# BUDGETWEAVER: LEARNING THE DOWNSTREAM VALUE OF AGENT READS AND MEMORY MUTATIONS UNDER SHARED BUDGETS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Every optional-memory action changes two things at once: the current context and the evidence–budget state inherited by later decisions. A write can preserve verified progress, an eviction can remove a stale hypothesis, and a summary can make later evidence affordable. Controllers that score each operation only for its immediate use miss this intertemporal choice when every read and mutation spends the same token–latency budget. BudgetWeaver learns the downstream value of a seven-operation sequence over `NOOP`, retrieval, reranking, graph traversal, summarization, writing, and eviction. Its policy combines remaining budget and memory state with paired read-versus-`NOOP` endpoint outcomes, support-aware offline value refinement, predicted-cost admission, and realized-cost accounting. We isolate the controller by fixing prompts, candidates, operators, retrievers, stores, schemas, and guards across all matched policies. At the predeclared medium operating point, BudgetWeaver improves native success over tuned full-interface non-RL control by 1.8–2.6 points on LOCOMO v2, SWE-bench Lite v1.1, and OdysseyBench v1.3; every paired interval excludes zero and all three tests remain significant after Holm correction. Relative to always-retrieve, success rises by 3.7–4.9 points while paired full-run token use falls by 4.1k–5.4k and end-to-end p95 latency falls by 1.6–2.7 seconds. Removing persistent mutations costs 1.3–2.8 points, with the largest descriptive effects in contradiction-heavy and longest-horizon episodes, and separately fitted Llama-3.1-8B-Instruct controllers retain a 1.7–2.6-point lead over tuned non-RL control. These results make downstream memory-state value an actionable control signal: coordinating reads and mutations improves success while reducing complete-run resource use.

## 1 INTRODUCTION

Long-horizon agents do not merely retrieve from memory; they repeatedly decide what to read, compress, preserve, or discard. These decisions compete for finite tokens and latency, yet their benefits arrive on different timescales. A retrieval can expose evidence immediately. A write or eviction may produce no immediate answer while changing every later decision that inherits the modified memory. Independent operation triggers therefore hide the opportunity cost that matters: spending resources now changes both future evidence and the set of actions that remains affordable.

Software repair makes the failure concrete. After a patch is rejected, another repository search may reveal the defect, but recording the verified failure and removing a contradicted hypothesis can improve all subsequent searches and edits. A controller that asks only whether the next retrieval is useful cannot compare those futures. The central problem is to value the memory state produced by a sequence of reads and mutations while a shared resource envelope is being depleted.

BudgetWeaver solves this problem as endpoint-valued sequence control. At each optional-memory epoch, it observes uncertainty, verifier and tool outcomes, managed memory state, cumulative write volume, and remaining token–latency budget, then chooses among `NOOP`, `RETRIEVE`, `RERANK`, `GRAPH_HOP`, `SUMMARIZE`, `WRITE`, and `EVICT`. Paired read-versus-`NOOP` continuations connect local read decisions to native task outcomes. Behavior cloning anchors the policy to logged actions;



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.

support-aware offline refinement propagates value through later memory states; predicted-cost admission and realized accounting expose the same remaining budget to every operator.

The evaluation isolates this controller from memory-system quality. Prompts, tools, candidate sets, retrievers, rerankers, stores, schemas, guards, and operator implementations are identical across matched policies. With a 16k optional-memory token threshold and benchmark-specific medium latency thresholds, BudgetWeaver improves success over tuned full-interface non-RL control by 1.8 / 1.9 / 2.6 points on LOCOMO, SWE-bench Lite, and OdysseyBench. Every paired confidence interval excludes zero and the three comparisons survive Holm correction. Against always-retrieve, success improves by 3.7 / 3.7 / 4.9 points while paired full-run tokens fall by 4.1k–5.4k and p95 latency falls by 1.6–2.7 seconds. Read-only ablations lose 1.3–2.8 points, and a full refit with Llama-3.1-8B-Instruct preserves the three-benchmark ordering.

Learned memory managers provide add, update, deletion, consolidation, and verifier-guided operations, while adaptive retrieval and BudgetMem learn whether or how much evidence to acquire (Yan et al., 2026; Huo et al., 2026; Yu et al., 2026; Sun et al., 2026; Jiang et al., 2026; Zhang et al., 2026). BudgetWeaver introduces a different control object: read and persistent-mutation choices share one observable remaining-budget state, and their value is learned from the native endpoint while the underlying memory machinery is held fixed. This formulation turns downstream memory quality into a measurable allocation target rather than a side effect of independently designed triggers.

Our contributions are:

- **Endpoint-valued memory control.** We formulate reads and persistent mutations as one constrained sequence with explicit memory transitions, paired read counterfactuals, delayed-value refinement, action-conditional support, predicted-cost admission, and realized-cost accounting.

- **Controller-isolating evidence.** Across conversational QA, software repair, and office workflows, a matched-stack design shows that BudgetWeaver raises native success over full-interface hand-designed coordination and aggressive retrieval while reducing complete-run tokens and tail latency.

- **Downstream-state mechanism.** Read-only and operator ablations, predefined density/contradiction/horizon slices, logged-support strata, and an independently fitted 8B replication connect the aggregate gain to persistent state value and supported sequential refinement.

Figure 1 locates the learned decision boundary between observed signals and fixed memory machinery.

