

ReserveGate: Reserve-Aware Evidence Scheduling for Long-Horizon Language Agents

Shuang Cao* and Rui Li*,† and Louise Ruihua Liu and Alexandre Duprey

Hill Research

619 Alexander Road, Princeton, NJ 08540, USA

sharon.cao@hillresearch.ai rui.li@hillresearch.ai

louise.liu@hillresearch.ai alexandre.duprey@hillresearch.ai

*Equal contribution. †Corresponding author

Abstract

Repeated retrieval and verification create a shared-cap scheduling problem in long-horizon language agents: each evidence action can resolve current uncertainty while consuming tokens and tool time required by later constraints. We introduce ReserveGate, an episode-level token-latency scheduler that performs constrained one-step policy improvement from an auditable off-policy advantage estimate. Component-wise masks preserve both hard resources, a consumption-adaptive price reacts to overspending, a reserve released with episode progress protects late-stage capacity, and overlap-aware fallback guards unsupported decisions. Across three controlled agent environments, ReserveGate improves tight-budget success by 2.6–3.8 percentage points over the strongest matched controller and by 4.6–6.2 points in the longest episode quartile. With full-system costs charged, ReserveGate preserves this success ordering while reducing end-to-end tokens by 5.5–6.6%, p99 tool time by 5.9–9.2%, and service-level incidents by 1.8–2.5 points relative to that controller. A matched LoCoMo replication across two generators and two retrievers preserves the gains, which also remain above tuned rate limiting and task-agnostic state encoders. Together, these results establish evidence timing as a distinct control layer for improving the success-resource frontier of long-horizon agents.

1 Introduction

Language agents increasingly interleave generation with retrieval, tool calls, verification, and memory updates (Yao et al., 2023; Schick et al., 2023; Asai et al., 2024). Larger context windows expand storage, while retrieval text, reranking, verification traffic, external-service calls, retries, timeouts, and

heavy-tailed latency remain metered operations. When dozens or hundreds of evidence decisions draw from shared episode-level limits, spending becomes intertemporal: an early call changes which later calls remain feasible. Across our three environments, scheduling this competition raises success while reducing end-to-end tokens, p99 tool time, and service-level agreement (SLA) incidents. The advantage reaches 4.6–6.2 percentage points in the longest quartile, where each spending decision can affect many downstream opportunities.

Adaptive retrieval and routing methods primarily optimize evidence for the current query or generation step (Jiang et al., 2023; Jeong et al., 2024; Su et al., 2024; Wang et al., 2026). Long-context and memory systems improve evidence representation and access (Chen et al., 2023; Hu et al., 2025), while cost-aware memory routing selects among memory modules (Zhang et al., 2026). ReserveGate targets the cross-step coupling created when repeated, heterogeneous retrieval and verification actions compete for the same hard token and tool-time caps. Its decision rule joins step-local information value with episode-global feasibility: a learned action advantage ranks evidence actions, separate masks enforce both resources without exchanging tokens for seconds, a consumption-adaptive price responds to realized overspending, and a gradually released reserve represents the intertemporal value of late-stage capacity.

At each decision point, ReserveGate updates a compact four-component uncertainty sketch and estimates the terminal-success advantage of each feasible action over NOOP under a fixed continuation policy. The estimator is trained by source-group cross-fitted doubly robust regression, exposes effective sample size (ESS), and invokes a conservative

trigger when logged support is weak. Repeated application implements an episode-level scheduling policy with 0.21–0.29 ms controller overhead per decision.

Our matched comparison fixes the base model, prompts, candidate pool, retriever, tools, verifier, realizations, and caps, thereby identifying the scheduler effect. The complementary full-accounting comparison charges prompts, generation, retrieval, reranking, retries, and timeouts. Against the strongest same-environment controller, ReserveGate gains 2.6–3.8 points in success while reducing full-accounting tokens by 5.5–6.6%; a matched LoCoMo study preserves the ordering across two generators and two retrievers.

The contributions are three technical advances with direct consequences. (1) *Shared-cap evidence scheduling*: a component-wise token–latency feasible set, adaptive resource price, and progress-released reserve allocate evidence capacity across an episode. (2) *Auditable policy improvement*: a source-group cross-fitted action-value estimator, overlap-aware fallback, and forced-action validation connect each deployed ranking to realized intervention direction. (3) *Mechanism-grounded systems evidence*: matched and full accounting, a strongly tuned rate limiter, horizon and schedule interventions, cross-stack replications, portability measurements, and tail-risk analysis show that scheduling improves both success and resource use when spending decisions compound.

2 Problem Setting

At decision point t , an agent observes o_t and chooses a task action together with an optional evidence action $m_t \in \mathcal{M}$. The decision object is the sequence of evidence actions across the episode: every issued action changes the two-dimensional budget available to all subsequent decisions. The primary action set is $\mathcal{M} = \{\text{NOOP}, \text{RETRIEVE}(k), \text{VERIFY}\}$, where $k \in \{8, 16, 32, 64\}$. SUMMARIZE and WRITE extend this interface in Appendix F, with action-specific support reported separately.

Each action has a two-dimensional estimated cost

$$\hat{\mathbf{c}}_t(m) = (\hat{c}_t^{\text{tok}}(m), \hat{c}_t^{\text{time}}(m)),$$

and the remaining per-episode budget is $\mathbf{B}_t = (B_t^{\text{tok}}, B_t^{\text{time}})$. Feasibility is enforced component-wise; tokens and seconds are never treated as interchangeable constraints. For ranking feasible ac-

tions, we use the dimensionless scalarization

$$\bar{C}_t(m) = \omega_{\text{tok}} \frac{\hat{c}_t^{\text{tok}}(m)}{B_0^{\text{tok}}} + \omega_{\text{time}} \frac{\hat{c}_t^{\text{time}}(m)}{B_0^{\text{time}}}, \quad (1)$$

where $\omega_{\text{tok}} + \omega_{\text{time}} = 1$. The weights are selected on development data and frozen for test. The core comparison models token and tool-time resources; storage cost for the optional WRITE action is reported separately.

We measure task success, end-to-end tokens, tool-time percentiles, and service-level agreement (SLA) incident rate. The matched comparison uses hard evidence-action caps to identify the controller effect, while the equally important full-accounting comparison includes prompt, generation, retrieval, reranking, and verification costs.

Formally, for a trajectory τ with terminal score $Y(\tau) \in [0, 1]$, the controller solves the constrained surrogate problem

$$\begin{aligned} \max_{\pi} \quad & \mathbb{E}_{\tau \sim \pi}[Y(\tau)] \\ \text{s.t.} \quad & \sum_t \hat{c}_t^j(m_t) \leq B_0^j, \quad j \in \{\text{tok}, \text{time}\}. \end{aligned} \quad (2)$$

The token constraint is enforced from exact tokenizer counts before an action is issued. Tool time is stochastic, so feasibility uses a calibrated upper-quantile estimate; an SLA incident records the residual event in which realized cumulative tool time exceeds its cap or a tool times out. Equation 2 defines the online decision boundary, and the reported tail metrics directly quantify residual cost-estimation failures.

The evidence controller leaves the agent’s task action, prompt, and candidate pool fixed. This separation creates two complementary evaluation views. The *matched* view fixes the surrounding agent and charges costs causally affected by m_t , which identifies differences among schedulers. The *full-accounting* view charges every prompt, generated token, retrieval, reranking pass, verification call, retry, and timeout, which tests whether the isolated advantage survives at the system boundary.

3 ReserveGate

3.1 Uncertainty sketch

The uncertainty sketch z_t tracks four latent components: (i) a categorical distribution over unresolved constraint types; (ii) a Beta posterior for tool reliability; (iii) the probability that required evidence is still missing; and (iv) a three-way episode-phase

distribution. The controller augments these four beliefs with observed resource state and recent-action history; these are measured covariates, not additional latent components. The controller state is

$$s_t = [p_t(c), \mathbb{E}_t[\rho], \text{Var}_t[\rho], p_t(f=1), p_t(\psi), \mathbf{B}_t/\mathbf{B}_0, t/\hat{H}, h_t^{\text{ret}}, h_t^{\text{ver}}]. \quad (3)$$

Here $\hat{H}$ is an estimated episode horizon and $h_t^{\text{ret}}, h_t^{\text{ver}}$ summarize recent evidence outcomes.

