyses.

J LIMITATIONS

Most benchmark cells use one sampled trajectory per task. The task-pair analysis in Appendix I.1 conditions on those observed outcomes; decoding-seed variation and correlations among task fam-

Table 12: Reading explicit context sizes. Four backbones estimate sizes on archive-triggered LOCA snapshots, without (−) or with (+) the dashboard. Size columns give median relative error; pairwise accuracy uses pairs within 2× in true size. The ledger eliminates median total-size error.

	Total size		Block size		Pairwise	
Backbone	−dash	+dash	−dash	+dash	−dash	+dash
Claude-Sonnet-4.5	0.84	**0.00**	0.37	**0.02**	0.67	0.83
DeepSeek-V4-Pro	0.44	**0.00**	0.28	**0.00**	0.75	1.00
GLM-5	0.48	**0.00**	0.35	**0.00**	0.73	0.88
Gemini-3-Flash	0.43	**0.00**	0.24	**0.00**	0.68	1.00

ilies remain unmeasured. The Claude Code interval spans zero, and the joint outcomes for Auto-Archive and the dashboard ablation are unavailable. Repeated matched runs and task-family identifiers would sharpen these comparisons.

The experiments evaluate the combined agent and runtime guard. Guard intervention counts and a ledger-aware fixed controller remain missing controls. The dashboard ablation removes several fields together; the direct probes measure token size. Field-wise tests, tokenizer calibration, summary quality, and ledger size are open dimensions of sensitivity. The displayed age field records creation recency. Failure categories have yet to be assigned to the 37 unsuccessful LOCA-Bench runs and 63 unsuccessful BrowseComp-Plus questions.

Cost estimates cover model API usage for four methods. Full-system latency, storage and I/O costs, and several token-accounting scopes remain unresolved (Appendix M). The EMem-style and Mem0-style AMA rows use local adapters. Matched evaluations against PACE, HiAgent, and learned context managers would extend the comparison set (Wei et al., 2026; Hu et al., 2025; Lu et al., 2026a; Yi et al., 2026; Yu et al., 2026; Zhang et al., 2026c).

The theory uses explicit information and paging assumptions. The RL study uses the original surrogate, with the boundary behavior derived in Section L.1; reproducing it requires the stabilization and split records listed in Appendix M. Its training curves track co-occurring changes, while a causal analysis would require matched interventions and length-normalized management rates. The bounded alternative in Appendix L.3 remains unevaluated.

K Context-Size Estimation Diagnostic

This appendix documents the diagnostic behind Table 12. The diagnostic measures token-size estimation and reading of supplied counts under the tested prompts.

Data. We anchor on the first archive event of each real LOCA-Bench run, the moment the agent itself decided to externalize content. We take the accumulated transcript just before that call as the snapshot, treating each message as one block. The runtime dashboard is not persisted in the transcript, so the stored messages are already free of the live ledger; we additionally strip the three injected artifacts that would leak state, namely the context-management protocol header, the hard-limit rejection notices that print token counts, and archived-block placeholders. A scan over all anchored snapshots confirms no residual token, budget, or usage strings remain, and there is no per-block usage annotation. Twenty-nine runs contain an archive; we cap each snapshot at 100K tokens by dropping trailing blocks so it fits every backbone window, and compute ground truth with the same tokenizer used by the harness.

Questions and conditions. We ask three quantities, each in its own request so the measurements stay independent. *Total size*: estimate the token count of the whole transcript. *Block size*: estimate the token count of four sampled blocks. *Pairwise*: for sampled block pairs, say which is larger, reported on the hard subset within 2× in true size. The *−dash* condition shows the cleaned transcript only; the *+dash* condition prepends the factual ledger (the columns of the implementation-details appendix). Size answers are scored as median relative error and pairwise as accuracy against the larger block. We run Gemini-3-Flash, Claude-Sonnet-4.5, DeepSeek-V4-Pro, and GLM-5 with greedy decoding. Claude-Sonnet-4.5 returns valid structured output slightly less often than the other three, but the qualitative gap and its closure with the dashboard hold for every backbone. These probes

