AGENTS NEED EVENT TIME: FRESHNESS-AWARE REASONING OVER DELAYED TOOL RESULTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using agents often treat the latest message as the latest state. Under asynchronous execution, this ordering is wrong: a result delivered last can describe a snapshot taken before the world changed. We introduce *event-time reasoning*, a runtime protocol that attaches snapshot versions and scope watermarks to tool results so agents can validate evidence against decision-time state. EVENT-TIMEBENCH makes this failure measurable with evolving SQLite, workflow-event, and Python-AST backends, while WATERMARK combines exact version invalidation with calibration-free query influence and optional service cost. Across 97,200 controlled tasks and 747,600 paired outcomes, a preregistered five-seed confirmation raises exact accuracy from 87.0% with version-only ranking to 90.1% with WATERMARK-I, a +3.09-point gain $[2.53, 3.62]$ that also exceeds a direct answer-flip predictor; the gain remains +4.15 points $[3.68, 4.61]$ under two unseen transition regimes. Natural code exposes a complementary requirement: when sources jointly determine an answer, selective refresh can create an incoherent state, so the cut must remain atomic. Frozen query-family allocation reaches 96.45% across 265 commits from ten unseen repositories; a frozen gate matches that accuracy at 96.50% while refreshing 48.7% fewer sources than full refresh. Executable OpenCode sessions across two model families demonstrate integration. Event time makes freshness an explicit, efficient reasoning primitive for agents operating over evolving state.

1 INTRODUCTION

Consider an agent that submits an analytics query against database version 41. New records advance the table to version 44 while the query is running. The version-41 result then arrives as the latest message in the transcript. The same pattern appears when an API inspection starts before a repository revision, or when a workflow read returns after the object has advanced. Asynchrony overlaps computation; event time answers the semantic question that arrival order leaves open: *which world does this result describe?*

Recent agent systems make tool use faster through dependency graphs (Kim et al., 2024; Qiao et al., 2025), speculative execution (Ye et al., 2026; Guan et al., 2026), futures (Feng et al., 2026), and explicit coordination under delayed feedback (Shi et al., 2026). These advances determine when computation may proceed, but not whether a resolved value still describes the world in which the agent must act. This exposes a distinct inference gap: whenever the world evolves within the lifetime of a call, conversational recency can reverse informational recency—the last result to arrive can be the first snapshot taken.

Data systems address a related ambiguity by distinguishing event time from processing time (Akidau et al., 2013; 2015). Database snapshots and build artifacts similarly preserve a version or input identity (Bernstein & Goodman, 1983; Mokhov et al., 2018). We establish event time as the missing validity layer in the agent inference loop. A tool result exposes the snapshot at which it observed state, not merely the time at which its message was delivered. At a decision boundary, the agent compares that version with a current scope watermark, identifies invalidated evidence, and refreshes the observations most capable of changing its answer.

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.

EVENTTIMEBENCH makes this distinction measurable. Every query is fixed before future evolution, every result is bound to an executable snapshot, and exact labels are hidden from the controller. Three different backends—SQLite revision queries, workflow-event replay, and Python AST inspection—provide controlled breadth. Counterfactual latency schedules preserve queries, values, snapshots, versions, budgets, and final state while changing only delivery order. The design therefore separates temporal semantics from task difficulty.

Our controller, WATERMARK, is deliberately compact. Version comparison identifies which task-relevant scopes advanced. In controlled executable environments, a query-specific influence term prioritizes observations near a decision boundary or argmax contender. Calibration-free WATERMARK-I matches calibrated quality without fit data; WATERMARK-E combines influence with calibrated change probability and refresh cost, while WATERMARK-IL omits the transition calibration and ranks influence per unit latency. All modes refresh the highest-value cut within budget, never observe a test answer or current value before selection, require no LLM training, and run efficiently on CPUs.

The evidence establishes two operating regimes. In controlled executable environments, calibration-free WATERMARK-I raises accuracy from 87.0% to 90.1% over matched version-only ranking and retains a +4.15-point gain under unseen transition regimes; cost-aware WATERMARK-IL adds a quality–service control. In natural code, partial cuts can mix incompatible snapshots, and frozen atomic policies instead reach 96.5% across 265 commits from ten unseen repositories. Executable model sessions then test the same raw values, versions, watermarks, and budgets across conditions. This hierarchy connects one validity principle to two mechanisms: influence-ranked partial refresh when sources are separable, and coherent atomic refresh when they are jointly constrained.

Our contributions are:

- **Problem and benchmark.** We formulate the event-time gap in tool-using agents, define counterfactual schedule equivalence as a causal test of delivery-order reliance, and release EVENTTIMEBENCH with three executable evolving backends, 97,200 controlled tasks, exact validators, paired uncertainty, costs, shifts, and graph noise.
- **Mechanism and controlled evidence.** We introduce WATERMARK, a test-answer-free controller whose calibration-free influence mode gains +3.09 points over matched version-only ranking and +0.73 over a direct answer-flip predictor, then gains +4.15 points across unseen transition regimes. Its efficient mode exposes a quality–service control without a fitted transition model.
- **Coherence and executable integration.** Natural code reveals that partial refresh can form an impossible mixed snapshot. Frozen query-family allocation reaches 96.45% across 265 commits from ten unseen repositories; a frozen gate matches it at 96.50% while refreshing 48.7% fewer sources than full refresh. Two tool-using model families demonstrate integration without model fine-tuning.

2 PROBLEM SETUP

Let environment state at logical time t be x_t. Tool call i is issued at event time e_i, observes scope S_i at version v_i, and returns value y_i at arrival time a_i. We represent its temporal envelope as

$$o_i = (y_i, e_i, a_i, v_i, S_i). \tag{1}$$

The agent commits to an answer at decision time $d \geq a_i$. Conventional transcripts encode a_i through message order while often dropping e_i, v_i, and S_i.

Let $W_d(S)$ denote the current decision-time watermark for scope S. Observation o_i is *version valid* when $v_i = W_d(S_i)$. Version invalidity does not mean an answer must change: a source can advance without changing a query-relevant feature, or the observation can be far from the query boundary. The runtime must therefore solve two different problems:

1. *state validity*: which observed scopes have advanced; and
2. *decision influence*: which invalid observations can change the answer.