Categorical variables use development-frozen add-one likelihood tables estimated on training logs:

$$p_t(x = q) \propto p_{t-1}(x = q) \cdot \frac{N_{\text{train}}(o_t, x = q) + 1}{N_{\text{train}}(x = q) + |\mathcal{O}_x|}.$$

For a verifier outcome $v_t \in \{0, 1\}$, the reliability posterior is Beta-Bernoulli:

$$p(\rho \mid v_{1:t}) \propto \rho^{\sum_i v_i} (1 - \rho)^{t - \sum_i v_i} \text{Beta}(\rho; 2, 2).$$

The symmetric Beta(2, 2) prior contributes two pseudo-observations, regularizing sparse early outcomes without preferring success or failure; its strength and categorical smoothing are fixed before development evaluation. The latent structure is shared across tasks; task adaptation consists of mappings from observable log fields to the four sketch components. Appendix E reports mapping counts, construction time, and an LLM-assisted mapping audit.

Table 1 makes the construction explicit. Constraint mass is redistributed when the parser detects an unresolved identity, temporal, tool, or precondition requirement. Tool reliability is updated after an observable success, failure, timeout, or verifier judgment. Missing-evidence probability uses retrieval coverage, source agreement, and unsupported-claim flags, while phase combines normalized progress with the horizon predictor. All updates are causal with respect to the current decision: no terminal outcome or future trace field enters s_t .

3.2 One-step action-value proxy

Let π_c denote the development-frozen behavior mixture used after the current intervention in logged and cloned rollouts. Following information-value theory (Howard, 1966), the marginal value of a feasible evidence action is its effect on terminal success under a common continuation policy. The

Component	Representation	Observable update signals
Constraint	categorical mass	parser flags, unmet slots, tool preconditions
Reliability	Beta posterior	success, failure, timeout, verifier agreement
Missing evidence	Bernoulli probability	coverage, novelty, source agreement
Episode phase	categorical mass	normalized step, predicted remaining horizon
Resource state	normalized vector	tokens, tool time, recent action outcomes

Table 1: The shared uncertainty sketch and task-specific observation interface.

corresponding one-step estimand is

$$Q^{\pi_c}(s, m) = \mathbb{E}[Y \mid S_t = s, \text{do}(M_t = m), M_{t+1:} \sim \pi_c].$$

We learn $\hat{Q}_\theta(s, m) \approx Q^{\pi_c}(s, m)$, where Y is terminal task success, and define

$$\hat{A}_\theta(s, m) = \hat{Q}_\theta(s, m) - \hat{Q}_\theta(s, \text{NOOP}).$$

Training uses source-group cross-fitting and a doubly robust pseudo-target (Dudík et al., 2014; Jiang and Li, 2016):

$$\tilde{Y}_i(m) = \hat{\mu}^{-k}(s_i, m) + \frac{\mathbb{I}[m_i = m]}{\max(\epsilon, \hat{\pi}_0^{-k}(m \mid s_i))} \cdot (Y_i - \hat{\mu}^{-k}(s_i, m_i)), \quad (4)$$

where $-k$ denotes nuisance models fit without source-group fold k . Importance weights are clipped and self-normalized. We report effective sample size (ESS) per action and fall back to a tuned uncertainty trigger when ESS is below a development-selected threshold. Identification uses consistency between the logged and forced action, conditional exchangeability given the causal pre-action state s_i , overlap after the prespecified propensity clip, source-group cross-fitting, and the fixed continuation policy π_c . The logged propensity calibration and action-level ESS assess overlap, groupwise folds prevent source leakage, the fallback avoids weak-support extrapolation, and cloned forced-action tests directly audit the resulting intervention ranking.

Each logged decision is a pseudo-target unit, while source conversation, API family, or repository is the cross-fitting and inference unit. We fit nuisance models without the held-out source-group

fold and train the final action-value ensemble on the concatenated out-of-fold targets, so adjacent decisions from one episode never cross folds. The deployed advantage is the ensemble mean; ensemble dispersion enters the fallback rule without changing the score. Thus $\hat{A}_\theta$ operationalizes information value as the terminal-success consequence of taking one feasible evidence action and then returning to π_c . Applying the improvement rule at every reached state converts these local marginal values into the deployed episode-level scheduling policy.

Policy-improvement bound. For an estimated feasible set $\hat{\mathcal{F}}_t$, suppose $|\hat{A}(s_t, m) - A(s_t, m)| \leq \epsilon$ for every $m \in \hat{\mathcal{F}}_t$. If $\hat{m}$ maximizes $\hat{A}(s_t, m) - \lambda_t \bar{C}_t(m)$ on this set and m^* maximizes the corresponding true one-step utility, then

$$\begin{aligned} & [A(s_t, m^*) - \lambda_t \bar{C}_t(m^*)] \\ & - [A(s_t, \hat{m}) - \lambda_t \bar{C}_t(\hat{m})] \leq 2\epsilon. \end{aligned}$$

The calibrated upper-quantile time estimate further gives a per-decision exceedance probability at most $1 - q$ on the calibration distribution for actions admitted by the time mask. Appendix B states the assumptions and proof.

3.3 Adaptive reserve-aware decision rule

Token cost is counted exactly with the frozen generator tokenizer before an action is issued. Tool-time cost uses a conditional $q = 0.95$ quantile model trained on source-group training folds from action type, candidate count, interface, recent latency, retry state, and tool outcome history; split-conformal residual calibration is fit on development groups and frozen for test. Retries and timeouts enter as realized censored-at-timeout durations, so the upper quantile reflects their observed tail mass. Equation 1 normalizes costs for ranking, while component-wise feasibility continues to enforce raw tokens and seconds separately.

Let $u_t = t/\hat{H}$ be normalized progress and let

$$d_t = \max_{j \in \{\text{tok}, \text{time}\}} \left[1 - \frac{B_t^j}{B_0^j} - u_t \right]_+$$

measure how far spending is ahead of schedule. The cost weight adapts as $\lambda_t = \lambda_0(1 + \gamma d_t)$. The protected reserve is $\mathbf{R}_t = r_t \mathbf{B}_0$, where $r_t = r_{\max} \max(0, 1 - 3u_t/2)$; it decays to zero at the start of the final third. The feasible set is

$$\mathcal{F}_t = \{m : \hat{c}_t(m) \preceq \mathbf{B}_t - \mathbf{R}_t\} \cup \{\text{NOOP}\},$$

Algorithm 1 ReserveGate at one evidence decision.

- 1: Update uncertainty sketch z_t and construct s_t (Eq. 3)
 - 2: Predict $\hat{c}_t(m)$ and $\hat{A}_\theta(s_t, m)$ for each $m \in \mathcal{M}$
 - 3: Compute overspending d_t , cost weight λ_t , and reserve $\mathbf{R}_t$
 - 4: Mask actions that violate either resource dimension or the reserve
 - 5: If action-level ESS is too small, return the tuned uncertainty trigger
 - 6: Otherwise return the feasible action maximizing Eq. 5
-

and the deployed rule is

$$m_t = \arg \max_{m \in \mathcal{F}_t} [\hat{A}_\theta(s_t, m) - \lambda_t \bar{C}_t(m)]. \quad (5)$$

The reserve supplies limited intertemporal structure to an otherwise myopic controller. Early in an episode, an action must remain feasible after preserving $\mathbf{R}_t$; reserve release is determined by normalized progress u_t , becoming fully available in the final third. The horizon predictor is a training-fold quantile regressor over task schema and the first two observable steps; it is frozen before development selection and test evaluation. Horizon underestimation releases reserve early, but the consumption multiplier and hard masks still react to realized spending; overestimation retains reserve longer, and a missing estimate activates the conservative trigger. The adaptive multiplier reacts when spending has outpaced progress, avoiding a blanket penalty on difficult episodes that legitimately need early retrieval. This inexpensive approximation to intertemporal resource value responds to realized consumption in both resource dimensions while preserving an inspectable decision rule.