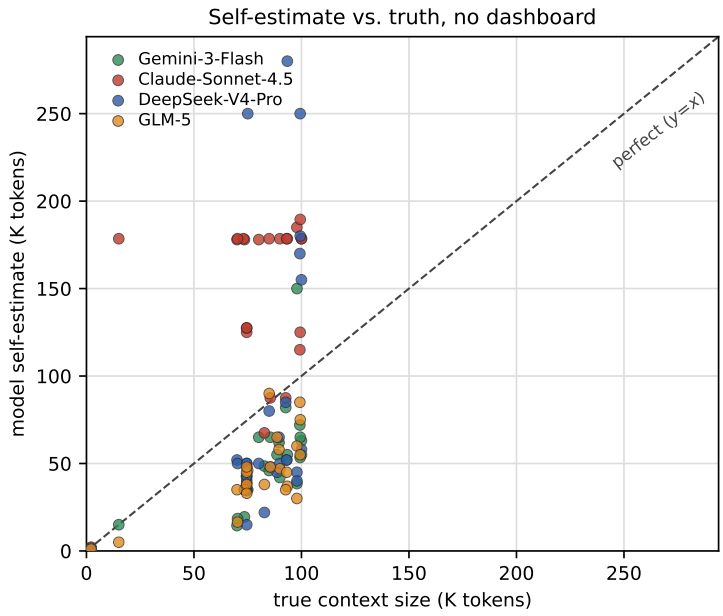


Figure 12: Dashboard-free size estimates. At archive-triggered LOCA snapshots, four backbones estimate total transcript size without the ledger. The diagonal marks exact agreement with harness token counts; the scatter shows substantial underestimation on many snapshots. Table 12 reports the dashboard-assisted counterpart and the block-level probes.

measure size-estimation errors under the tested conditions. The interface also exposes creation age and archive status. The information quantity in Theorem 1 follows its explicit observation model.

L RL Adaptation of the VISTA Interface

L.1 Trajectory Optimization

The interface above is training-free: an instruction-following model can use the dashboard and context tools without changing its parameters. We additionally consider a post-training branch that refines *when* and *how* the same policy invokes archive and recovery. This branch leaves the workspace, dashboard, and lossless tool contract unchanged. Following the trajectory-level optimization pattern of Context-Folding (Sun et al., 2025), we optimize complete interaction trajectories while conditioning every model token on the workspace that was actually visible when that token was generated.

Workspace-consistent rollouts. For a training task g, the old policy samples a group of G trajectories $\{\tau_i\}_{i=1}^{G}$. Let $W_{i,t}$ be the workspace immediately before token $x_{i,t}$ and let $F(W_{i,t})$ denote the actual post-preflight input $C_{i,t}$ in Eq. (5), including dashboard, notices, message roles, system instructions, and tool schemas. The notation suppresses these additional runtime inputs; the action mask must also be identical during sampling and likelihood recomputation. Archive actions remove their selected payloads from subsequent renderings, whereas recovery actions restore only the content actually read. Let $H_{i,t} := F(W_{i,t}) \oplus x_{i,<t}^{\text{turn}}$ include the autoregressive prefix already generated in the current LLM turn. The importance ratio is therefore

$$r_{i,t}(\theta) = \frac{\pi_\theta(x_{i,t} \mid H_{i,t})}{\pi_{\text{old}}(x_{i,t} \mid H_{i,t})}, \tag{9}$$

and $m_{i,t} = \mathbf{1}_{\text{LLM}}(x_{i,t})$ masks tool observations. Thus rollout and policy optimization use the same managed context; reconstructing the raw, append-only transcript for the actor update would train a different policy.