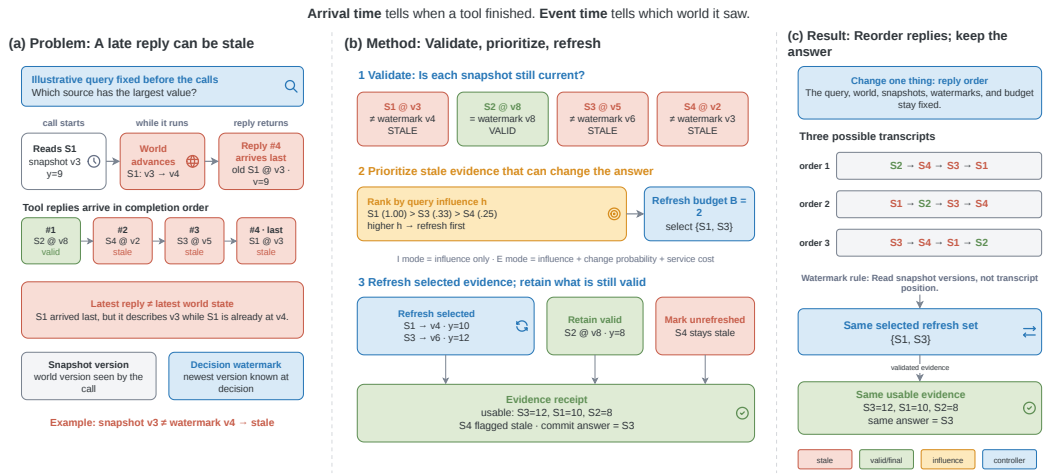

Figure 1: WATERMARK in one illustrative four-scope argmax episode. (a) A tool result can return late while still carrying an old snapshot, so transcript position does not imply decision-time validity. (b) The controller compares each snapshot with the decision watermark, retains valid evidence, prioritizes stale evidence by query influence under a refresh budget, and records usable and flagged evidence before committing. Values and scores are schematic. (c) When only reply order changes, the controller preserves the same refresh set, usable evidence, and answer. The operation is explicit: validate the snapshot first, then spend refresh budget where stale evidence can change the decision.

Version metadata answers the first exactly. WATERMARK estimates the second from calibration data and the structured query.

Counterfactual schedule equivalence. Two executions are schedule-equivalent when they share the query, state trajectory, issued calls, snapshot values and versions, decision-time watermarks, refresh budget, and model configuration, but assign different arrival times to independent calls. A policy is schedule-consistent if it commits the same output under every equivalent schedule. This is a targeted causal intervention: the world and available evidence are fixed; only delivery order changes.

Why clocks alone are insufficient. Arrival age $d - a_i$ measures how recently a message was delivered. Event age $d - e_i$ measures how long ago its snapshot was taken, but cannot reveal whether its scope actually changed. A version vector identifies concrete changes, yet under a tight budget still needs a task-aware rule for choosing among invalidated observations.

3 WATERMARK: VERSIONED, INFLUENTIAL EVIDENCE

WATERMARK sits between an asynchronous tool runtime and an otherwise unchanged agent. Its inputs are the envelopes in Eq. 1, the pending structured query, a decision-time scope watermark, and a refresh-call budget B.

3.1 VERSION INVALIDATION

Let $\mathcal{R}(q)$ be the source scopes relevant to query q. The controller first constructs

$$\mathcal{I}_q = \{i : S_i \in \mathcal{R}(q) \ \wedge \ v_i \neq W_d(S_i)\}. \tag{2}$$

Unchanged scopes are retained without a tool call. We charge one metadata lookup and its latency to every version-aware policy. The benchmark supplies exact relevance in its main condition and perturbs $\mathcal{R}(q)$ with both missing and spurious edges in Section 5.2.

3.2 Optional calibration for efficiency

For the efficiency mode, disjoint episodes estimate $\hat{p}_i$, the probability that source i's query value changes after an observation. The empirical estimator conditions on backend, source volatility class, and a two-step event-age bin, with smoothed backoff. It is fitted independently for every confirmatory seed. No terminal value, answer label, or test episode enters these estimates. We also evaluate $\hat{m}_i = \mathbb{E}[|\Delta_i|]$ as a magnitude factor. The quality mode below needs neither statistic; calibration is reserved for cost-sensitive variants and analysis.

3.3 Query-influence cut

For linear threshold and band queries with coefficient w_i, influence grows with $|w_i|$. For an argmax query, an observation is influential when its value is close to the observed leader. We use

$$h_i(q,o) = \begin{cases} |w_i|, & q \text{ is threshold or band,} \\ \left(1 + |y_i - \max_{j \in \mathcal{R}(q)} y_j|\right)^{-1}, & q \text{ is argmax,} \end{cases} \tag{3}$$

Accuracy-oriented WATERMARK-I assigns each invalid observation the calibration-free score

$$s_i^{\mathrm{I}} = h_i(q,o). \tag{4}$$

The calibrated quality comparator WATERMARK-Q uses $s_i^{\mathrm{Q}} = \hat{p}_i h_i(q,o)$. The earlier multi-budget grid and interactive sessions use WATERMARK-M, $s_i^{\mathrm{M}} = \hat{p}_i \hat{m}_i h_i(q,o)$, where $\hat{m}_i$ is the fit-partition mean absolute change for the source class. Efficiency-oriented WATERMARK-E additionally uses the recorded refresh service ℓ_i:

$$s_i^{\mathrm{E}} = \frac{\hat{p}_i \, h_i(q,o)}{\ell_i^{\alpha}}. \tag{5}$$

The transition-calibration-free cost mode WATERMARK-IL uses $s_i^{\mathrm{IL}} = h_i(q,o)/\ell_i^{\alpha}$. For each confirmatory seed, α is chosen from $\{0, .25, .5, .75, 1, 1.5, 2\}$ using a disjoint tuning partition: among candidates whose version-aware serial service does not exceed hazard-only, we select the highest-accuracy candidate, breaking near-ties toward lower service. Each mode refreshes the top $\min(B, |\mathcal{I}_q|)$ scores. Version comparison supplies instance-level invalidation, h_i supplies answer-level relevance, and optional calibration in Eq. 5 exposes a deployable cost control; WATERMARK-IL tests whether its transition model is necessary. No mode uses current test values or evaluates candidate cuts against test answers.

Schedule property. Equations 2–5 contain event time, snapshot state, current watermarks, and the query, but not arrival time. The selected cut is consequently identical for schedule-equivalent inputs. End-to-end consistency is still empirical because the downstream model must read differently ordered results, execute the refreshes, and compute the answer.