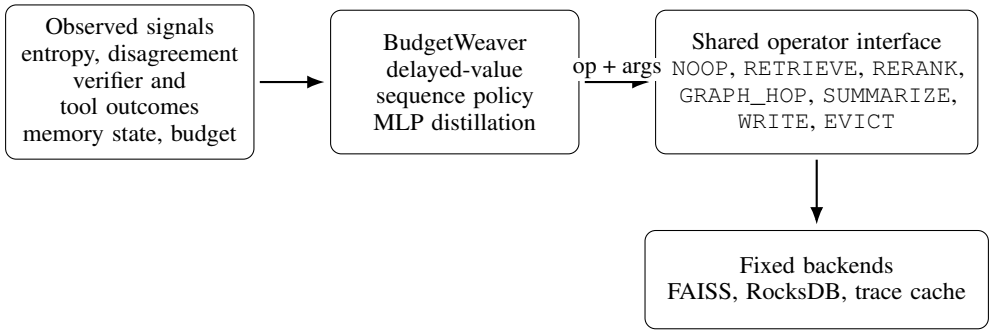


Figure 1: Endpoint-valued memory-control interface. BudgetWeaver selects one admitted read or mutation from current signals, remaining budget, and predicted downstream state value. Retrieval, reranking, graph, write, and guard components are shared by every matched policy.

# 2 INTERTEMPORAL MEMORY CONTROL

## 2.1 SHARED-CAP DECISION PROCESS

A decision epoch is one invocation of the optional-memory controller before the base agent continues. At most one memory operation executes per epoch; `NOOP` returns control immediately. Observation $o_t = (u_t, e_t, m_t, b_t, w_t)$ contains uncertainty $u_t$, environment feedback $e_t$, managed memory state $m_t$, remaining token–latency budget $b_t$, and cumulative write volume $w_t$. The action $a_t = (\mathtt{op}_t, \theta_t)$ chooses

$$\mathtt{op}_t \in \{\mathtt{NOOP}, \mathtt{RETRIEVE}, \mathtt{RERANK}, \mathtt{GRAPH_HOP}, \mathtt{SUMMARIZE}, \mathtt{WRITE}, \mathtt{EVICT}\}$$

and parameters $\theta_t$, including retrieval depth, rerank depth, maximum chunks, TTL, and tags. Read operators expose evidence to working context; mutation operators transform managed state through $T_{\mathtt{op}_t}$. Candidate generation, ranking models, graph construction, summary prompts, write schemas, and eviction implementations are shared across policies.

Before execution, the controller estimates token and latency cost $\hat{c}_t(a)$ for admission. Execution then produces realized vector cost $c_t = (c_t^{\mathrm{tok}}, c_t^{\mathrm{lat}}, c_t^{\mathrm{write}})$ and the coupled transition

$$m_{t+1} = T_{\mathtt{op}_t}(m_t, \theta_t, e_t), \qquad b_{t+1} = b_t - \big(c_t^{\mathrm{tok}}, c_t^{\mathrm{lat}}\big), \qquad w_{t+1} = w_t + c_t^{\mathrm{write}}.$$

Given remaining caps $b_t$, the admission set is

$$\mathcal{F}(b_t) = \{a : \hat{c}_t^{\mathrm{tok:lat}}(a) \preceq b_t\} \cup \{\mathtt{NOOP}\}.$$

The mask is prospective, whereas realized accounting is authoritative: predicted consumption determines admission and measured consumption updates the remaining budget after execution. We therefore report realized overrun counts rather than interpret $\mathcal{F}$ as a certified worst-case bound; the held-out 400-episode stress test records zero realized overruns (Appendix C). Cost estimation and masking are shared by every comparator, isolating the policy that allocates the admitted operations. Because $T_{\mathtt{op}_t}$ changes both the evidence and admitted actions inherited by successors, the return prices immediate reads and downstream memory-state effects.

The signal state instantiates four families. Uncertainty uses mean token entropy, log-probability margin, and disagreement across $K{=}5$ self-consistency samples, following calibration and failure-detection practice (Lakshminarayanan et al., 2017; Guo et al., 2017; Hendrycks & Gimpel, 2017). Environment feedback includes verifier pass/fail, tool errors, and latency spikes. Memory state contains novelty, density, contradiction count, and freshness. Novelty is $1 - \max_{m \in \mathcal{M}} \cos(e_q, e_m)$; density is the mean cosine similarity to the $k{=}16$ nearest memory embeddings, so larger density consistently denotes greater local redundancy. Budget signals record remaining optional-memory tokens, remaining optional-memory latency, and cumulative write volume.

Token caps count optional-memory consumption by the six non-`NOOP` operators, separate from full-run agent tokens. Latency caps cover cumulative optional-memory wall time, separate from end-to-end runtime; write bytes remain an observed and reward-penalized cost. If no non-`NOOP` action is admitted, the mask forces `NOOP` while the base agent continues. We report *cap-pass* as the fraction of episodes that never require this cap-forced fallback; this quantity is distinct from the observed overrun count.

## 2.2 BUDGETWEAVER

BudgetWeaver estimates endpoint value over the admitted operator sequence. A four-layer, hidden-size-256 causal sequence teacher with four attention heads encodes the latest 16 observation, action, cost, and outcome steps. Separate categorical heads score the operator and low-cardinality knobs, including `top_k`$\in\{8,16,32,64\}$ and `rerank_k`$\in\{0,16,32\}$; masks remove incompatible operator–parameter pairs before normalization. Sequence context carries mutation effects into the scores of subsequent reads. Controllers are fitted separately for each benchmark, base-model scale, and seed, so neither policy parameters nor support neighborhoods are pooled across task families.

The predicted-cost mask and learning objective play different roles. Masking removes actions outside $\mathcal{F}(b_t)$; reward terms price all three realized cost dimensions before a cap binds:

$$R = \text{Success} - \lambda\,\text{Tokens} - \mu\,\text{Latency} - \nu\,\text{WriteBytes} - \xi\,\text{BadRetrieval}.$$

Units are 1k optional-memory tokens, optional-memory seconds, and MB. The medium operating point uses $\lambda$=0.020, $\mu$=0.40, $\nu$=0.006, and $\xi$=0.80; Pareto sweeps vary $\lambda \in [0, 0.04]$ and $\mu \in [0.20, 0.80]$.