Inference cost and fail-safe behavior. All candidate actions are scored in one network batch and the feasibility mask is deterministic given the cost estimates. Controller overhead is 0.21, 0.27, and 0.29 ms per decision on LoCoMo-Episodic, StableToolBench-PO, and SWE-bench-Verified-420, respectively. If no non-null action is feasible, if the horizon estimate is missing, or if action-level ESS falls below the frozen threshold, the controller returns either NOOP or the tuned uncertainty trigger without modifying the task policy.

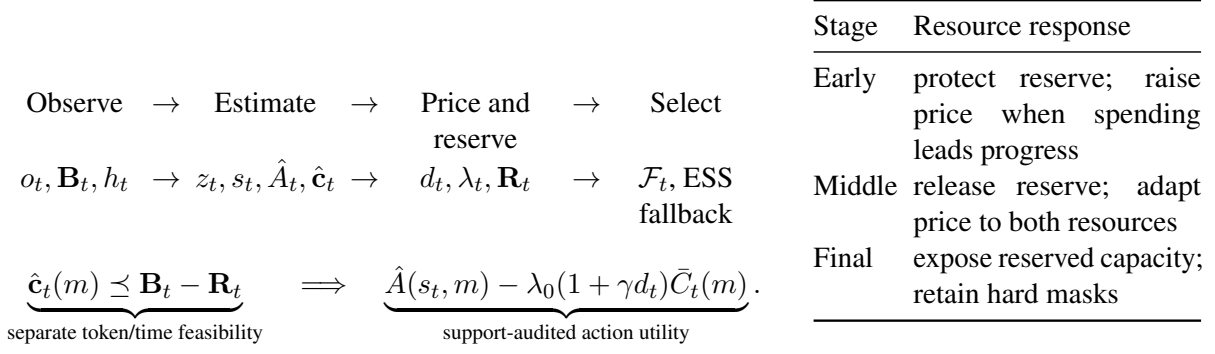


Figure 1: ReserveGate closes the loop between uncertainty, action value, and episode resources. Consumption-responsive pricing and progress-based reserve release couple the current decision to later feasibility while token and time constraints remain separate.

4 Experimental Design

4.1 Benchmark-derived environments

We construct three controlled benchmark-derived agent environments and interpret controller comparisons within each environment. LOCOMO-EPISODIC derives 2,400 multi-session evidence episodes from LoCoMo (Maharana et al., 2024). Each episode begins at a sampled conversational checkpoint and contains a sequence of questions whose required facts span sessions. Because LoCoMo contains 10 source conversations, we use five grouped folds with 6/2/2 train/development/test conversations; models are refit per fold and every conversation appears in test exactly once. STABLETOOLBENCH-PO derives 800 partial-observability episodes from StableToolBench (Qin et al., 2024; Guo et al., 2024). The wrapper hides selected API state and intermediate tool outputs until queried, while preserving the official executable tool goal; API families, rather than individual calls, define split groups. Its 40 API families form ten grouped folds with 32/4/4 train/development/test families, so every family appears in test exactly once. SWE-BENCH-VERIFIED-420 uses 420 reproducible instances from SWE-bench Verified (Jimenez et al., 2024). During subset construction, we exclude 37 non-deterministic setup failures, 28 network-dependent tasks, and 15 tasks requiring unavailable services. The retained set spans all 12 source repositories; we use six grouped folds with 8/2/2 train/development/test repositories, and every repository appears in test exactly once.

Table 2 reports the exact unit of evaluation and grouping key. Rollout construction occurs after grouping, and all controller folds inherit these

groups. Consequently, each test conversation, API family, and repository remains entirely outside the value model’s training folds. Wrapper versions, split identifiers, exclusion codes, and source revisions are fixed before test evaluation and recorded for each run.

4.2 Stacks, budgets, and accounting

The primary stack uses Llama-3.1-70B-Instruct, bge-large-en-v1.5, a shared top-64 hierarchical navigable small world (HNSW) candidate pool, five environment seeds, and five stochastic roll-outs per episode-seed. Retrieval actions select $k \in \{8, 16, 32, 64\}$ from that fixed pool; verification prompts and tool versions are identical across controllers. The cross-stack study substitutes Qwen2.5-72B-Instruct and e5-large-v2 without refitting test-time thresholds. All inference uses 8 NVIDIA H100 80GB GPUs with vLLM 0.6.3; controller fitting and the complete evaluation consume approximately 1,184 GPU-hours. Stack parity makes the scheduling contrast causal: each same-stack controller receives identical generation and retrieval capabilities.

The main tight caps are 12k evidence-action tokens and 6.0 seconds of cumulative tool time; medium caps are 16k and 8.0 seconds. The scarcity frontier additionally evaluates severe 8k/4s and relaxed 24k/12s evidence-action caps. These caps isolate the resources controlled by evidence actions, including retrieved text inserted into context and verifier traffic, independently of the generator’s maximum context length. For full accounting, we additionally count the system prompt, retained dialogue, task-action generation, reranker input and output, retries, and failed calls. This broader measurement boundary leaves the evidence-action fea-

Environment	Train	Dev	Test	Median decisions [IQR]	Split group	Terminal success
LoCoMo-Episodic [†]	1,440	480	480	84 [58, 112]	conversation	evidence-grounded answer
StableToolBench-PO	640	80	80	126 [91, 168]	API family	executable tool goal
SWE-bench-Verified-420 [‡]	280	70	70	48 [31, 66]	repository	target tests pass

Table 2: Environment construction and leakage-resistant episode splits. Interquartile range (IQR) summarizes decision counts. [†]LoCoMo counts are per fold; [‡]SWE-V counts are fold averages, with test folds ranging from 42 to 161 instances.

sibility masks and paired controller executions unchanged. The SLA threshold equals the environment’s tool-time cap; both p99 latency and incident frequency are reported because an average can conceal rare verifier stalls.

4.3 Logged training and baselines

Training trajectories come from a frozen mixture of Always-Retrieve, Entropy Trigger, and Conf-Router policies with 10% uniform exploration over feasible actions. The exact behavior propensity is logged before action execution. The action-value network has three 512-unit GELU layers with dropout 0.1; optimization uses AdamW at 2×10^{-4} , batch size 2,048, five source-group folds, and five ensemble members. Propensity and outcome nuisance models use the same state but separate parameters. All clipping, effective-sample-size, reserve, and cost-weight thresholds are selected on development episodes and then frozen.

We compare with several levels of controller complexity. Always-Retrieve- k spends whenever the requested action is feasible. Entropy Trigger uses a development-tuned uncertainty threshold. Conf-Router, inspired by model self-knowledge (Kadavath et al., 2022), retrieves below a confidence threshold and verifies within a tuned marginal band. MLP Controller learns action scores from the generic logged feature vector but has neither the structured sketch nor an adaptive reserve. BudgetRate is the strongest rule-based baseline: it jointly tunes entropy threshold, early spending cap, phase boundary, and a retrieval interval that grows when consumption outpaces progress; a fixed verification-on-failure rule fires when latency reserve remains. Its prespecified grid contains 144 configurations and receives the same development episodes and feasibility masks as ReserveGate. Long Context inserts all available evidence that fits the model window, providing a competitive stress test of whether indiscriminate context can replace scheduled evidence use.

4.4 Evaluation protocol

Controllers run on identical episode realizations, candidate shards, and environment seeds. Success is averaged over the five paired rollouts within each episode-seed and then over seeds. Bootstrap and permutation inference resample the source grouping unit—conversation, API family, or repository—while keeping every derived episode, seed, and rollout together; the evaluated test predictions cover 10, 40, and 12 held-out source groups, respectively. The primary endpoint is terminal success; secondary endpoints are evidence-action and end-to-end tokens, p95 and p99 tool time, SLA incidents, and action frequency. Uncertainty intervals use 10,000 paired source-group bootstrap resamples, and significance uses 100,000 paired source-group permutations. We apply Holm correction to the three primary comparisons against the strongest matched baseline. Horizon quartiles and the failure taxonomy are specified before inspecting method identities. For the responsiveness exercise, transcripts are shuffled, and three automated profiles receive a fixed definition of necessary evidence use and visible interruption. Appendix A provides additional split, controller, accounting, and statistical details.