Coherent-cut extension. Selective refresh can combine old and current values into a state that never existed. For natural code we therefore add a confidence-gated atomic mode: high-risk tasks refresh all query sources, while low-risk tasks retain the complete parent cut. The gate uses only parent-snapshot structure and is calibrated before evaluation; it never mixes source versions or reads current values before acting.

4 EventTimeBench

4.1 Three executable evolving backends

Each task contains 8–13 versioned sources, of which 4–8 participate in a fixed threshold, band, or argmax query. Calls are issued at ticks 0–7, incur 1–5 ticks of transport latency, and resolve by decision tick 12. Source processes are stable, active, or volatile, with per-tick transition probabilities 0.025, 0.090, and 0.220. Queries, weights, relevant scopes, and thresholds use a dedicated random stream and depend only on tick-0 state; future state and terminal labels are never consulted. Transport uses a third independent stream.

The backends share the temporal protocol while implementing different state semantics:

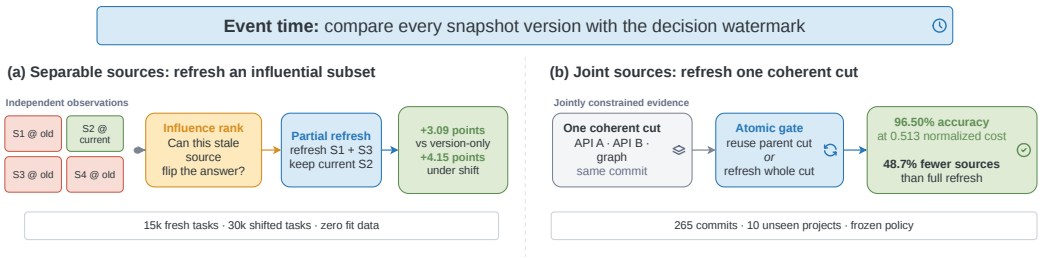

Figure 2: One event-time validity primitive supports two operating regimes. When sources are independently refreshable, WATERMARK-I spends a tight budget on stale evidence with the greatest query influence. When sources jointly determine an answer, the controller instead preserves a coherent cut and chooses between complete reuse and an atomic refresh. The result cards report fresh held-out evidence, not schematic values.

- **SQLite revisions.** Queries read versioned metric rows from an in-memory relational history; later transactions mutate signed values.
- **Workflow event replay.** Reads reconstruct object state from `advance`, `enqueue`, `resolve`, and `pause` events.
- **Python API revisions.** Reads parse revisioned Python functions with the AST and derive a score from parameters, async status, and return annotation.

Every observation is executed against its issue snapshot. Every refresh executes at decision time. Exact programmatic validators score the answer; no LLM judge is used.

4.2 SPLITS, BUDGETS, AND COSTS

The three-seed grid pairs 7,200 disjoint test tasks across 18%, 30%, and 50% budgets (237,600 rows). The separate selector confirmation uses five unseen seeds with 900 fit, 400 tune, and 1,000 test episodes per backend: 15,000 tasks and 90,000 rows. A later deterministic I replay adds 15,000 outcomes, explaining the 105,000-row study-map total. Fit estimates $\hat{p}$, tune chooses α, and test is evaluated once. A third five-seed draw uses 15,000 tasks/45,000 rows to compare WATERMARK-Q, Direct-Flip, and version-only.

The shift study freezes base-regime calibrators for 30,000 new tasks under accelerated or compressed transition rates, with a disjoint in-regime refit reference. A locked deterministic reanalysis then reproduces 45,000 saved Q rows before adding I on the 30,000 existing confirmation/audit tasks. WATERMARK-I becomes the method of record only before a wholly new 15,000-task, 75,000-row five-seed run; task-, seed-cluster-, and seed-level intervals are prespecified. Random ordering is another exact-reproduction reanalysis.

Finally, a locked replay of the calibration-audit tasks reproduces all 15,000 saved WATERMARK-E rows and adds $\hat{p}/\ell^{\alpha}$ and h/ℓ^{α} under the same split, alpha grid, metadata charge, budget, and service ceiling. Its 30,000 new outcomes bring the controlled total to 747,600 rows without adding an independent task.

Each version-aware decision pays metadata latency $0.18 + 0.015\,n$ service units for n sources. Refresh latency is heterogeneous and recorded per scope. We report refresh calls, serial service (the sum of metadata and refresh latency), and parallel critical path (metadata plus the longest refresh). **Full refresh** re-reads every invalidated relevant scope. **Synchronous** re-reads every relevant scope at decision time. Both reach the current answer and provide explicit cost endpoints.

4.3 BASELINES

Reuse makes no refresh. **Arrival-age** and **event-age** prioritize old observations by their respective clocks. **Hazard-only** ranks every relevant scope by calibration-only change probability, leaving instance versions unobserved. **Version-only** sees the same invalidation vector as WATERMARK and ranks invalidated scopes by change probability and event age, isolating the influence factor. Matched

factor variants rank by $\hat{p}\hat{m}$, $\hat{p}h$, or $\hat{p}\hat{m}h$; WATERMARK-E adds the calibrated latency denominator in Eq. 5. Matched version-aware efficiency controls use $\hat{p}/\ell^{\alpha}$ or h/ℓ^{α} (WATERMARK-IL) under the same alpha grid and tuning service ceiling. **Influence-only** is WATERMARK-I: after exact invalidation it ranks solely by h, isolating whether population change calibration improves quality. A post-review **random-invalidated** control receives the same invalidation vector and budget but uses 20 locked hash-randomized orderings per task. **Direct-Flip** (preregistered as Direct-VoI) estimates from fit episodes the probability that refreshing one invalidated scope changes the committed answer, conditioning on backend, query kind, event age, observed margin, and source influence, then greedily ranks by that estimate. It predicts answer changes rather than decision-theoretic utility. **Oracle-cut** is the only policy allowed to inspect the test answer; it finds the smallest correct subset within budget and is reported only as an upper bound.

4.4 NATURAL COMMITS AND INTERACTIVE AGENTS

We parse 600 commits each from Click, Pluggy, Requests, and Rich, retaining 799 signature changes in 44 commits and fixing five queries against each parent revision (220 tasks). A disjoint scan of later history finds 1,132 changes; 934 changes in 154 eligible commits yield 487 capped sources and 770 competitive queries. The code evolution is natural; the five AST-derived argmax queries are constructed and fixed from parent state. All intervals resample commits.