`BadRetrieval` anchors local read decisions to endpoint utility. Let $Y(\tau)$ denote native task success for a completed rollout and define

$$\Delta_t^{\text{read}} = Y(\tau_t^{\text{read}}) - Y(\tau_t^{\text{NOOP}}).$$

The paired branches use common random numbers and start from identical serialized agent, tool, memory, prompt, seed, and budget state. After forcing the first action, both branches follow the same frozen logging-mixture continuation policy through the native endpoint; common random numbers cover model sampling and tool nondeterminism. For any read operator, $\texttt{BadRetrieval}_t = \mathbf{1}[\Delta_t^{\text{read}} \le 0]$, so nonpositive endpoint value penalizes the read in the refinement reward. This counterfactual term does not relabel the behavior-cloning target: BC-only clones logged mixture actions, while the refined policy additionally optimizes the endpoint-aware return. The estimand values the first read under a common continuation, and sequence refinement propagates return through mutation-induced state changes.

Behavior cloning first fits all operator and parameter heads to the logging mixture. For transition reward $r_t$, resource penalties accrue at each step and native success is added only at the terminal transition. IQL-style refinement then minimizes

$$\mathcal{L}_V = \mathbb{E}\big[L_2^\tau(\bar{Q}(o_t, a_t) - V(o_t))\big], \quad \mathcal{L}_Q = \mathbb{E}\big[(r_t + \gamma V(o_{t+1}) - Q(o_t, a_t))^2\big],$$

and updates the actor by advantage-weighted log likelihood,

$$\mathcal{L}_\pi = -\mathbb{E}\big[\exp\{\beta(\bar{Q}(o_t, a_t) - V(o_t))\} \log \pi(a_t \mid o_t)\big],$$

where $L_2^\tau(z) = |\tau - \mathbf{1}[z < 0]|z^2$ and $\bar{Q}$ is the EMA target. We use $\gamma$=0.99, $\tau$=0.7, $\beta$=3.0, actor/critic learning rate $3\times10^{-4}$, batch size 512, EMA rate 0.005, 220k updates, and a per-update KL cap of 0.02 (Kostrikov et al., 2022; Kumar et al., 2020).

Support is computed in the frozen 16-step sequence representation used by the value heads. An action is supported when a development trace contains the same action inside its action-specific radius; each radius is calibrated on development traces and fixed before test. Unsupported refined actions are removed before normalization, and the controller falls back to the masked BC distribution when no supported refined non-`NOOP` choice remains. For deployment, an MLP student consumes the same fixed feature window and mirrors the teacher's operator and parameter heads. Distillation matches teacher and student logits after compatibility and feasibility masks, using fitting traces only; development masked-logit error selects the student. All reported BudgetWeaver evaluations execute this distilled path, while the frozen sequence representation supplies the support test. A typed write schema and fixed redaction/filter gate are shared by every policy.

*Algorithm 1: cost-aware endpoint-valued inference.* Given agent state $s_t$, memory $\mathcal{M}_t$, remaining budget $b_t$, and write volume $w_t$: (1) construct the 16-step feature window; (2) enumerate compatible operator parameters and predicted costs $\hat{c}_t$; (3) remove actions outside $\mathcal{F}(b_t)$ and unsupported refined actions; (4) score the remainder with the distilled policy, using the masked BC distribution when refinement has no supported non-`NOOP` choice and `NOOP` when no non-`NOOP` action is admitted; (5) execute the selected action and update $\mathcal{M}_{t+1}$, $b_{t+1}$, and $w_{t+1}$ from realized cost.

## 3 Related work

**Closest learned memory controllers.** Memory-R1 learns long-term add, update, delete, and skip decisions from task reward, while AtomMem expands the atomic transformation space (Yan et al., 2026; Huo et al., 2026). Agentic Memory jointly trains long- and short-term operations; Verifiable Memory adds local transition and global trajectory verifiers; and MemCon adapts retrieval, reuse, consolidation, and forgetting online (Yu et al., 2026; Sun et al., 2026; Jiang et al., 2026). BudgetWeaver makes three orthogonal choices explicit in one control contract: reads and mutations spend one remaining token–latency vector, read supervision is measured at the native task endpoint, and offline value refinement is admitted only in action-supported regions. Holding candidates, representations, operator implementations, and storage fixed then identifies the value of coordinating those choices rather than conflating it with a new memory backend.

**Retrieval and budget routing.** Fixed and adaptive RAG methods select whether, when, or how much evidence to read (Lewis et al., 2020; Borgeaud et al., 2022; Izacard et al., 2023; Asai et al., 2024; Jiang et al., 2023). BudgetMem makes a sequence of learned low/mid/high compute-tier choices along an on-demand extraction pipeline (Zhang et al., 2026). In BudgetWeaver, allocation spans both evidence access and state transition: `WRITE`, `SUMMARIZE`, and `EVICT` alter the candidates and context priced by later decisions.

**Memory representations and fixed maintenance.** Generative Agents, Reflexion, Voyager, MemGPT, LongMem, and A-MEM couple external state with retrieval, reflection, tier movement, or evolving links (Park et al., 2023; Shinn et al., 2023; Wang et al., 2024; Packer et al., 2023; Wang et al., 2023; Xu et al., 2025). They motivate the backends and maintenance primitives used here. Fixing those components across policies turns operation timing and parameterization under a common resource state into the controlled variable.