5 Results

5.1 Matched and full-accounting comparison

Table 3 gives the matched controller comparison and the LoCoMo system-boundary view. ReserveGate improves success over Conf-Router by +2.6, +3.8, and +2.9 points on LoCoMo-Episodic, StableToolBench-PO, and SWE-bench-Verified-420, respectively. Paired tests give $p = 0.008$, 0.003, and 0.011 after Holm correction. Against BudgetRate—a 144-configuration combination of entropy, reserve, phase boundary, consumption-responsive frequency, and verification gating—the gains are +3.1, +4.2, and +3.1 points. For the primary Conf-Router comparison, paired 95% confidence intervals are [1.1, 4.1], [1.7, 5.9], and [0.9,

4.8] points; each excludes zero because controller outcomes are paired within the same source groups.

Under full accounting, ReserveGate uses 5.5% fewer end-to-end tokens than Conf-Router while increasing success. Its default p99 is 5.9% lower, and ReserveGate-SLA further reduces p99 to 5.8 seconds and incidents to 1.4%. The constrained variant exchanges 0.7–0.8 success points for lower tail risk, making the quality–reliability operating choice explicit. Across ToolBench-PO and SWE-V, end-to-end tokens fall by 6.4% and 6.6%, p99 falls by 9.2% and 7.8%, and incident rates fall by 2.5 and 1.8 points relative to Conf-Router. Grouped tail bootstraps place the absolute p99 reductions at [-0.7, -0.1], [-1.1, -0.3], and [-0.7, -0.1] seconds. Incident reductions are 44.9%, 44.6%, and 42.9% in relative terms, with absolute-reduction intervals of [1.4, 3.0], [1.6, 3.4], and [0.9, 2.7] points.

The full-accounting view preserves the matched success ordering across all three environments (Appendix Table 7). Matched and full-accounting tables share success values because they score the same paired executions: the accounting boundary changes which tokens and latency are measured, while the evidence-action caps and realized task outcomes remain fixed. ReserveGate-SLA trades 0.7–0.8 success points for a further 0.6–0.9 seconds at p99 and reaches 1.2–1.7% incident rates.

Scarcity frontier. The advantage attenuates as the cap relaxes (Table 4), identifying resource scarcity as the operating condition in which scheduling changes outcomes. At medium budget, BudgetRate, Conf-Router, and ReserveGate obtain 63.8/64.2/65.1 on LoCoMo, 48.2/48.8/50.0 on ToolBench-PO, and 26.2/26.5/27.4 on SWE-V. The resulting 0.9–1.2-point margin complements the 2.6–3.8-point tight-cap gain and isolates evidence timing as the changing factor under a fixed task policy.

5.2 Compounding-horizon gains

The estimated margin increases with episode length (Table 5). On the longest quartile, ReserveGate gains 4.6 points on LoCoMo, 6.2 points on ToolBench-PO, and 5.2 points on SWE-V over Conf-Router. Grouped 95% confidence intervals are [2.8, 6.4], [3.5, 8.7], and [2.4, 7.9]; paired wins/ties/losses across Q4 episode-seeds are 764/1,610/626, 247/568/185, and 112/328/85. This pattern supports the proposed mechanism: when more decisions share one cap, early spending errors

have more opportunities to compound.

5.3 Mechanism tests and stronger alternatives

BudgetRate trails ReserveGate by 3.1–4.2 points; replacing the sketch with a leave-one-environment-out (LOEO) generic encoder costs 2.0–3.0 points, and fixed pricing/reserve costs 1.0–1.3 points (Appendix Table 16). Thus structured uncertainty determines what to acquire, while consumption adaptation determines when it remains affordable. Holding all learned components fixed, adaptation supplies most of the schedule gain, reserve release adds 0.2–0.3 points, and a learned schedule differs by at most 0.1 points (Appendix Table 17). A budget-matched two-step model-predictive controller (MPC-2) changes success by +0.3, +0.4, and -0.2 points relative to ReserveGate, while requiring 3.82–5.06 ms per decision. The one-step rule therefore remains on the practical quality–controller-cost frontier (Appendix I).

Cross-stack and intervention checks. On tight-budget LoCoMo, Llama/Qwen and BGE/E5 pairings preserve ReserveGate’s 2.6–3.0-point gain over the strongest same-stack controller while reducing tokens by 5.1–8.9%. For cloned m versus NOOP interventions, propensity expected calibration error (ECE) is below 0.06, outcome area under the receiver operating characteristic curve (AUROC) is at least 0.82, and RETRIEVE/VERIFY direction agreement is 84.1–88.0%/75.6–80.3%; Appendix G reports ESS-stratified calibration and Appendix J reports cross-stack intervals.

Portability. The four latent components transfer unchanged; audited proposals reduce 12/18/9-map construction by 66–68% with at most 0.4 success-point change, while generic encoders trail by 1.6–3.0 points (Appendix Table 11).

5.4 Tail control and support-aware fallback

On tight-budget LoCoMo, RETRIEVE consumes 58.4% of evidence-action tokens, while VERIFY is 8.4% of decisions but 61.8% of tool time; ReserveGate-SLA uses this asymmetry to cut p99.9 from 9.6 to 7.4 seconds for 0.8 success points. Across 120 randomized Q4 transcripts, three blinded automated profiles prefer ReserveGate/Conf-Router/tie in 77/28/15 cases, while longest stalls fall from 4.1 to 2.9 seconds. The failure audit shows that support fallback and SLA control directly contain the two estimator

Method	Success (%)			LoCoMo full accounting		
	LoCoMo	ToolBench-PO	SWE-V	E2E tok. (k)	p99 (s)	SLA inc. (%)
Always-Retrieve-32	54.7	36.8	16.4	28.6	8.8	9.1
Entropy Trigger	55.3	37.1	17.4	25.4	7.5	6.3
BudgetRate	58.1	40.9	19.9	24.8	6.9	4.7
MLP Controller	57.9	40.1	19.6	24.9	7.3	5.2
Conf-Router	58.6	41.3	20.1	25.3	6.8	4.9
Long Context [†]	54.8	36.3	16.9	31.2	2.1	1.6
ReserveGate	61.2	45.1	23.0	23.9	6.4	2.7
ReserveGate-SLA	60.4	44.3	22.3	24.1	5.8	1.4

Table 3: Tight-budget comparison over five seeds and five rollouts per episode-seed. E2E denotes end-to-end, and SWE-V denotes SWE-bench-Verified-420. [†]Long Context’s low tool-time tail reflects fewer external calls alongside lower success.

Evidence cap	LoCoMo	ToolBench-PO	SWE-V
8k/4s	57.2/53.4	39.1/34.6	19.9/16.2
12k/6s	61.2/58.6	45.1/41.3	23.0/20.1
16k/8s	65.1/64.2	50.0/48.8	27.4/26.5
24k/12s	68.8/68.1	55.2/54.4	31.5/30.8

Table 4: ReserveGate/Conf-Router success (%) across paired evidence-action caps. The full six-controller frontier is in Appendix Table 8.

Horizon quartile	LoCoMo	ToolBench-PO	SWE-V
Q1 (shortest)	+0.5	+1.2	+0.8
Q2	+1.8	+2.8	+2.0
Q3	+3.5	+5.0	+3.6
Q4 (longest)	+4.6	+6.2	+5.2

Table 5: Success gain (percentage points) over Conf-Router by equal-sized episode-length quartile.

boundaries; Appendices L and M report uncertainty, raw counts, and both denominators.

6 Related Work

RAG supplies external evidence (Lewis et al., 2020; Karpukhin et al., 2020), while adaptive retrieval estimates evidence need for a current query, generation span, or model state (Asai et al., 2024; Jiang et al., 2023; Jeong et al., 2024; Su et al., 2024). X-Router jointly chooses retrieval and reasoning modes under a user-specified cost trade-off (Wang et al., 2026). These methods optimize a local pipeline choice; ReserveGate schedules recurrent evidence actions whose token and tool-time consumption changes the feasible choices later in the same episode. Its component-wise masks, consumption-adaptive resource price, progress-released reserve, and support-aware fallback jointly implement this shared-cap control layer.