Outcome reward and process penalties. Each trajectory receives a task reward $R_i \in [0, 1]$ from the benchmark scorer. Sparse task reward determines whether management preserved useful evidence. We use heuristic penalties for high occupancy and executor-detected management events. Let $p_{i,t} =$

$|F(W_{i,t})|/B$ be active-context pressure and ρ a soft pressure threshold, with $0 \le \rho < 1$. Pressure here refers to the admitted management input, not the possibly over-budget candidate $\widetilde{C}_{i,t}$; these two quantities must not be interchanged in an implementation. The token-level process signal is

$$\begin{aligned} Q_{i,t} &= Q_{i,t}^{\text{pressure}} + Q_{i,t}^{\text{invalid}} + Q_{i,t}^{\text{undo}}, \\ Q_{i,t}^{\text{pressure}} &= -\lambda_p \max\left(0, \frac{p_{i,t} - \rho}{1 - \rho}\right). \end{aligned} \tag{10}$$

Pressure is computed once from the pre-action rendering—including visible blocks, handles, notices, and dashboard—and shared by all LLM tokens in that turn; a workspace transition affects pressure from the next turn onward. $Q_{i,t}^{\text{invalid}} = -\lambda_i$ iff the context-tool executor returns an error or the canonical workspace hash is unchanged, covering missing handles, malformed calls, and exact repetitions. $Q_{i,t}^{\text{undo}} = -\lambda_u$ marks an otherwise valid archive–recovery pair intended to restore the same logical working set before any intervening new task evidence or context growth; it is assigned post hoc to the LLM tokens of both action turns. An exact hash of the full append-only execution state would generally not repeat after a read. Reproducing this detector therefore requires the canonicalization rules for payload identities, summaries, ordering, archive metadata, new block IDs, and tool-call logs, as well as the treatment of the recovery read itself. These implementation rules remain to be verified (Appendix M). The three terms add when they co-occur and are clipped only in Eq. (11). We use $\rho = 0.8$, $\lambda_p = \lambda_i = 1$, and $\lambda_u = 0.2$. Archiving below a fixed occupancy threshold is not penalized: a small prompt may still contain low-density content worth externalizing proactively.

The reported formulation uses the following outcome-normalized, process-shaped token advantage, with $\epsilon > 0$:

$$\widehat{A}_{i,t} = \frac{\text{clip}(R_i + Q_{i,t}, 0, 1) - \mu_R}{\sigma_R + \epsilon}, \quad \mu_R = \frac{1}{G}\sum_{j=1}^{G} R_j, \tag{11}$$

where $\sigma_R := \sqrt{G^{-1}\sum_{j=1}^{G}(R_j - \mu_R)^2}$ is the within-task outcome standard deviation. The group baseline and scale come from outcome reward, while $Q_{i,t}$ supplies local process shaping. This is a specified surrogate, not an unbiased policy-gradient derivation for cumulative management cost. In particular, the pre-action pressure is shared by every token in a turn; future pressure is not propagated to earlier actions through a cost-to-go. The policy maximizes the clipped objective

$$J_{\text{RL}} = \mathbb{E}\left[\frac{1}{\sum_i |\tau_i|}\sum_{i,t} m_{i,t} \min\Big(r_{i,t}\widehat{A}_{i,t}, \text{clip}(r_{i,t}, 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})\widehat{A}_{i,t}\Big)\right]. \tag{12}$$

Here $|\tau_i| := \sum_t m_{i,t}$ is the number of model-generated tokens in trajectory i. The expectation is over training tasks and groups sampled from π_{old}, with advantages held fixed during the update; groups must have $\sum_i |\tau_i| > 0$. This normalization weights model tokens equally within each group, so trajectories and turns with more tokens receive greater total weight. Only model-generated tokens contribute to Eq. (12); ordinary reasoning, context actions, recovery decisions, and final synthesis are optimized jointly. Training prompts are screened using training-split pilot rollouts to select prompts that produced both successful and unsuccessful samples. This can yield more informative outcome comparisons, but does not guarantee positive outcome variance in later online groups.