**Sequential allocation and support.** Adaptive-depth models allocate internal computation (Elbayad et al., 2020; Raposo et al., 2024); offline RL and doubly robust evaluation address sequential learning from logged interaction (Levine et al., 2020; Kostrikov et al., 2022; Kumar et al., 2020; Dudík et al., 2014; Jiang & Li, 2016). BudgetWeaver instantiates these ideas for external memory transitions through a problem-specific combination: paired endpoint labels price avoidable reads, sequence value carries mutation effects forward, action-conditional support limits offline extrapolation, and predicted-cost masking exposes one remaining resource state to every operation.

# 4 Experimental setup

## 4.1 Benchmarks, splits, and budgets

We evaluate LOCOMO v2 conversational-memory QA, SWE-bench Lite v1.1 software repair, and OdysseyBench v1.3 long-horizon office workflows (Maharana et al., 2024; Jimenez et al., 2024; Wang et al., 2025). The final test contains 1,540 scored LOCOMO questions, 300 SWE issues, and 602 Odyssey workflows. Policy fitting traces, baseline-tuning instances, and final test units are disjoint.

*Protocol summary. Fit and development counts are disjoint logged trajectories; test counts are native benchmark units.*

| Benchmark (unit) | Fit / dev / test | Seeds; evaluated episodes | Native success; latency caps (s) |
|---|---|---|---|
| LOCOMO v2 (QA) | 13,800 / 1,500 / 1,540 | 5; 7,700 | Category-aware answer match; 5.0 / 7.0 / 10.0 |
| SWE-bench Lite v1.1 (issue) | 12,000 / 1,350 / 300 | 5; 1,500 | Official tests resolve; 15.0 / 20.0 / 27.0 |
| OdysseyBench v1.3 (workflow) | 12,000 / 1,350 / 602 | 5; 3,010 | Workflow grader passes; 7.0 / 9.0 / 13.0 |

Tight/medium/loose optional-memory token thresholds are 12k/16k/20k per episode; the corresponding benchmark-specific latency thresholds appear in the protocol table. The medium point is the predeclared primary operating point. The feasibility mask uses both thresholds for admission, while write bytes are carried in state and penalized by the objective. Full-run tokens and end-to-end p95 latency remain outcome measurements of the complete agent run. The logging set contains 42,000 trajectories, split 37,800 / 4,200 between policy fitting and development. Its behavior mixture is 35% always-retrieve / 25% entropy gate / 20% maintenance heuristic / 20% randomized feasible actions; operator frequencies are 41.2 / 28.6 / 10.1 / 6.2 / 6.7 / 4.5 / 2.7% in the interface order above. Support coverage is the fraction of evaluation state–action pairs with a same-action neighbor inside the development-calibrated representation radius.

### 4.2 COMPARATORS, STATISTICS, AND HARDWARE

Each comparator targets one part of the intertemporal-control hypothesis. Always-retrieve tests aggressive immediate access. Entropy, heuristic, and supervised trigger policies adapt reads while retaining shared maintenance rules. The tuned full-interface non-RL comparator uses cost-sensitive boosted action scores and finite-state transition guards over the same four signal families, seven actions, operator parameters, prompts, backends, tuning instances, and thresholds. BC-only retains the sequence architecture but clones logging-mixture actions without IQL refinement or the endpoint-aware reward. Read-only and operator removals measure the contribution of persistent-state transitions beyond selective retrieval. The tuned non-RL comparison identifies the complete learned sequence-controller package; the ablations identify only the components they remove.

We use seeds $\{13, 37, 61, 71, 97\}$, refitting the sequence teacher and distilled MLP for each seed while holding the base LLM, prompts, and memory machinery fixed. Seeds cover initialization, trace order, self-consistency sampling, tool stochasticity, and evaluation. Success is mean$\pm$standard deviation across fitted seeds. For comparator $C$, each native unit contributes $d_i = \frac{1}{5}\sum_s (Y_{is}^{\mathrm{BW}} - Y_{is}^{C})$; paired permutation tests sign-flip $d_i$, paired confidence intervals resample native units, and Holm correction covers the three benchmark comparisons. The paired odds ratio is $n(d_i > 0)/n(d_i < 0)$, omitting ties. Seed standard deviations and unit-level intervals expose fitting and test-unit variation separately; the intervals condition on the five realized fits.

Resource intervals use 10,000 paired native-unit bootstrap replicates after seed averaging. The token effect is the median of within-unit $\mathrm{BW} - C$ token differences. The p95 effect subtracts the comparator p95 from the BudgetWeaver p95 within each resample, so its point estimate need not equal a difference of marginal medians. In Table 3, $\pm$ is the standard deviation across the five seed-wise, equal-benchmark-weight effects. Slice intervals are descriptive paired unit intervals and are not additional confirmatory tests.

Runs use four 80GB NVIDIA A100 accelerators on Ubuntu 22.04 with CUDA 12.4 and PyTorch 2.3.1. The primary base agent is Llama-3.1-70B-Instruct at temperature 0.2 and `top_p`$= 0.9$. The 8B study repeats trace collection, controller fitting, distillation, and evaluation with Llama-3.1-8B-Instruct; it is a cross-scale replication, not direct reuse of 70B policy weights. Embeddings use `bge-large-en-v1.5`, and optional reranking uses `bge-reranker-large`.

## 5 RESULTS

The experiments ask three questions in evidence order: whether endpoint-valued control improves matched end-task decisions, whether persistent mutations matter where later decisions inherit their effects, and whether the ordering persists after an 8B refit and under sparse logged support. Figure 2 consolidates the quality, resource, mechanism, and scale contrasts.