Long-horizon memory systems organize or learn operations over growing histories (Chen et al., 2023; Hu et al., 2025; Yan et al., 2026). BudgetMem routes among cost tiers inside a runtime memory pipeline (Zhang et al., 2026), and strong long-context retrieval baselines test whether capacity can replace selective evidence access (Laitenberger et al., 2025). ReserveGate adds an episode-global resource state and late-capacity mechanism to these local access decisions. Within our evaluation, BudgetRate, Conf-Router, MLP Controller, learned scheduling, and MPC-2 isolate tuned rate limiting, confidence routing, generic state, schedule flexibility, and online lookahead, respectively.

7 Conclusion

ReserveGate makes episode-level resource state a first-class input to retrieval and verification. Its auditable one-step rule combines action value with separate token/time masks, consumption-adaptive pricing, and released reserve. Across three environ-

ments, this design improves tight-budget success by 2.6–3.8 points, reaches 4.6–6.2-point gains in the longest quartile, and raises success while reducing end-to-end tokens and tail incidents under full accounting. A four-stack LoCoMo replication extends the gain across generators and retrievers. Evidence quality and timing are separate optimization axes; episode-level scheduling matters most where sequential spending errors compound.

Limitations

The fixed-continuation one-step estimand captures longer-range credit through repeated improvement and reserve control; doubly robust identification relies on the observed state and action overlap, with ESS fallback containing weak support. The cross-stack replication covers tight-budget LoCoMo, while the three-environment study uses the primary stack. Calibrated tool-time quantiles and audited observation mappings may shift under new interfaces, and the lower-support SUMMARIZE/WRITE actions are evaluated separately. Responsiveness and failure analyses use fixed blinded automated profiles to quantify rubric-consistent interaction behavior.

Ethical Considerations

The evaluation uses public research benchmarks and derived agent environments; we collect no new personal data and recruit no human participants. Logged tool traces can contain private content if the same controller is deployed on live systems, so production use should apply redaction, access control, and retention limits. Budgeted retrieval and verification improve the success–resource frontier under the reported caps, but they do not certify factual correctness or authorize unsupervised action on user data.

Acknowledgments

We thank Coco Qu, Kaige Zheng, and Zhiyuan Chen for their assistance with running experiments and evaluations, and for valuable discussions on system design and evaluation methodology. We also thank the anonymous reviewers and area chairs for their constructive comments.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. [Self-RAG: Learning to](#)

[retrieve, generate, and critique through self-reflection](#). In *International Conference on Learning Representations*.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. 2023. [Walking down the memory maze: Beyond context limit through interactive reading](#). *arXiv preprint arXiv:2310.05029*.

Miroslav Dudík, Dumitru Erhan, John Langford, and Lihong Li. 2014. [Doubly robust policy evaluation and optimization](#). *Statistical Science*, 29(4):485–511.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. [StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11143–11156.

Ronald A. Howard. 1966. [Information value theory](#). *IEEE Transactions on Systems Science and Cybernetics*, 2(1):22–26.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 2025. [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*, pages 32779–32798.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. [Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 7036–7050.

Nan Jiang and Lihong Li. 2016. [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*, pages 652–661.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. [Active retrieval augmented generation](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. [SWE-bench: Can language models resolve real-world GitHub issues?](#) In *International Conference on Learning Representations*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and

- 17 others. 2022. [Language models \(mostly\) know what they know](#). *arXiv preprint arXiv:2207.05221*.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. [Dense passage retrieval for open-domain question answering](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781.
- Alex Laitenberger, Christopher D. Manning, and Nelson F. Liu. 2025. [Stronger baselines for retrieval-augmented generation with long-context language models](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32559–32569.
- 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. 2020. [Retrieval-augmented generation for knowledge-intensive NLP tasks](#). In *Advances in Neural Information Processing Systems*.
- Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. [Evaluating very long-term conversational memory of LLM agents](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 13851–13870.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. [ToolLLM: Facilitating large language models to master 16000+ real-world APIs](#). In *International Conference on Learning Representations*.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. [Toolformer: Language models can teach themselves to use tools](#). In *Advances in Neural Information Processing Systems*.
- Weihsang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. [DRAGIN: Dynamic retrieval augmented generation based on the real-time information needs of large language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 12991–13013.
- Zixuan Wang, Yinze Ding, Zihan Wang, Jinyu Guo, Zhenhong Zhou, Junhao Dong, and Chaomeng Chen. 2026. [X-Router: Decoupling knowledge and reasoning for cost-effective LLM inference](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 19856–19874.
- Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. 2026. [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*, pages 12805–12825.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. [ReAct: Synergizing reasoning and acting in language models](#). In *International Conference on Learning Representations*.
- Haozhen Zhang, Haodong Yue, Tao Feng, Quanyu Long, Jianzhu Bao, Bowen Jin, Weizhi Zhang, Xiao Li, Jiaxuan You, Chengwei Qin, and Wenya Wang. 2026. [Learning query-aware budget-tier routing for runtime agent memory](#). In *Proceedings of the 43rd International Conference on Machine Learning*.

A Implementation and Evaluation Details

This appendix describes the implementation and evaluation protocol. It specifies the benchmark-derived splits, grouping rules, controller configuration, accounting procedure, and statistical tests. Each run record contains the episode and step identifiers, pre-action state, chosen action, behavior propensity, predicted action values, estimated and realized costs, remaining budgets, tool outcome, terminal score, environment seed, model and retriever revisions, and software revision.

Environment	Train	Dev	Test	Grouping key
LoCoMo-Episodic [†]	1,440	480	480	Source conversation
StableToolBench-PO	640	80	80	API family
SWE-bench-Verified-420 [‡]	280	70	70	Repository

Table 6: Episode splits used by the benchmark-derived environments. [†]LoCoMo counts are per fold; [‡]SWE-V counts are fold averages. Their out-of-fold tests cover all 2,400 and 420 episodes.

In the protocol, all methods are evaluated on the same episode realization and environment seed, with five paired stochastic rollouts per episode-seed. StableToolBench-PO uses ten grouped folds over 40 API families, with 32/4/4 families and 640/80/80 episodes in each train/development/test split. The retained SWE-V repository counts are 160/70/42/32/32/22/22/19/10/8/2/1; pairing large and small repositories yields six test folds of 161/72/50/42/51/44 instances. Paired confidence intervals and permutation tests resample source conversations, API families, or repositories, not derived decisions or episodes. All episodes, seeds, and rollouts from one grouping unit remain in the same resampling cluster. The primary hypothesis

test uses a paired permutation test on source-group-clustered episode outcomes; the three environment-level tests are Holm-corrected. Tail metrics are computed from measured episode-level tool time, and provider or local tokenizer logs supply token accounting.

Controller configuration. The action-value network has three layers with 512 hidden units, GELU activations, and dropout 0.1. Training uses AdamW, learning rate 2×10^{-4} , batch size 2,048, five source-group cross-fitting folds, and five ensemble members. The default cost weight, maximum reserve, overspending coefficient, importance-weight clip, and ESS threshold are selected on development data from prespecified grids and then frozen. The categorical sketch tables use add-one smoothing, and the symmetric Beta(2, 2) reliability prior supplies one virtual success and one virtual failure before any observed tool outcome. Both choices are weak regularizers fixed before development selection.

Cost and horizon estimators. Pre-action token costs are exact counts from the frozen generator tokenizer, including inserted retrieval text and verifier prompts. Tool time uses a gradient-boosted conditional quantile model at $q = 0.95$ with action type, retrieval depth, interface, recent latency, retry state, and observable tool history as features. Models fit source-group training folds; split-conformal residuals from development groups calibrate the upper quantile, after which all parameters are frozen. Timeouts contribute their realized timeout duration, and retries contribute every attempt, preserving the empirical tail. The horizon predictor uses task-schema features and the first two observable steps to estimate an upper conditional quantile of remaining decisions. An underestimated horizon releases reserve earlier while the consumption penalty and hard masks remain active; an overestimate preserves more reserve; missing predictions invoke the frozen uncertainty-trigger fallback.