Boundary behavior and reproducibility. Equation (11) has two consequential boundary cases. If every $R_i = 1$, then $\mu_R = 1$, $\sigma_R = 0$, and a token with $Q_{i,t} = -0.2$ has $\widehat{A}_{i,t} = -0.2/\epsilon$. Thus all-equal outcomes do not imply zero shaped advantages, and a small numerical ϵ can amplify the process term severely. PPO ratio clipping alone does not bound the advantage magnitude. Conversely, because $Q_{i,t} \le 0$, every failed trajectory with $R_i = 0$ has $\text{clip}(R_i + Q_{i,t}, 0, 1) = 0$: process differences are erased, and all-failure groups have zero advantage. With $\lambda_i = 1$, an invalid event also saturates the shaped reward at zero for any $R_i \in [0, 1]$, suppressing incremental pressure or undo penalties. Consequently, the three penalties are additive only before clipping.

Reproduction requires the actual low-variance handling, advantage cap, gradient-norm rule, and values of G, ϵ, and the PPO clip parameters. These settings remain pending implementation checks. Appendix L.3 gives an unevaluated bounded alternative.

This reward deliberately excludes the number of context-tool calls, active-context reduction, and self-perception probe accuracy. These remain held-out mechanism measurements: directly optimizing them would permit trivial over-archiving and make post-training perception gains circular.

L.2 Adaptation Results

Table 2 reports a Qwen3-8B adaptation study. In-Distribution (ID) uses BrowseComp-Plus for training and evaluation; the reported Out-Of-Distribution (OOD) setting trains on GAIA and evaluates on BrowseComp-Plus. The training-free interface scores 20.0%, compared with 13.3% for Base. The reported RL variant scores 31.3% ID and 21.3% OOD, compared with 27.3% and 17.3% for context-tool GRPO.

Training prompts were screened with training-split pilot rollouts; validation and test examples were excluded from screening. These runs use the original surrogate in Section L.1. The bounded alternative in Appendix L.3 is unevaluated. Split manifests, training budgets, and stabilization settings are recorded as pending implementation checks in Appendix M.

Figure 7 tracks validation reward, a self-perception probe, management calls per trajectory, and context reduction across checkpoints. Hard groups have more calls per trajectory than easy groups. Comparing management rates requires episode-length and occupancy controls.

L.3 Proposed Bounded Alternative (Not Evaluated)

The following proposal addresses the numerical coupling and clipping behavior of Eq. (11). It is not a claim about the algorithm used to produce the reported results. Choose an outcome-scale floor $s_0 > 0$, a process-cost cap $c_{\max} > 0$, a process weight $\beta \geq 0$, and an advantage cap $A_{\max} > 0$. Define

$$c_{i,t} = \mathrm{clip}(-Q_{i,t}, 0, c_{\max}), \tag{13}$$

$$\widetilde{A}_{i,t} = \mathrm{clip}\left(\frac{R_i - \mu_R}{\max(\sigma_R, s_0)} - \beta c_{i,t}, -A_{\max}, A_{\max}\right). \tag{14}$$

The outcome and process scales are now separate. If all outcomes are equal, the outcome component is zero and the process component stays bounded; in particular, it is not divided by ϵ. When $R_i = 0$, differences in $c_{i,t}$ remain visible unless the explicit cost or advantage cap saturates. The pre-cap magnitude is at most $1/s_0 + \beta c_{\max}$ and the final magnitude is at most $A_{\max}$. Identical outcomes and identical costs still supply no within-group preference; this proposal does not manufacture missing task signal.

This is still a local, token-weighted shaping surrogate. Keeping pre-action pressure retains the credit-assignment limitation discussed in Section L.1. If the intended objective is cumulative management cost, it should instead be specified at the action/turn level, for example

$$J_{\mathrm{task-cost}}(\theta) = \mathbb{E}_{g,\tau\sim\pi_\theta}\left[R(\tau) - \beta \sum_{u=1}^{T} \gamma^{u-1} c_u\right], \qquad 0 < \gamma \leq 1.$$

Here c_u must have a declared timing convention and may depend on the state transition induced by action u. An estimator for this objective must account for downstream costs, using an appropriate return-to-go and baseline, and must declare the action-to-token weighting. It is not obtained merely by renaming the original pre-action penalty a cumulative return.