The observed partial-refresh failure on these 154 commits motivates a post-diagnostic atomic gate. To preserve temporal separation, its model and threshold are fitted only on the earlier 44 commits. Leave-one-repository-out predictions choose a threshold under a Watermark-I cost ceiling from parent-only query type, source count, budget, margin, ties, and dispersion. Evaluation has zero commit overlap and exposes no current signature or target. A 20,000-draw control matches activations within query/source strata. A later locked hand rule refreshes three named query families; no subset is searched.

Before inspecting any content from ten additional public repositories, we freeze their remote heads, an 800-commit scan per repository, both atomic policies, the original gate fit and threshold, and a repository-macro commit-bootstrap analysis. The replication uses no project, query-family, feature, threshold, or policy selection after outcomes.

Agent sessions call `start`, `wait`, `refresh`, and `commit`. Conditions share prompts, queries, raw values, times, versions, watermarks, and budgets; plain chooses a cut, while WATERMARK enforces the grid's $\hat{p}\hat{m}h$ cut, and commit returns no feedback. DeepSeek-V4-Flash runs 48 cases under three schedules (288 matched case-runs); GLM-5.2-Fast runs one matched 24-case schedule before its predeclared limit.

5 RESULTS

5.1 CALIBRATION-FREE INFLUENCE IMPROVES ACCURACY WITHOUT FIT DATA

WATERMARK-I reaches 90.09% exact accuracy on the preregistered 15,000-task confirmation, compared with 86.99% for matched version-only ranking: +3.09 points $[2.53, 3.62]$ (Table 1). This gain is positive for every seed, backend, and query family; the seed-cluster interval is $[2.68, 3.49]$ and the seed-level t interval is $[2.46, 3.72]$. Calibration adds no measurable quality: WATERMARK-Q reaches 90.20%, and I minus Q is -0.11 points $[-0.49, 0.29]$, wholly inside the locked ± 0.75-point equivalence margin. WATERMARK-I also exceeds Direct-Flip by +0.73 points $[0.38, 1.08]$. Fresh data therefore establish the simplest quality mechanism: exact invalidation plus query influence, with no fitted transition model.

Cost-aware influence improves the service frontier. On the separate selector-confirmation draw, WATERMARK-E simultaneously improves accuracy and serial tool service over hazard-only. Accuracy reaches 89.1%, a +4.60-point gain $[4.05, 5.13]$, while mean serial service falls from 2.924 to 2.803 (-4.2%; paired interval $[-0.136, -0.106]$). Its 0.87-point deficit to WATERMARK-Q meets the locked one-point retention rule by point estimate; the paired interval $[-1.25, -0.48]$ spans the threshold. Calibration selects $\alpha = 1.5, 2, 1.5, 1.5, 1$ across the five seeds.

Table 1: Preregistered method-of-record confirmation at an 18% budget (15,000 new tasks). All rows share one fresh draw; accuracy is percentage exact match. The calibration-free row gains 3.09 points over version-only while matching its service.

Policy	Fit data?	SQLite	Workflow	Repo.	Macro	Δ vs. p	Serial svc.
Hazard-only	Yes	82.0	84.1	86.0	84.0	−2.95	2.94
Version-only (p)	Yes	85.4	87.3	88.3	87.0	0.00	3.15
Direct-Flip	Yes	87.2	90.1	90.8	89.4	+2.36	3.15
WATERMARK-Q (ph)	Yes	**88.7**	**90.8**	91.1	**90.2**	+3.21	3.14
WATERMARK-I (h)	**No**	88.4	90.5	**91.4**	**90.1**	**+3.09**	3.15

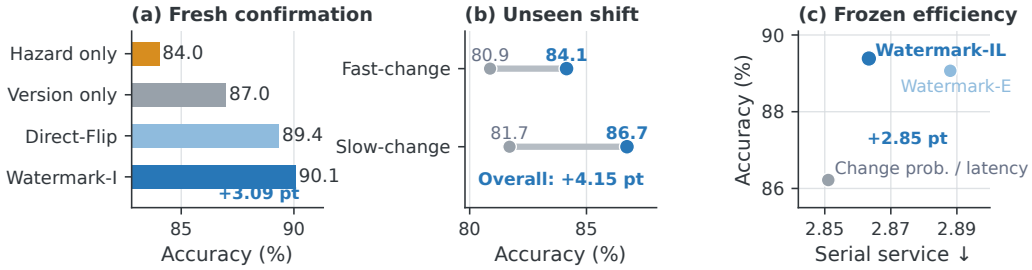

Figure 3: Controlled evidence at paper scale. (a) On 15,000 fresh tasks, query influence raises accuracy from 86.99% to 90.09% over matched version-only ranking and exceeds Direct-Flip. (b) The gain persists under unseen faster- and slower-change regimes. (c) On a frozen 15,000-task efficiency audit, influence-per-latency provides the best accuracy while retaining essentially the same service as probability-per-latency.

A frozen audit supplies stricter version-aware controls (Figure 3c): probability-per-latency reaches 86.22% at 2.851 service, WATERMARK-IL reaches 89.39% at 2.863, and WATERMARK-E reaches 89.07% at 2.888. E exceeds probability-per-latency by +2.85 points $[2.45, 3.25]$; IL saves 0.025 service units over E $[0.018, 0.031]$ while improving accuracy by 0.32 points. Influence and latency therefore identify the efficient point without a fitted transition model.

The complementary grid supplies breadth. WATERMARK-M reaches 88.8%, 93.9%, and 97.2% at 18%, 30%, and 50%, while version-only reaches 86.1%, 92.5%, and 96.4%. At 18% it uses 1.53 refreshes, 51% of full refresh, and 54% of full-refresh serial service. Influence is most valuable in the intended tight-budget regime; its advantage narrows as most invalidated evidence becomes affordable.

5.2 INFLUENCE GENERALIZES ACROSS BACKENDS AND TRANSITION SHIFTS

At the primary 18% budget, influence improves version-only ranking by 5.3 points on argmax tasks (92.9 vs. 87.6), 1.9 points on band queries, and 0.6 on thresholds. This hierarchy explains the aggregate gain: proximity is most informative when the answer depends on competition among a few candidates, whereas version invalidation already solves much of a one-sided threshold decision.