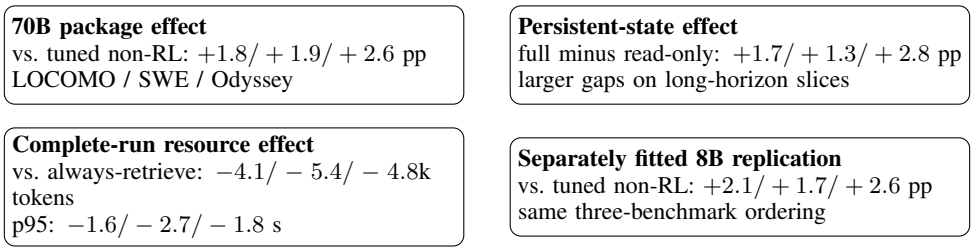


Figure 2: Medium-budget effect summary; triplets are LOCOMO / SWE / Odyssey. Tables 2, 3, and 4 report uncertainty and denominators.

### 5.1 RQ1: DOES ENDPOINT-VALUED CONTROL IMPROVE MATCHED DECISIONS?

Table 1: Comparator map for the shared-budget hypothesis. Tuned non-RL receives the complete interface and budget state; BC-only clones logged actions with the 16-step architecture; BudgetWeaver adds endpoint-aware, support-filtered value refinement. Memory machinery and predicted-cost masking are shared throughout.

| Policy family | Action control | Budget state | Temporal state | Endpoint CF |
|---|---|---|---|---|
| Always-retrieve | fixed read | mask only | – | – |
| Adaptive retrieval gates | learned read | partial | – | varies |
| Tuned full-interface non-RL | all operators | full | 2-step guards | – |
| BC-only | all operators | full | 16-step | – |
| BudgetWeaver | all operators | full | 16-step + value | read labels |

Table 2: Matched-stack outcomes at the medium budget. Panel A reports success (%, mean$\pm$std), the confirmatory endpoint; paired tests use native units and Holm correction. Panel B reports secondary full-run token median [IQR] and end-to-end p95 latency. Every policy uses the 16k token and benchmark-specific latency admission thresholds.

| Benchmark | Always-retrieve | Tuned non-RL | BudgetWeaver |
|---|---|---|---|
| Panel A: task success | | | |
| LOCOMO v2 | $58.6 \pm 1.4$ | $60.5 \pm 1.5$ | $62.3 \pm 1.3$ |
| SWE-bench Lite | $22.4 \pm 1.1$ | $24.2 \pm 1.1$ | $26.1 \pm 1.2$ |
| OdysseyBench | $40.8 \pm 1.6$ | $43.1 \pm 1.7$ | $45.7 \pm 1.8$ |
| Panel B: full-run token median [IQR]; p95 latency (s) | | | |
| LOCOMO v2 | 18.7k [17.1k,20.4k]; 6.9 | 15.0k [13.5k,16.8k]; 5.7s | 14.4k [12.9k,16.1k]; 5.3 |
| SWE-bench Lite | 24.9k [22.6k,27.8k]; 22.1 | 20.6k [18.4k,23.1k]; 20.2s | 19.2k [17.3k,21.6k]; 19.4 |
| OdysseyBench | 20.4k [18.7k,23.2k]; 9.6 | 16.1k [14.4k,18.7k]; 8.4s | 15.3k [13.8k,17.9k]; 7.8 |

Against tuned full-interface non-RL control, BudgetWeaver gains +1.8 pp [0.6,3.0], $p$=0.009, +1.9 pp [0.5,3.3], $p$=0.013, and +2.6 pp [0.9,4.3], $p$=0.006 on LOCOMO, SWE, and Odyssey. Holm-adjusted $p$-values are 0.018 / 0.018 / 0.018, and paired odds ratios are 1.24 / 1.23 / 1.34. Matching the signals, action interface, tuning units, caps, and backends removes those stack-level explanations; together with BC-only, the comparison supports the advantage of the complete endpoint-valued sequence controller over hand-designed coordination.

Relative to always-retrieve, success improves by 3.7, 3.7, and 4.9 points, with paired 95% CIs $[2.1, 5.3]$, $[1.8, 5.6]$, and $[2.4, 7.4]$. Paired full-run token differences are -4.1k [-5.0k, -3.3k], -5.4k [-6.6k, -4.3k], and -4.8k [-5.9k, -3.8k]; p95 differences are -1.6s [-2.2s, -1.0s], -2.7s [-4.0s, -1.5s], and -1.8s [-2.6s, -1.0s]. Thus the endpoint gain does not come from spending more complete-run computation.

### 5.2 RQ2: WHERE DOES MUTATION AVAILABILITY MATTER MOST?

Table 3: Mechanism tests at the medium cap. Panel A averages within-benchmark changes from full BudgetWeaver with equal benchmark weight. Panel B resolves persistent-state ablations by benchmark, exposing variation in the effect of mutation availability.

| Variant | Success effect | Token effect | Decision implication |
|---|---|---|---|
| Panel A: signal and training ablations | | | |
| w/o uncertainty features | $-2.8 \pm 0.6$ pp | $+10.3\%$ | Largest quality and cost loss |
| w/o verifier/tool signals | $-1.7 \pm 0.5$ pp | $+6.8\%$ | Feedback changes memory timing |
| w/o `GRAPH_HOP` | $-1.3 \pm 0.4$ pp | $-1.9\%$ | Structured reads trade cost for quality |
| w/o `SUMMARIZE` | $-0.8 \pm 0.3$ pp | $+3.2\%$ | Compression preserves later capacity |
| w/o budget reward term | $+0.4 \pm 0.3$ pp | $+23.6\%$ | Reward shapes pre-cap allocation |
| BC-only | $-2.0 \pm 0.5$ pp | $+8.1\%$ | Endpoint-aware refinement adds lift |
| Panel B: success by mutation availability (LOCOMO / SWE / Odyssey) | | | |
| BudgetWeaver | $62.3 \pm 1.3$ | $26.1 \pm 1.2$ | $45.7 \pm 1.8$ |
| w/o `WRITE` | $61.4 \pm 1.4$ | $25.7 \pm 1.2$ | $44.3 \pm 1.8$ |
| w/o `EVICT` | $61.8 \pm 1.3$ | $25.9 \pm 1.2$ | $44.8 \pm 1.7$ |
| Read-only | $60.6 \pm 1.5$ | $24.8 \pm 1.2$ | $42.9 \pm 1.8$ |