Validation requires reporting $G, s_0, c_{\max}, \beta, A_{\max}$, PPO clip values, any gradient-norm limit, and the sampling/update protocol. Diagnostics should include rates of all-success/all-failure and low-variance groups, advantage quantiles and maxima, cap activation rates, and invalid/undo events stratified by task outcome. Comparisons should include outcome-only optimization, the original surrogate with its actual stabilization, and the proposed variant, using matched training budgets and held-out evaluation. No hyperparameter value or experimental improvement for this proposal is inferred from the supplied source.

M Implementation and Measurement Record

The following implementation and measurement records remain to be verified against the evaluated code and run logs.

1. **RL configuration.** Record group size, numerical epsilon, PPO clip parameters, online low-variance handling (including skip/resample rules), advantage clipping, gradient clipping, token masks, and empty-group handling. Clarify which safeguards, if any, were used in the reported runs. Supply the actual train/evaluation split manifests, training-token budgets, update counts, and whether rewards and rubrics were matched across the RL comparison. Define the training-curve perception task, sample count, fixed evaluation set, difficulty strata, and context-reduction denominator.
2. **Training state and events.** Verify that stored policy inputs are post-preflight inputs with identical schemas and action masks at update time. Check whether logged pressure used the candidate or admitted input. Publish the invalid/undo detector, canonicalization fields, and representative event traces, including recovery reads that add tool messages and new IDs.
3. **Budget contract.** Specify the token estimator and relation to the provider tokenizer; show the minimum protocol/management input bound, treatment of oversized pinned tasks, ledger paging or hierarchical representation, final request checks, infeasibility returns, and iteration/time limits. Distinguish workspace cap, provider hard limit, and reserved output. Log each guard trigger, its selection rule, candidate/admitted lengths, moved blocks/bytes, and whether it affected task evidence. Compare guard-on and guard-off at the same safely reduced workspace cap, retaining provider request validation in both arms. The normative pseudocode is not a substitute for verifying harness behavior.
4. **Theory-aligned measurements.** Log distinct future-relevant blocks archived, distinct archived blocks subsequently accessed, request outcomes for unaccessed blocks, batch sizes, repeated reads, and early termination. Event counts alone cannot evaluate the theorem's block-level quantity. Report that the no-recovery condition's 57 attempts failed and that all other plotted recovery counts are successful events; preserve both outcome fields in logs. Reconcile whether the Auto-archive row retains the dashboard: the single-flag description and State column must refer to the same configuration.
5. **Statistical and accounting scope.** Report paired outcomes and repeated-run uncertainty. Reconcile BrowseComp-Plus's 160K stopping budget versus 247K/423K reported trajectory tokens: specify run sets, tokenizer, thinking tokens, management calls, repeated inputs, and aggregation scope. Identify whether window selection used development or test gold-document statistics; align AMA construction, retrieval, and QA timing. For dashboard traces, export same-turn text, displayed rows/groups, block counts, and overhead subtraction. Report per-baseline input/output tokens, all management calls, bytes written/read, peak archive size, dashboard construction time, tool I/O time, and end-to-end latency. Preserve reported totals pending verification.
6. **Mechanism controls.** Use matched task/seed pairs, the same lossless store, and an identical guard to compare no ledger, total-budget-only, full factual ledger, and equal-format perturbed-size ledgers. Add a fixed size-aware controller with declared pinning and tie-breaking rules. Test summary quality, row count, tokenizer error, and stale metadata separately; creation age must not be labeled access history. Predeclare failure categories and annotate all unsuccessful runs, permitting unknown or multiple causes.