The fresh confirmation localizes the mechanism: I beats version-only in SQLite (+2.96), workflow (+3.18), repository (+3.14), argmax (+6.61), band (+1.74), and threshold (+0.90), all with positive intervals. An earlier exact 15,000-task replay independently finds +3.21 $[2.67, 3.77]$ and equivalence with Q. Quality comes from exact invalidation plus query influence, without calibration data.

Locked controls isolate the mechanism. Across 20 random cuts per task, WATERMARK-I exceeds random invalidated selection by +4.01 points $[3.61, 4.42]$, while version-only adds only +0.92 $[0.52, 1.32]$. Removing adapter-declared volatility classes leaves a +3.66-point $[3.19, 4.12]$ influence gain, and an earlier disjoint confirmation places Q +0.92 $[0.49, 1.37]$ above Direct-Flip. Influence, rather than calibration detail, supplies the dominant ranking signal.

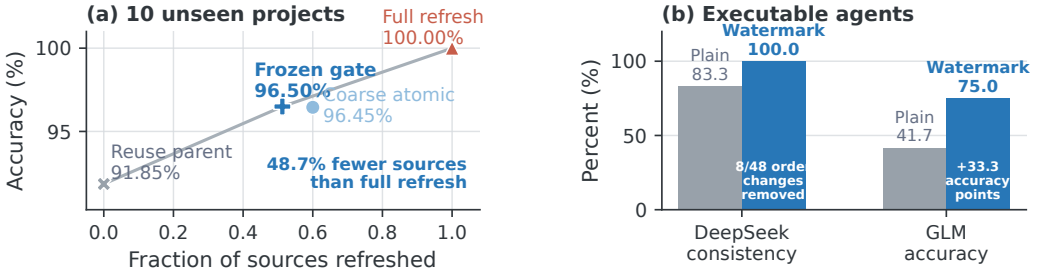

Figure 4: External and end-to-end evidence. (a) Across 265 commits from ten unseen repositories, the frozen atomic gate reaches 96.50% repository-macro accuracy at 0.513 normalized refresh cost; the accuracy-matched coarse policy costs 0.600 and full refresh costs 1.000. (b) In executable OpenCode sessions, event-time control removes all eight DeepSeek delivery-order disagreements and raises GLM exact accuracy from 41.7% to 75.0%. Metrics differ by model and are labeled directly.

Calibration-free influence also survives transition shift. WATERMARK-I reaches 85.44% across 30,000 new tasks, compared with 81.29% for frozen version-only: +4.15 points $[3.68, 4.61]$. It is statistically tied with frozen WATERMARK-Q (+0.31 points, $[-0.05, 0.67]$) and finishes only 0.37 points behind a disjoint in-regime recalibration $[0.08, 0.67]$. Direct-Flip uses its full five-feature table cell for 82.4% of 120,359 scored candidates, so pervasive sparse backoff does not explain the comparison. The shift result confirms the operational advantage: WATERMARK-I transfers without fitting a new transition model.

Exact dependency graphs are not essential. Independently deleting true relevance edges and inserting spurious edges with probability 10% lowers WATERMARK-M accuracy from 88.8% to 87.7%; 20% noise yields 86.6%. The latter remains +2.8 points above hazard-only. The graph test shows that the event-time advantage remains useful with noisy planner structure.

5.3 EXECUTABLE AGENTS PRESERVE DECISIONS ACROSS DELIVERY SCHEDULES

Figure 4b reports sessions in which the model performs every tool action. Across three counterfactual schedules, WATERMARK-M preserves the same answer in all 48 DeepSeek cases, eliminating the eight delivery-order disagreements made by the plain agent. At the same 1.67 refresh calls, accuracy rises from 81.2% to 85.4%; repository inspection improves from 79.2% to 100%. The aggregate accuracy interval $[-4.9, 13.2]$ includes zero because the fixed cut trades 4.2 and 4.1 points on SQLite and workflow tasks. A second model supplies an independent positive result on the matched SQLite/workflow subset: accuracy rises from 41.7% to 75.0%, +33.3 points $[16.7, 54.2]$. The models receive differently ordered transcripts, invoke tools, read refreshed values, compute each query, and commit independently. The complete per-backend accounting is reported in Appendix D.

5.4 FROZEN ATOMIC REFRESH TRANSFERS TO TEN UNSEEN REPOSITORIES

Natural code tests a complementary regime in which sources jointly determine the answer and partial refresh can create a state that never existed. The preregistered external replication provides the strongest evidence for the resulting atomic policy family (Figure 4a). Across 265 commits from ten unseen repositories, frozen coarse atomic refresh reaches 96.45% repository-macro accuracy versus 91.85% reuse: +4.60 points $[3.03, 6.34]$. The frozen gate reaches 96.50%, +4.65 $[2.66, 6.75]$ over reuse, while refreshing 48.7% fewer sources than full refresh. Accuracy matches the coarse rule (+0.05 $[-1.58, 1.61]$), but normalized refresh cost falls by 0.087 $[0.044, 0.131]$. All ten repositories supply eligible commits and none is removed; leave-one-repository-out effects range from +3.94 to +5.12, with every interval excluding zero. Frozen coherent cuts therefore transfer across repository boundaries at lower cost than unconditional refresh.

The preceding natural studies explain this result. On 44 commits, version-only reaches 87.8% versus 81.2% reuse, +6.6 points $[2.8, 10.4]$. A disjoint 154-commit study then shows why the selected cut must remain coherent. Full refresh reaches 100%, reuse 90.13%, partial WATERMARK-I 89.61%,

and version-only 86.75%; partial I fixes 24 reuse errors but disrupts 28 reuse-correct cases by mixing snapshots. Its preregistered transfer gate missed because the +2.86-point interval over version-only crosses zero $[-0.52, 6.36]$, motivating the atomic follow-up.

An active atomic refresh is correct by construction; coarse and gate cover 74.7% and 58.9% of 95 answer-changing tasks. The earlier 154-commit diagnostic confirms the mechanism: the atomic gate reaches 97.14% versus 90.13% reuse, +7.01 points $[4.68, 9.48]$, while a cost-matched random atomic control reaches 96.81% and a fixed three-query-family rule reaches 97.01%. These controls localize the portable gain to atomicity and coarse risk allocation rather than fine-grained learned selection (Table 3).