Uncertainty is the largest common signal contributor. BC-only loses $2.0 \pm 0.5$ points while using 8.1% more tokens; because this comparison removes the complete endpoint-aware refinement objective, it supports that refinement as a package rather than isolating counterfactual labels from delayed value. Persistent-state availability is directly isolated by read-only control: removing all mutations costs 1.7 / 1.3 / 2.8 points on LOCOMO, SWE, and Odyssey. `WRITE` appears in 29.7% / 10.8% / 36.4% of episodes, totaling 3,984 / 247 / 2,116 actions; `EVICT` appears in 17.9% / 6.3% / 25.8%, totaling 1,906 / 142 / 1,231 actions. The larger LOCOMO and Odyssey effects coincide with more frequent mutations and longer-lived state, whereas SWE obtains more of its gain from selective reads and summaries.

The prespecified descriptive slices sharpen that pattern: full BudgetWeaver exceeds read-only by +3.6 pp [1.7,5.5] on contradiction-heavy LOCOMO episodes, +5.1 pp [2.5,7.7] on the longest Odyssey workflows, and +1.8 pp [0.3,3.3] on SWE's longest-horizon slice. These are within-slice estimates rather than slice-versus-complement interaction tests. Appendix B.1 reports all nine slices, their sample sizes, and mutation-active rates.

### 5.3 RQ3: CROSS-SCALE REPLICATION AND THE SUPPORT BOUNDARY

Table 4: Separately fitted Llama-3.1-8B-Instruct replication under unchanged prompts, backends, operators, development protocol, seeds, and medium caps. Cells are success (%, mean$\pm$std); the final column is the paired BudgetWeaver effect.

| Benchmark | Always-retrieve | Tuned non-RL | BudgetWeaver | Paired effect |
|---|---|---|---|---|
| LOCOMO v2 | $52.7 \pm 1.8$ | $55.0 \pm 1.7$ | $57.1 \pm 1.6$ | +2.1 pp, $p$=0.011 |
| SWE-bench Lite | $16.9 \pm 0.9$ | $18.6 \pm 1.0$ | $20.3 \pm 1.1$ | +1.7 pp, $p$=0.026 |
| OdysseyBench | $34.8 \pm 1.9$ | $37.2 \pm 1.8$ | $39.8 \pm 1.7$ | +2.6 pp, $p$=0.008 |

Refitting at 8B lowers absolute success but preserves every policy ordering; Holm-adjusted $p$-values are 0.024 / 0.026 / 0.024. Relative to always-retrieve, 8B BudgetWeaver reduces full-run tokens by 20.9% [17.2,24.6], 18.3% [14.1,22.5], and 23.4% [19.0,27.8], with p95 latency reductions of 18.7% [12.9,24.5], 9.8% [4.0,15.6], and 16.2% [10.1,22.3]. The replication shows that the package-level ordering is not confined to the 70B uncertainty profile.

On SWE, BC-to-RL gains track logged support: +2.6 pp [1.1,4.1] for $n$=132 high-support tasks, +1.5 pp [0.2,2.8] for $n$=119 medium-support tasks, and +0.3 pp [-0.9,1.5] for $n$=49 low-support

tasks. The 12.4% fallback rate prevents unsupported refined choices; the count-weighted effect is 1.8 points.

## 6 DISCUSSION

The matched comparisons identify a quality–resource improvement in the controller itself. Always-retrieve tests maximal feasible access, and tuned non-RL tests hand-designed coordination over the complete signal and operator interface. BudgetWeaver wins both comparisons on every benchmark while using fewer full-run tokens and less tail latency than always-retrieve. Because retrievers, stores, candidates, prompts, guards, and operator implementations do not change, backend quality and candidate exposure cannot explain this ordering.

The mechanism results explain why the effect varies by task. LOCOMO and Odyssey invoke writes and evictions more often and lose 1.7 and 2.8 points when persistent mutations are disabled; their largest descriptive read-only gaps occur in contradiction-heavy or longest-horizon episodes. SWE loses 1.3 points without mutations and relies more heavily on selective reads, summaries, and support fallback. BC-only establishes the joint value of endpoint-aware offline refinement, while the read-only and individual-operator comparisons separately identify persistent-state availability. Together, these interventions support downstream state value without assigning the BC-only gap to any single refinement component.

The learned ordering also persists when the complete protocol is refit around the 8B base model. Runtime budget compliance is a measured systems property rather than a theorem of the predicted-cost mask: the accounting wrapper records realized consumption, and the 400-episode adaptive-waste stress test observes zero overruns while BudgetWeaver limits in-envelope token inflation and success loss relative to tuned non-RL control (Appendix C).

The claim is calibrated to the predeclared medium operating point across the three benchmark families. The LOCOMO tight/loose sweep maps the adjacent quality–resource frontier, and the five fitted seeds plus native-unit intervals report complementary sources of variation. The evaluated regime therefore establishes that endpoint-valued read–mutation control can dominate aggressive access and full-interface hand-designed coordination on a fixed memory stack; broader cap frontiers and model families would test how far that ordering extends.