Compute environment. Generator inference uses 8 NVIDIA H100 80GB GPUs and vLLM 0.6.3; retriever inference uses the same hosts with FAISS 1.8.0. Controller training, nuisance fitting, forced interventions, and all paired evaluations consume approximately 1,184 GPU-hours.

Licenses and data provenance. The three benchmark-derived environments are built entirely

from public research sources: LoCoMo (Maharana et al., 2024), StableToolBench (Qin et al., 2024; Guo et al., 2024), and SWE-bench Verified (Jimenez et al., 2024). Each source is used under the standard public research license and terms of use released with the respective benchmark, and solely for the non-commercial research evaluation for which those benchmarks are intended. We collect no new data and recruit no participants; the environments are derived transformations of these existing artifacts, with the model and retriever stacks identified in Section 4.2. The derived split sizes, grouping keys, and filtering rules are documented in Table 6 and Section 4.1.

B One-Step Utility Guarantee

Fix a reached state s , the estimated feasible set $\hat{\mathcal{F}}(s)$, resource price λ , and normalized cost $\bar{C}$. Let $U(s, m) = A^{\pi_c}(s, m) - \lambda \bar{C}(s, m)$ and $\hat{U}(s, m) = \hat{A}(s, m) - \lambda \bar{C}(s, m)$. Assume (i) the continuation policy π_c is fixed across candidate interventions, (ii) all candidates are evaluated on the same state and cost scale, and (iii) $|\hat{A}(s, m) - A^{\pi_c}(s, m)| \leq \epsilon$ for every $m \in \hat{\mathcal{F}}(s)$. If $\hat{m} \in \arg \max_m \hat{U}(s, m)$ and $m^* \in \arg \max_m U(s, m)$ on $\hat{\mathcal{F}}(s)$, then

$$U(s, m^*) - U(s, \hat{m}) \leq \epsilon + \epsilon = 2\epsilon,$$

because the intervening estimated-utility difference is non-positive by optimality of $\hat{m}$, and each endpoint estimation error is at most ϵ . For tool time, split-conformal calibration of the conditional q -quantile gives marginal coverage q on exchangeable calibration/test groups; therefore an action admitted by the time mask has per-decision overrun probability at most $1 - q$ on that distribution. The guarantee concerns utility regret within the estimated feasible set, while the reported incidents measure residual cost-model failures directly.

C Full Accounting Results

Table 7 expands the accounting columns that are compressed in the main paper. End-to-end tokens include the system prompt, episode history, task-action generation, retrieval and reranking text, and verifier text. Tool latency is measured wall-clock and includes retries and timeouts. On tight-budget LoCoMo, RETRIEVE represents 32.7% of decisions and 58.4% of evidence-action tokens, whereas VERIFY represents 8.4% of decisions and 61.8% of tool time.

Environment	Method	Success (%)	E2E tok. (k)	p99 (s)	SLA inc. (%)
LoCoMo	BudgetRate	58.1	24.8	6.9	4.7
	Conf-Router	58.6	25.3	6.8	4.9
	ReserveGate	61.2	23.9	6.4	2.7
	ReserveGate-SLA	60.4	24.1	5.8	1.4
ToolBench-PO	BudgetRate	40.9	20.0	7.8	5.1
	Conf-Router	41.3	20.4	7.6	5.6
	ReserveGate	45.1	19.1	6.9	3.1
	ReserveGate-SLA	44.3	19.3	6.0	1.7
SWE-V	BudgetRate	19.9	17.8	5.3	4.0
	Conf-Router	20.1	18.2	5.1	4.2
	ReserveGate	23.0	17.0	4.7	2.4
	ReserveGate-SLA	22.3	17.2	4.0	1.2

Table 7: Per-environment tight-budget results under full accounting; E2E denotes end-to-end.

At the medium budget, ReserveGate preserves the ordering with a smaller margin, identifying resource scarcity as the condition that amplifies the value of selective scheduling (Table 9).

The complete sweep comprises 60,000, 20,000, and 10,500 rollout outcomes per method and cap, clustered into 10 conversations, 40 API families, and 12 repositories for inference. Table 8 reports all six controllers’ success; Table 10 exposes the request-level resource and tail behavior of the proposed controller at every cap.

LoCoMo cap	BudgetRate	Conf-R.	MLP	Long Ctx.	ReserveGate	ReserveGate-SLA
8k/4s	52.9	53.4	52.5	50.8	57.2	56.3
12k/6s	58.1	58.6	57.9	54.8	61.2	60.4
16k/8s	63.8	64.2	63.4	59.9	65.1	64.4
24k/12s	67.5	68.1	67.2	65.8	68.8	68.3
ToolBench cap	BudgetRate	Conf-R.	MLP	Long Ctx.	ReserveGate	ReserveGate-SLA
8k/4s	34.0	34.6	33.8	30.7	39.1	38.0
12k/6s	40.9	41.3	40.1	36.3	45.1	44.3
16k/8s	48.2	48.8	47.7	43.9	50.0	49.2
24k/12s	53.7	54.4	53.4	50.6	55.2	54.6
SWE-V cap	BudgetRate	Conf-R.	MLP	Long Ctx.	ReserveGate	ReserveGate-SLA
8k/4s	15.9	16.2	15.7	13.8	19.9	19.1
12k/6s	19.9	20.1	19.6	16.9	23.0	22.3
16k/8s	26.2	26.5	25.9	23.1	27.4	26.7
24k/12s	30.4	30.8	30.0	28.3	31.5	30.9

Table 8: Full-system success across paired evidence-action token/tool-time caps (percent). ReserveGate has its largest margin at the tightest cap and remains competitive as resources loosen; Long Context gains capacity without matching targeted scheduling.

Method	LoCoMo	ToolBench-PO	SWE-V
BudgetRate	63.8	48.2	26.2
Conf-Router	64.2	48.8	26.5
ReserveGate	65.1	50.0	27.4

Table 9: Success (%) under the medium budget.

Environment	Cap (tok./s)	Success (%)	E2E tok. (k)	p95 (s)	p99 (s)	SLA inc. (%)	Actions /100 steps	Ctrl. ms/decision
LoCoMo	8k/4	57.2	21.8	3.3	5.7	2.9	34.6	0.21
	12k/6	61.2	23.9	3.8	6.4	2.7	32.1	0.21
	16k/8	65.1	27.6	4.5	7.2	2.4	30.8	0.21
	24k/12	68.8	34.9	5.6	8.9	2.1	29.4	0.21
ToolBench-PO	8k/4	39.1	17.4	3.9	6.2	3.5	39.8	0.27
	12k/6	45.1	19.1	4.4	6.9	3.1	37.2	0.27
	16k/8	50.0	22.8	5.1	8.0	2.7	35.0	0.27
	24k/12	55.2	28.7	6.4	10.4	2.3	33.1	0.27
SWE-V	8k/4	19.9	15.2	2.6	4.0	2.7	25.8	0.29
	12k/6	23.0	17.0	3.0	4.7	2.4	24.0	0.29
	16k/8	27.4	19.7	3.5	5.6	2.1	22.7	0.29
	24k/12	31.5	24.6	4.3	7.1	1.8	21.5	0.29

Table 10: ReserveGate’s system-level outcomes as evidence-action caps vary. Incidents use the cap-specific SLA; action frequency counts non-NOOP evidence actions. Each cap aggregates five seeds and five paired rollouts over 2,400 LoCoMo, 800 ToolBench-PO, or 420 SWE-V out-of-fold test episodes.