6 Related Work

Tool-using and asynchronous agents. ReAct and Toolformer establish model–tool interaction (Yao et al., 2023; Schick et al., 2023); AgentBench and τ-bench broaden executable evaluation (Liu et al., 2024; Yao et al., 2024). Graph and asynchronous systems schedule execution (Kim et al., 2024; Qiao et al., 2025; Feng et al., 2026; Shi et al., 2026). Speculative agents validate predictions against realized actions (Ye et al., 2026; Guan et al., 2026); EventTimeBench instead validates already observed values against decision-time revisions—whether a result still describes the current world.

Evolving agent environments. SWE-bench fixes code snapshots (Jimenez et al., 2024); TimeWarp, ProEvolve, and self-evolving suites vary environments across tasks (Ishmam & Marino, 2026; Li et al., 2026; Guo et al., 2026). Our unit is within-call change; counterfactual schedules isolate validity without changing the task or world.

Temporal state in data systems and sequential inference. Logical clocks, MVCC, builds, and stream watermarks separate order, snapshots, and progress (Lamport, 1978; Bernstein & Goodman, 1983; Berenson et al., 1995; Mokhov et al., 2018; Akidau et al., 2013; 2015). Belief state, age, and value-of-information formalize latent state, freshness, and acquisition value (Kaelbling et al., 1998; Kaul et al., 2012; Yates et al., 2021; Lindley, 1956). Watermark connects these foundations after tool return: versions identify invalid evidence and query influence allocates refresh budget.

7 Demonstrated Scope and Extensions

Event-time validity supports two mechanisms. Query influence adds 3.09 points on a fresh controlled draw and 4.15 under transition shift, while Watermark-IL retains quality with lower service. When natural-code sources are jointly constrained, atomic refresh preserves snapshot coherence: frozen policies reach 96.5% across ten unseen projects, and the gate reduces normalized refresh cost relative to the accuracy-matched coarse rule. Existing snapshots, revisions, commits, and hashes make both mechanisms incremental.

The three controlled backends share one evolving-state generator, making cross-backend agreement a consistency test across distinct state representations. The demonstrated scope combines structured exact answers, 463 natural commits from 14 one-language projects, 48 interactive model cases, and a 24-case replication. Open-ended tasks, same-schedule model repeats, partial watermarks, and noisy observations define direct extensions. Across the measured settings, explicit versions separate state recency from arrival order and improve freshness-aware decisions.

8 Conclusion

Arrival time says when a tool finished; event time says which world it observed. EventTimeBench turns that distinction into a causal, executable evaluation. In controlled environments, calibration-free influence improves accuracy across state representations and unseen transition regimes. In natural code, frozen atomic policies preserve coherent cuts across ten unseen repositories while reducing refresh cost. Snapshot versions and decision-time watermarks give asynchronous agents an explicit foundation for reasoning as the world changes.

REFERENCES

Tyler Akidau, Alex Balikov, Kaya Bekiroğlu, Slava Chernyak, Josh Haberman, Reuven Lax, Sam McVeety, Daniel Mills, Paul Nordstrom, and Sam Whittle. Millwheel: Fault-tolerant stream processing at internet scale. *Proceedings of the VLDB Endowment*, 6(11):1033–1044, 2013.

Tyler Akidau, Robert Bradshaw, Craig Chambers, Slava Chernyak, Rafael J. Fernández-Moctezuma, Reuven Lax, Sam McVeety, Daniel Mills, Frances Perry, Eric Schmidt, and Sam Whittle. The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proceedings of the VLDB Endowment*, 8(12): 1792–1803, 2015.

Hal Berenson, Philip Bernstein, Jim Gray, Jim Melton, Elizabeth O'Neil, and Patrick O'Neil. A critique of ANSI SQL isolation levels. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pp. 1–10, 1995.

Philip A. Bernstein and Nathan Goodman. Multiversion concurrency control—theory and algorithms. *ACM Transactions on Database Systems*, 8(4):465–483, 1983.

Guangyu Feng, Huanzhi Mao, Prabal Dutta, and Joseph E. Gonzalez. Concurrency without model changes: Future-based asynchronous function calling for llms. *arXiv preprint arXiv:2605.15077*, 2026.

Yilin Guan, Qingfeng Lan, Fei Sun, Dujian Ding, Devang Acharya, Chi Wang, William Yang Wang, and Wenyue Hua. Dynamic speculative agent planning. In *International Conference on Learning Representations*, 2026.

Dadi Guo, Tianyi Zhou, Dongrui Liu, Chen Qian, Qihan Ren, Shuai Shao, Zhiyuan Fan, Yi R. Fung, Kun Wang, Linfeng Zhang, and Jing Shao. Towards self-evolving agent benchmarks: Validatable agent trajectory via test-time exploration. In *International Conference on Learning Representations*, 2026.

Md Farhan Ishmam and Kenneth Marino. Timewarp: Evaluating web agents by revisiting the past. *arXiv preprint arXiv:2603.04949*, 2026.

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.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

Sanjit Kaul, Roy Yates, and Marco Gruteser. Real-time status: How often should one update? In *Proceedings of the IEEE INFOCOM*, pp. 2731–2735, 2012.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An llm compiler for parallel function calling. In *International Conference on Machine Learning*, 2024.

Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

Guangrui Li, Yaochen Xie, Yi Liu, Ziwei Dong, Xingyuan Pan, Tianqi Zheng, Jason Choi, Michael Morais, Binit Jha, Shaunak Mishra, et al. The world won't stay still: Programmable evolution for agent benchmarks. *arXiv preprint arXiv:2603.05910*, 2026.

Dennis V. Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Ke Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, 2024.

Andrey Mokhov, Neil Mitchell, and Simon Peyton Jones. Build systems à la carte. *Proceedings of the ACM on Programming Languages*, 2(ICFP):79:1–79:29, 2018.

Shuofei Qiao, Runnan Fang, Zhisong Qiu, Xiaobin Wang, Ningyu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Benchmarking agentic workflow generation. In *International Conference on Learning Representations*, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

Kou Shi, Ziao Zhang, Shiting Huang, Avery Nie, Zhen Fang, Qiuchen Wang, Lin Chen, Huaian Chen, Zehui Chen, and Feng Zhao. Asynctool: Evaluating the asynchronous function calling capability under multi-task scenarios. *arXiv preprint arXiv:2605.27995*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Roy D. Yates, Yin Sun, D. Richard Brown, Sanjit K. Kaul, Eytan Modiano, and Sennur Ulukus. Age of information: An introduction and survey. *IEEE Journal on Selected Areas in Communications*, 39(5):1183–1210, 2021.