## 7 CONCLUSION

BudgetWeaver turns the future memory state created by retrieval, reranking, traversal, summarization, writing, and eviction into the object of control. At the predeclared medium budgets, it gains 1.8–2.6 success points over tuned full-interface non-RL coordination and 3.7–4.9 points over always-retrieve across conversational QA, software repair, and office workflows. The latter comparison also reduces paired full-run tokens by 4.1k–5.4k and p95 latency by 1.6–2.7 seconds. Separately fitted 8B controllers retain a 1.7–2.6-point lead, while disabling persistent mutations costs 1.3–2.8 points and hurts most in long-lived, contradiction-heavy state. The resulting design principle is concrete: when memory operations share resources, optimize the sequence of memory states inherited by future decisions, not each operation's immediate utility in isolation.

## REPRODUCIBILITY STATEMENT

Sections 2 and 4 specify the operator contract, cap semantics, model and backend configuration, training objective, splits, seeds, statistics, and hardware. Appendices A–C provide comparator settings, exact test accounting, slice definitions, support analysis, and systems stress evidence.

## AI USE STATEMENT

Generative AI assistants were used during manuscript preparation for language editing, presentation, formatting, and consistency checks. The authors reviewed the final manuscript and take responsibility for its contents.

## REFERENCES


Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240, 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

Miroslav Dudík, Dumitru Erhan, John Langford, and Lihong Li. Doubly robust policy evaluation and optimization. *Statistical Science*, 29(4):485–511, 2014. doi: 10.1214/14-STS500. URL https://doi.org/10.1214/14-STS500.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJg7KhVKPH.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Hkg4TI9xl.

Yupeng Huo, Yaxi Lu, Zhong Zhang, Haotian Chen, and Yankai Lin. AtomMem: Learnable dynamic agentic memory with atomic memory operation. *arXiv preprint arXiv:2601.08323*, 2026. URL https://arxiv.org/abs/2601.08323.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43, 2023. URL https://jmlr.org/papers/v24/23-0037.html.

Eric Hanchen Jiang, Zhi Zhang, Yuchen Wu, Levina Li, Dong Liu, Xiao Liang, Rui Sun, Yubei Li, Edward Sun, Haozheng Luo, Zhaolu Kang, Aylin Caliskan, Kai-Wei Chang, and Ying Nian Wu. Memory as a controlled process: Learned adaptive memory management for LLM agents. *arXiv preprint arXiv:2607.13591*, 2026. URL https://arxiv.org/abs/2607.13591.

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 652–661, 2016. URL https://proceedings.mlr.press/v48/jiang16.html.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, 2023. doi: 10.18653/v1/2023.emnlp-main.495. URL https://aclanthology.org/2023.emnlp-main.495/.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=EblVBDNalKu.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1179–1191, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/0d2b2061826a5df3221116a5085a6052-Abstract.html.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6402–6413, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. URL https://arxiv.org/abs/2005.01643.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

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. URL https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024. URL https://arxiv.org/abs/2404.02258.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Xiaolong Sun, Qichao Wang, Hangyu Li, and Liang Chen. Verifiable memory: Learning unified memory management with local and global verifiers for large language model agents. *arXiv preprint arXiv:2608.03137*, 2026. URL https://arxiv.org/abs/2608.03137.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Weixuan Wang, Dongge Han, Daniel Madrigal Diaz, Jin Xu, Victor Rühle, and Saravan Rajmohan. OdysseyBench: Evaluating LLM agents on long-horizon complex office application workflows. *arXiv preprint arXiv:2508.09124*, 2025. URL https://arxiv.org/abs/2508.09124.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/ebd82705f44793b6f9ade5a669d0f0bf-Abstract-Conference.html.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825, 2026. doi: 10.18653/v1/2026.acl-long.583. URL https://aclanthology.org/2026.acl-long.583/.

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. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21457–21483, 2026. doi: 10.18653/v1/2026.acl-long.981. URL https://aclanthology.org/2026.acl-long.981/.

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. *arXiv preprint arXiv:2602.06025*, 2026. URL https://arxiv.org/abs/2602.06025.

# A TRAINING, COUNTERFACTUALS, AND COMPARATOR SPECIFICATION

The tuned full-interface non-RL search evaluates 384 configurations under the same development cap and objective used for other tunable baselines. Its selected controller uses depth-4 boosted scores, 180 trees, learning rate 0.05, an 18% reserve guard, and a two-step cooldown; finite-state guards prevent incompatible successive mutations, while action scores select among all feasible operators. No test instance contributes to search or selection.

Counterfactual pairs serialize the complete pre-step agent, tool, memory, and cap state. The first branch forces the logged read and the second forces `NOOP`; both then use the frozen behavior-mixture continuation. Common random numbers align model samples and tool events, and both branches run through the native episode endpoint. This same procedure defines `BadRetrieval` and trigger labels, preventing label/objective drift.

The LOCOMO trigger registry is: entropy threshold 0.734 / 0.491 / 0.127, heuristic trigger 0.781 / 0.564 / 0.103, learned classifier 0.806 / 0.612 / 0.088, and BudgetWeaver 0.847 / 0.673 / 0.064 (AUROC / precision at 80% recall / ECE). Step-index and signal-shuffle controls reach $58.9 \pm 1.5\%$ ($p$=0.004) and $57.4 \pm 1.7\%$ ($p$=0.001) versus BudgetWeaver, respectively. These controls show that aligned episode state drives action selection beyond schedule position or arbitrary channels.

LOCOMO operating points are 60.1±1.4% success, 11.4k median full-run tokens, 4.7s end-to-end p95, and 91.2% cap-pass under the tight cap; 62.3±1.3%, 14.4k, 5.3s, and 95.6% at medium; and 62.8±1.2%, 17.2k, 6.1s, and 97.8% at loose. The medium point captures most of the observed success increase while retaining lower full-run cost than the loose setting.