D BudgetRate Baseline

BudgetRate is tuned with the same development budget as ReserveGate. Its grid covers entropy thresholds $\{0.45, 0.55, 0.65, 0.75\}$, early spending caps $\{0.5, 0.6, 0.7\}$, phase boundaries $\{0.5, 0.67, 0.75\}$, and retrieval intervals $\{1, 2, 4, 8\}$. At step t , the interval is multiplied by $1 + \max(0, q_t - u_t)$, where q_t is the consumed budget fraction and u_t is normalized progress. The selected policy caps spending before the final third and invokes verification after a tool failure when the latency reserve permits it. This baseline therefore combines a tuned uncertainty trigger, consumption-responsive rate limiter, and late-stage reserve.

E Portability and Mapping Construction

The latent sketch structure is shared across environments, while each interface maps observable signals into the same four likelihood updates. In the cold-start protocol, mapping proposals receive tool schemas, log-field definitions, and a fixed development-trace quota, but no test outcomes. An audit pass merges redundant accepted proposals, validates update direction on development traces, and freezes the map before test rollout.

The validated-proposal workflow reduces construction time by 66–68% while keeping success within 0.4 points. Proposal acceptance rates are 83%, 78%, and 86%; no mapping changes after test evaluation begins.

Environment	Mapping	Proposed/accepted	Dev traces	Minutes	Success (%)	Δ [95% CI] (points)	Fallback (%)
LoCoMo	Manual	12/12	240	96	61.2	0	3.9
	Validated proposals	18/15	240	31	61.0	-0.2 [-0.7, 0.3]	4.2
	Generic in-domain	–	240	–	59.5	-1.7 [-2.5, -0.9]	5.3
	Generic LOEO	–	240	–	59.0	-2.2 [-3.3, -1.1]	7.8
ToolBench-PO	Manual	18/18	160	118	45.1	0	5.2
	Validated proposals	23/18	160	39	44.7	-0.4 [-1.1, 0.2]	5.7
	Generic in-domain	–	160	–	42.8	-2.3 [-3.5, -1.1]	6.6
	Generic LOEO	–	160	–	42.1	-3.0 [-4.5, -1.5]	9.6
SWE-V	Manual	9/9	126	74	23.0	0	6.0
	Validated proposals	14/12	126	25	22.7	-0.3 [-0.9, 0.3]	6.4
	Generic in-domain	–	126	–	21.4	-1.6 [-2.7, -0.5]	7.4
	Generic LOEO	–	126	–	21.0	-2.0 [-3.3, -0.7]	10.4

Table 11: Cold-start portability matrix. Intervals resample conversations, API families, or repositories. Proposed/accepted counts precede audit merging into the final 12/18/9 maps.

Interface	Observable	sig-Component	Update/freeze point
LoCoMo	unsupported tality after retrieval	en-missing dence	evi-raise freeze after dev audit
ToolBench- PO	timeout schema rejection	or reliability	Beta failure; freeze after API-family dev
SWE-V	unresolved ing target test	fail-constraint	shift type mass; freeze after repository dev

Table 12: Executable mapping examples for the shared sketch.

F Optional Actions and Forced Interventions

The main controller focuses on retrieval and verification. We separately test SUMMARIZE and WRITE at states where both the action and NOOP are feasible. For each intervention, we clone the simulator or deterministic environment snapshot, execute the two arms, and roll both forward with shared exogenous randomness.

Action	LoCoMo feasible	ToolBench-PO n / sign correctness	SWE-V (%)
RETRIEVE	4,820 / 88.0	2,140 / 85.7	1,008 / 84.1
VERIFY	2,340 / 80.3	1,286 / 77.9	612 / 75.6
SUMMARIZE	2,120 / 71.8	880 / 68.9	402 / 66.7
WRITE	1,640 / 67.4	624 / 64.8	318 / 63.2

Table 13: Action-level validation with cloned counterfactual rollouts. Each cell reports feasible n / sign correctness.

The smaller optional-action sample sizes explain their wider confidence intervals. Removing SUMMARIZE changes success by -0.7 , -0.5 , and

-0.4 points; removing WRITE changes it by -0.3 , -0.2 , and -0.5 points. The primary gain is carried by RETRIEVE and VERIFY, while SUMMARIZE and WRITE provide smaller complementary improvements.

G Support-Stratified Proxy Validation

The aggregate action-level diagnostics are:

We pool the prespecified forced interventions across environments after computing effects within source groups, then stratify by action-level ESS at the reached state. Calibration slope and intercept regress realized paired intervention effects on predicted advantages; fallback is the fraction for which the deployed controller invokes its conservative trigger. Across environments, grouped 95% intervals for RETRIEVE sign agreement span [81.2, 90.6]% and its MAE spans [0.78, 1.69] points; the corresponding VERIFY ranges are [71.0, 84.1]% and [1.27, 2.53].

The support trend supplies a direct mechanism check: the learned ranking is most accurate where logged overlap is strongest, and the controller channels degrading support into conservative fallback.

Environment	RETRIEVE intervention			VERIFY intervention			Logged-data diagnostics		
	n	Sign (%)	MAE (pp)	n	Sign (%)	MAE (pp)	ECE	AUROC	Fallback (%)
LoCoMo	4,820	88.0	0.96	2,340	80.3	1.56	0.041	0.861	3.9
ToolBench-PO	2,140	85.7	1.21	1,286	77.9	1.88	0.055	0.833	5.2
SWE-V	1,008	84.1	1.44	612	75.6	2.14	0.058	0.821	6.0

Table 14: Forced-action agreement and logged-data support. Sign is agreement with realized intervention direction; mean absolute error (MAE) measures intervention-effect error in percentage points.

Action	ESS	n	Sign [95% CI]	MAE [95% CI]	Slope/int.	Fallback
RETRIEVE	< 50	612	66.8 [62.9,70.5]	2.43 [2.18,2.71]	0.68/0.021	92.6
RETRIEVE	50–149	2,516	82.7 [80.9,84.4]	1.37 [1.25,1.49]	0.89/0.009	8.4
RETRIEVE	≥ 150	4,840	89.2 [88.1,90.2]	0.94 [0.88,1.00]	0.97/0.003	1.2
VERIFY	< 50	384	62.7 [57.8,67.3]	2.88 [2.50,3.29]	0.59/0.028	95.1
VERIFY	50–149	1,442	74.8 [72.0,77.5]	2.09 [1.91,2.28]	0.81/0.014	11.7
VERIFY	≥ 150	2,412	82.1 [80.5,83.6]	1.52 [1.41,1.64]	0.92/0.006	2.3

Action	Quartile	Sign [95% CI]
RETRIEVE	Q1	81.5 [79.4,83.4]
RETRIEVE	Q4	90.3 [88.7,91.8]
VERIFY	Q1	71.9 [68.9,74.7]
VERIFY	Q4	82.8 [80.3,85.0]

Table 15: Support- and horizon-stratified forced-action calibration. Direction agreement strengthens with support and horizon, while weak-support states activate fallback instead of unchecked extrapolation. Sign and fallback are percentages; MAE is in percentage points.

H Adaptive Schedule and Sensitivity

The component ablations under the tight cap are:

We compare fixed and adaptive schedules using the same action-value model.

The deterministic adaptive rule matches the learned schedule while preserving a directly inspectable decision mechanism. Its grouped difference from the deterministic rule is $[-0.4, 0.3]$, $[-0.5, 0.4]$, and $[-0.3, 0.5]$ points on LoCoMo, ToolBench-PO, and SWE-V. Across $\lambda_0 \in \{0.12, 0.18, 0.24\}$ and $r_{\max} \in \{0.20, 0.30, 0.40\}$, LoCoMo success remains within 0.6 points of the best development configuration, indicating a broad operating region around the selected setting.

Variant	LoCoMo	ToolBench-PO	SWE-V
Full ReserveGate	61.2	45.1	23.0
Fixed λ and reserve	60.1	43.8	22.0
LOEO generic log encoder	59.0	42.1	21.0
No uncertainty sketch	58.5	41.6	20.8
No reserve	59.2	42.6	21.2
No VERIFY	59.0	42.0	21.1

Table 16: Component-ablation success (%) under the tight budget.

Schedule	LoCoMo	ToolBench-PO	SWE-V
Fixed λ , fixed reserve	60.1	43.8	22.0
Phase-linear λ_t	60.7	44.5	22.5
Consumption-adaptive λ_t	61.0	44.8	22.8
Adaptive λ_t + released reserve	61.2	45.1	23.0
Learned schedule	61.1	45.0	23.1

Table 17: Success (%) for alternative budget schedules.