Naimeng Ye, Arnav Ahuja, Georgios Liargkovas, Yunan Lu, Kostis Kaffes, and Tianyi Peng. Speculative actions: A lossless framework for faster agentic systems. In *International Conference on Learning Representations*, 2026.

A Controlled Generation and Validation

Table 2: Study-to-claim map. Fresh controlled draws support influence and efficiency; disjoint natural commits support coherent cuts; OpenCode sessions test integration.

Layer	Independent units	Rows	Controller	Claim
Selector confirm	15,000 tasks	105,000	I/Q/E	quality + efficiency
Calibration audit+eff.	15,000 tasks	135,000	I/Q/E/IL	class-free + cost
Direct-Flip confirm	15,000 tasks	45,000	Q/flip	strong selector
Calibration shift	30,000 tasks	150,000	I/frozen/refit Q	unseen shift
Fresh I confirm	15,000 tasks	75,000	I/Q/flip	method of record
Budget grid	7,200 tasks	237,600	M	budget + graph noise
Natural code	44 commits	220	semantic/version	real changes
Natural compete	154 commits	770	I/version	partial boundary
Natural gate	44+154 commits	990	gated atomic	coherence + cost
Natural replication	265 commits	1,325	frozen atomic	cross-project
OpenCode–DS	48 cases	288	M	schedules
OpenCode–GLM	24 cases	48	M	model replication

For each seed and budget, fit, tuning, and test partitions use disjoint episode identifiers. The multi-budget grid uses calibration and test partitions; the fresh confirmation adds a separate tuning split for α. Backend evolution, query construction, and transport latency use separate random streams. Threshold and band offsets are sampled around the tick-0 score before future values are read by the query generator. Argmax tie-breaking uses the earliest listed scope. Test labels are computed only after an episode is fully instantiated and are available exclusively to the evaluator and named oracle.

The source distribution is 35% stable, 40% active, and 25% volatile. A SQLite mutation adds one of $\{-9, -5, -2, 2, 5, 9\}$ and increments the partition version. Workflow events update stage, queue length, or blocked status; the scalar query feature is $4 \cdot \text{stage} + \text{queue} - 3 \cdot \mathbf{1}[\text{blocked}]$. Repository revisions change parameter count, async status, or return annotation; the AST-derived feature is parameter count plus twice the async indicator plus an ordinal annotation code.

B Statistical Procedure

Every controlled comparison is paired by seed, backend, and base-task identifier. Confidence intervals resample independent tasks with replacement 3,000 times and report percentile endpoints. The preregistered Watermark-I confirmation additionally reports a 20,000-draw seed-cluster bootstrap and a seed-level t interval over five seed effects. The random baseline averages 20 locked hash-randomized orderings within task before paired resampling, so random replicates are never treated as independent tasks. Natural-repository intervals resample coherent commits within project 20,000 times, recompute each project score, and macro-average projects. Agent accuracy intervals first average schedules within case and then resample cases 20,000 times. Repeated schedules and five query types from one commit are never counted as independent units. For the natural confidence gate, leave-one-repository-out calibration predictions select the threshold on 44 earlier commits; reported effects then resample the 154 evaluation commits. The matched-random distribution uses 20,000 draws while fixing activation counts within query-type and source-count strata. The external replication resamples commits within each of ten repositories and macro-averages projects.

Table 3: Commit-clustered natural-code comparisons. Accuracy and effects are percentage points unless marked; calibration and evaluation use disjoint pinned ranges.

Comparison	Units	A	B	Effect [95% interval]
Version vs. reuse	44	87.8	81.2	+6.6 [2.8, 10.4]
Semantic vs. version	44	86.6	87.8	-1.2 [-4.0, 0.7]
WATERMARK-I vs. version	154	89.61	86.75	+2.86 [-0.52, 6.36]
WATERMARK-I vs. reuse	154	89.61	90.13	-0.52 (descriptive)
I vs. version, changed	76 tasks	31.58	38.16	-6.58 (descriptive)
Atomic gate vs. reuse	154	97.14	90.13	+7.01 [4.68, 9.48]
Atomic gate vs. WATERMARK-I	154	97.14	89.61	+7.53 [4.81, 10.52]
Atomic gate vs. matched random	154	97.14	96.81	+0.34 [-0.21, 0.95]
Coarse atomic vs. reuse	154	97.01	90.13	+6.88 [4.55, 9.35]
Gate vs. coarse atomic	154	97.14	97.01	+0.13 [-0.52, 0.78]
Gated partial I vs. reuse	154	89.35	90.13	-0.78 (descriptive)
Gate vs. full (refreshes)	770 tasks	1.684	3.162	-46.7%
Full refresh, compete	154	100.0	–	endpoint
Fresh coarse vs. reuse	265	96.45	91.85	+4.60 [3.03, 6.34]
Fresh gate vs. reuse	265	96.50	91.85	+4.65 [2.66, 6.75]
Fresh gate vs. coarse	265	96.50	96.45	+0.05 [-1.58, 1.61]
Fresh gate vs. coarse, cost	265	0.513	0.600	-0.087 [-0.131, -0.044]

C Additional Controlled Results

The calibration audit was locked before five new seeds were generated. It uses the same 900/400/1,000 fit/tune/test episodes per backend as the first confirmation, withholds the stable/active/volatile class in its first contrast, and tunes both cost-aware policies over the same exponent grid and serial-service constraint. The later locked replay adds two version-aware controls on these immutable tasks and exactly reproduces every saved WATERMARK-E outcome before comparison.

WATERMARK-I exceeds frozen version-only by +4.15 points $[3.68, 4.61]$, is tied with frozen Q (+0.31, $[-0.05, 0.67]$), and trails refit Q by just 0.37 $[0.08, 0.67]$. Direct-Flip uses its full five-feature table cell for 82.4% of 120,359 scored candidates and backs off for 17.6%.

In the complementary grid, the named oracle reaches 98.5% at 18% using 0.26 calls on average. It can use zero calls when the stale answer already matches the current answer, and it can remain wrong when no subset within the budget recovers the label. This endpoint is deliberately excluded from learned-policy comparisons because it observes the test target.

Table 4: Paired effects in the second 15,000-task audit. Accuracy effects are percentage points; service effects are benchmark units.