# B EXACT ACCOUNTING AND SUPPORT SLICES

Table 5: Exact test accounting across five seeds. Each accounting cell is task count / evaluated episodes / successes / failures / cap-clipped episodes. Counts aggregate the final test runs without changing the seed-averaged success estimates in Table 2.

| Benchmark | Always-retrieve | Tuned non-RL | BudgetWeaver |
|---|---|---|---|
| LOCOMO | 1,540 / 7,700 / 4,512 / 3,188 / 1,086 | 1,540 / 7,700 / 4,659 / 3,041 / 493 | 1,540 / 7,700 / 4,797 / 2,903 / 339 |
| SWE | 300 / 1,500 / 336 / 1,164 / 276 | 300 / 1,500 / 363 / 1,137 / 132 | 300 / 1,500 / 392 / 1,108 / 87 |
| Odyssey | 602 / 3,010 / 1,228 / 1,782 / 482 | 602 / 3,010 / 1,297 / 1,713 / 211 | 602 / 3,010 / 1,376 / 1,634 / 139 |

Paired win/tie/loss counts for BudgetWeaver versus always-retrieve are 179 / 1,239 / 122, 39 / 233 / 28, and 75 / 482 / 45. Against tuned non-RL they are 143 / 1,282 / 115, 32 / 242 / 26, and 63 / 492 / 47. These discordant-pair counts yield the paired odds ratios reported in Sec. 5.

## B.1 PREDEFINED MUTATION-SENSITIVE SLICES

*Full BudgetWeaver minus read-only success difference. Parentheses give slice size, 95% CI, and mutation-active episode rate.*

| Slice | LOCOMO | SWE | Odyssey |
|---|---|---|---|
| Memory-dense | +2.8 pp ($n$=388, [1.3,4.3], 42.8%) | +1.1 pp ($n$=72, [-0.2,2.4], 15.7%) | +4.1 pp ($n$=151, [2.0,6.2], 48.6%) |
| Contradiction-heavy | +3.6 pp ($n$=246, [1.7,5.5], 47.3%) | +1.4 pp ($n$=58, [0.0,2.8], 18.4%) | +4.8 pp ($n$=112, [2.3,7.3], 53.2%) |
| Longest-horizon | +3.2 pp ($n$=310, [1.4,5.0], 45.6%) | +1.8 pp ($n$=60, [0.3,3.3], 19.8%) | +5.1 pp ($n$=121, [2.5,7.7], 56.4%) |

Memory-dense denotes the prespecified top quartile of pre-action density, contradiction-heavy requires at least one unresolved contradiction before the decision, and longest-horizon is the top quintile by native episode steps. Slice membership is fixed before outcome comparison.

## B.2 OFFLINE SUPPORT ON SWE-BENCH LITE

State overlap ranges from 79–89% on LOCOMO and 75–86% on Odyssey, with positivity violations of 2.4–4.8% and 3.1–7.6%. SWE overlap is 71–83%, positivity violations are 5.3–11.7%, and OPE standard deviation is 1.0–1.9 points. On SWE, operator-specific positivity violations are 3.8 / 5.1 / 8.7 / 10.9 / 6.4 / 13.6 / 15.2% in interface order; support-aware fallback reduces the low-support success effect from -0.4 pp to +0.3 pp and changes full-run tokens by -3.1%. This pattern connects the 12.4% fallback rate to stable low-support behavior.

# C RUNTIME AND WRITE-PATH STRESS EVIDENCE

The base model, backend integrity, and accounting wrapper are trusted. Prompts and tool outputs may be adversarial. Cap compliance and write filtering are empirical properties evaluated on held-out stress cases.

Table 6: Held-out runtime and write-path stress matrix. Rows pair each protected object with its trust boundary, matched comparator, observed utility, and residual failure.

| Object; adversary | Trust; property | Comparator; held-out result | Utility, failure, evidence |
|---|---|---|---|
| Optional-memory budget; operator-manipulation and adaptive waste prompts | Trusted accounting wrapper; empirical zero realized overrun | Tuned non-RL; BudgetWeaver inflation 13.6%, success -1.9 pp, versus 34.2%, -5.7 pp | 400 episodes / 0 overruns; residual in-cap inflation; adaptive aggregate and 62.1 LOCOMO stress success |
| Persistent memory; malicious tool output and contradictory writes | Trusted backend and typed gate; harmful-write rejection | No-guard writes; accepted 14/520 (2.7%) versus 47/520 (9.0%) | Benign rejection 8/510 (1.6%); residual accepted payloads; guarded-write aggregate |
| Latency budget; $2\times$ tool slowdown | Trusted clock and accounting wrapper; empirical cap compliance | Tuned non-RL; predicted-cost mask under slowdown | 61.4 LOCOMO success, 3.4% SLA incidents; slowdown stress trace |
| Benign writes; ordinary verified tool output | Trusted verifier interface; schema-valid persistence | Unguarded schema path; held-out benign writes | 98.4% accepted; measured guard rejection; benign-write aggregate |

Additional LOCOMO perturbations preserve the medium-cap interpretation: success is 61.6 under minor tool drift, 59.8 under major drift, 61.4 under doubled tool latency, 60.7 with 10% injected tool errors, and 62.1 under operator-manipulation prompts, compared with 62.3 at the standard operating point. Incident rates span 0.6% forced operations to 4.7% SLA violations. Under adaptive budget-waste prompts, the held-out stress set records zero realized overruns, while BudgetWeaver limits the quality and resource damage observed within the admitted envelope.