I Budget-Matched Two-Step Lookahead

MPC-2 uses the same state, action set, cost estimates, learned continuation model, feasibility masks, development budget, and test episodes as ReserveGate. For each feasible first action it expands one additional evidence decision, scores the two-step branch with the learned continuation

model, and then executes the first action. It receives no additional retrieval candidates or test-time tuning.

MPC-2's paired success differences from ReserveGate are +0.3 points (95% CI [-0.4, 1.0]), +0.4 ([-0.5, 1.3]), and -0.2 ([-1.1, 0.8]). Its 16.6–18.2 $\times$ controller-cost increase yields no resolved success advantage, showing that the one-step rule captures the practically attainable scheduling gain under these caps.

Environment	Controller	Success (%)	E2E tok. (k)	p99 (s)	Inc. (%)	Ctrl. ms/decision
LoCoMo	BudgetRate	58.1	24.8	6.9	4.7	0.07
	Conf-Router	58.6	25.3	6.8	4.9	0.18
	Learned schedule	61.1	24.0	6.5	2.8	0.46
	ReserveGate	61.2	23.9	6.4	2.7	0.21
	MPC-2	61.5	24.0	6.3	2.8	3.82
ToolBench-PO	BudgetRate	40.9	20.0	7.8	5.1	0.08
	Conf-Router	41.3	20.4	7.6	5.6	0.22
	Learned schedule	45.0	19.2	7.0	3.2	0.55
	ReserveGate	45.1	19.1	6.9	3.1	0.27
	MPC-2	45.5	19.2	6.9	3.0	4.47
SWE-V	BudgetRate	19.9	17.8	5.3	4.0	0.09
	Conf-Router	20.1	18.2	5.1	4.2	0.25
	Learned schedule	23.1	16.9	4.8	2.5	0.61
	ReserveGate	23.0	17.0	4.7	2.4	0.29
	MPC-2	22.8	17.1	4.8	2.5	5.06

Table 18: Tight-cap comparison with a budget-matched two-step model-predictive controller. All methods use identical candidate actions and paired episodes.

J Task-Agnostic Transfer

The matched tight-budget LoCoMo cross-stack replication refits nuisance, cost, and action-value models while preserving episode IDs, action interface, grids, and caps. Best-baseline/ReserveGate success and ReserveGate’s token change are 58.6/61.2/−5.5% for Llama-BGE, 57.8/60.6/−5.1% for Llama-E5, 56.4/59.4/−8.9% for Qwen-BGE, and 55.9/58.8/−8.2% for Qwen-E5. The grouped success-gain intervals are [1.1, 4.1], [1.2, 4.4], [0.9, 5.0], and [0.8, 5.1].

The task-agnostic encoder receives normalized budgets, phase, tool outcomes, retrieval coverage, and model confidence but no named latent variables or benchmark-specific likelihood maps. When trained within each environment, it reaches 59.5%, 42.8%, and 21.4%; in the harder leave-one-environment-out setting, it reaches 59.0%, 42.1%, and 21.0% on LoCoMo, ToolBench-PO, and SWE-V. Full ReserveGate reaches 61.2%, 45.1%, and 23.0%. The gap quantifies the value of the task-specific observation interface: generic logs recover most of the structured sketch benefit, and audited mappings supply the remaining 1.6–3.0 points.

K Operational Robustness Matrix

The robustness study targets operational stressors: resource exhaustion, heavy-tailed latency, horizon error, weak overlap, mapping error, and stack shift. The matrix identifies the protected object, enforcement boundary, strongest comparator, and empirical trace for each stressor.

Protected object	Operational stressor	Trust assumption	Enforced property	Comparator	Adaptive/held-out setting	Utility/overhead	Observed boundary	Evidence trace
Episode tokens	early high- k retrieval	tokenizer and action log	exact pre-action token mask	BudgetRate	held-out groups, caps	source four 0.21–0.29 ms	success frontier; residual loading	front- Tabs. 4, 10
Tool-time cap	heavy verifier tail, retry, timeout	calibrated tency log	la- upper-quantile time mask	ReserveGate-SLA	cap-specific bootstrap	tail 0.7–0.8 point SLA trade-off	cost underestimation	Tab. 7; Sec. M
Late capacity	horizon underestimation	causal progress features	adaptive price plus hard reserve	fixed schedule	Q1–Q4 schedule	and abla- lease gain	0.2–0.3 point re-der short horizon	early release un- Tabs. 17, 5
Action ranking	weak logged overlap	propensity group split	and ESS-triggered fallback	Conf-Router	forced actions by ESS bin	fallback rises before extrapolation	lower action support	optional- Tabs. 15, 13
Shared state	observation-map error	schema/log fields are observable	development audit and freeze	generic coder	en- cold-start quota, LOEO transfer	66–68% lower construction time	residual mapping error	Tabs. 11, 12
Scheduling effect	generator–retriever shift	same-stack parity	par- refit nuisance and value models	strongest same-stack controller	four generator–retriever pairs	2.6–3.0 point gains	stack-specific absolute success	Sec. 5.3

Table 19: Systems evidence matrix connecting each operational stressor to assumptions, enforcement, comparison, overhead, failure boundary, and paper-internal evidence.

L Responsiveness Exercise

We sample 12 Q4 out-of-fold episodes from each of the 10 LoCoMo source conversations and use one prespecified environment seed per episode, yielding 120 distinct episode-seed transcript pairs comparing ReserveGate and Conf-Router. Three automated annotator profiles score which interaction is more responsive while preserving necessary evidence use. No human annotators were recruited or involved; the profiles are automated LLM personas applying a fixed rubric, and a deterministic tie-break resolves programmed disagreements. ReserveGate is preferred in 64.2% of comparisons, Conf-Router in 23.3%, with 12.5% ties. These are majority-aggregated episode outcomes: 77 ReserveGate wins, 28 Conf-Router wins, and 15 ties. The mean longest tool stall falls from 4.1 to 2.9 seconds, and visible retrieval or verification interruptions fall from 3.8 to 2.5 per episode. Agreement measured by Fleiss’ κ is 0.61. Conversation-clustered 95% intervals are [55.4, 72.5]% for ReserveGate preference, [16.2, 31.8]% for Conf-Router preference, and [7.5, 19.4]% for ties. Paired changes are -1.2 seconds (95% CI [-1.5, -0.9]) for the longest stall and -1.3 interruptions per episode ([-1.6, -1.0]).

Episodes are stratified by paired task outcome before sampling, and transcript order is randomized independently for each profile. The rubric defines responsiveness as minimizing avoidable waiting and redundant evidence actions without omitting evidence needed to complete the task; profiles may select a tie. The profiles receive no method name, cost summary, or terminal score.

Category	Count	All (%)	Loss (%)
Verification overuse	783	5.4	37.4
Front-loading	537	3.7	25.6
Sketch error	363	2.5	17.3
Weak support	232	1.6	11.1
Cost-tail error	160	1.1	7.6
Other/ambiguous	139	1.0	6.6

Table 20: Failure taxonomy with raw counts and both denominators. Categories co-occur: 118 cases receive multiple primary labels, so percentages need not sum to 100.

M Failure Taxonomy

An automated two-profile audit labels 2,096 loss cases among 14,500 pooled test episode-seeds where ReserveGate loses to either Conf-Router or BudgetRate and resolves programmed disagreements; the two auditors are automated LLM profiles, and no human annotators were involved. Table 20 reports both pooled and loss-conditional denominators.

Observable signatures are repeated verification checks, early high- k retrievals, incorrect observation mappings, conservative fallback, and underestimated tool tails. The error classes connect directly to the controller safeguards: weak support activates fallback, cost tails motivate ReserveGate-SLA, and mapping error is measured by the cold-start audit. Conf-Router is most competitive on short episodes with well-calibrated local confidence, while the generic MLP can prevail in sparse logged states where ReserveGate invokes its conservative fallback.