Comparison	Effect	95% interval
Class-free ph vs. class-free p, accuracy	+3.66	[3.19, 4.12]
Class-free ph vs. class-aware ph, accuracy	+0.10	$[-0.27, 0.45]$
WATERMARK-E vs. cost-aware hazard, accuracy	+4.99	[4.48, 5.52]
WATERMARK-E vs. cost-aware hazard, service	+0.162	[0.148, 0.177]
WATERMARK-E vs. hazard-only, accuracy	+4.62	[4.11, 5.15]
WATERMARK-E vs. hazard-only, service	-0.046	$[-0.060, -0.032]$
Cost-aware hazard vs. hazard-only, accuracy	-0.37	$[-0.65, -0.09]$
Cost-aware hazard vs. hazard-only, service	-0.209	$[-0.216, -0.201]$
WATERMARK-E vs. p/ℓ^{α}, accuracy	+2.85	[2.45, 3.25]
WATERMARK-E vs. p/ℓ^{α}, service	+0.037	[0.030, 0.044]
WATERMARK-IL vs. WATERMARK-E, accuracy	+0.32	$[-0.03, 0.66]$
WATERMARK-IL vs. WATERMARK-E, service	-0.025	$[-0.031, -0.018]$

Table 5: Fresh Direct-Flip confirmation (preregistered as Direct-VoI; 15,000 tasks, matched 18% budget). Effects are paired accuracy percentage points.

Comparison	Effect	95% interval
WATERMARK-Q vs. Direct-Flip	+0.92	[0.49, 1.37]
Direct-Flip vs. version-only	+2.58	[2.04, 3.12]
WATERMARK-Q vs. version-only	+3.50	[3.05, 3.93]

D INTERACTIVE AGENT PROTOCOL

The interactive pool contains 16 cases per backend. Up to eight challenge cases per backend are selected from episodes where stale reuse changes the answer; the remaining cases are ordinary episodes. Selection is fixed before model runs and shared across conditions. Each 24-case batch is executed by an OpenCode primary agent with filesystem, shell, network, and delegation unavailable. The only callable operations are the four benchmark tools. The controller condition exposes the invalidated scopes and recommended cut after `wait`; `refresh` enforces that cut. The plain condition exposes the same per-scope versions and watermarks, and executes the model's requested cut. Neither condition can read current values without spending a refresh.

The DeepSeek full matrix cost approximately USD 0.12. The GLM runs cost approximately USD 1.22, including one long-reasoning plain batch that reached its token stopping condition without committing and is excluded from matched analysis. No retry or additional model expansion was performed for that matrix. Bounded same-schedule repeat attempts later failed at the provider before inference and incurred no additional cost. Total completed interactive inference cost was approximately USD 1.34.

E NATURAL REPOSITORY CONSTRUCTION

We scan non-merge commits and parse full parent/current Python files. A signature change records positional and required parameter counts, keyword-only parameters, variadic arguments, async status, and return-annotation presence. Only commits with at least two changed functions become query groups; groups larger than eight are deterministically subsampled. Each query is fixed from the parent revision before current signatures are examined. All five query types share the commit as their statistical cluster.

An exploratory learned subset selector on this suite reaches 84.5% at 1.10 calls, compared with 86.6% at 1.76 calls for the method-of-record semantic score. It is a different cost operating point and is not used in matched-call claims. The exact-answer oracle reaches 92.1% at 0.31 calls and remains an explicitly privileged upper bound.

Table 6: Frozen calibration under unseen transition regimes (30,000 tasks). Values are accuracy percentages. Refit Q uses a disjoint in-regime calibration reference.

Policy	Accelerated	Compressed	Overall
Version-only, frozen	80.87	81.71	81.29
Direct-Flip, frozen	83.32	85.21	84.27
WATERMARK-I	84.15	**86.73**	85.44
WATERMARK-Q, frozen	84.51	85.75	85.13
WATERMARK-Q, refit	**84.90**	**86.73**	**85.81**

Table 7: Preregistered Watermark-I confirmation at the primary 18% budget. Accuracy effects are percentage points.

Scope	Comparison	Effect	95% interval
All	WATERMARK-I vs. version-only	+3.09	[2.53, 3.62]
Seed cluster	WATERMARK-I vs. version-only	+3.09	[2.68, 3.49]
Seed t	WATERMARK-I vs. version-only	+3.09	[2.46, 3.72]
All	WATERMARK-I vs. WATERMARK-Q	−0.11	[−0.49, 0.29]
All	WATERMARK-I vs. Direct-Flip	+0.73	[0.38, 1.08]
All	WATERMARK-I vs. hazard-only	+6.05	[5.45, 6.66]
All	WATERMARK-I vs. random-invalidated	+4.01	[3.61, 4.42]
All	Version vs. random-invalidated	+0.92	[0.52, 1.32]
SQLite	WATERMARK-I vs. version-only	+2.96	[2.02, 3.92]
Workflow	WATERMARK-I vs. version-only	+3.18	[2.20, 4.14]
Repository	WATERMARK-I vs. version-only	+3.14	[2.26, 4.02]

F ARTIFACT MAP

The anonymous supplement contains the benchmark generator, analysis scripts, exact configuration, aggregate and task-level CSV files, natural public-repository parser, OpenCode agent prompts and custom tools, raw model traces, cost logs, and a manifest mapping every paper claim to its generating command. Development data rejected after the first independent review remains labeled as invalid and is not referenced by any table in this manuscript. The artifact is available at `https://anonymous.4open.science/r/event-time/`.

AI USE STATEMENT

Generative AI tools were used to assist with phrasing and language editing of the manuscript and with auxiliary research tasks, including literature exploration, implementation support, and analysis checks. All AI-assisted material was reviewed and verified by the authors, who take full responsibility for the final content, claims, and artifacts.

Table 8: Interactive OpenCode results. N counts independent cases; schedules are counterfactual latency assignments. Accuracy and consistency are percentages.

Model	Controller	N	Sched.	SQLite	Workflow	Repo.	Overall	Calls	Consist.
DeepSeek-V4-Flash	Plain	48	3	91.7	72.9	79.2	81.2	1.68	83.3
	WATERMARK-M	48	3	87.5	68.8	100.0	85.4	1.67	100.0
GLM-5.2-Fast	Plain	24	1	56.2	12.5	–	41.7	1.67	–
	WATERMARK-M	24	1	87.5	50.0	–	75.0	1.67	–