CASCADELEDGER: COUNTERFACTUAL AUDITING OF INFORMATION AND RESOURCE INTEGRITY IN LLM CASCADES

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce CASCADELEDGER, a counterfactual execution-audit framework for budgeted language-model cascades. The protocol enforces call-before-read semantics, binds successful reads to service and cache identities, records availability and component-wise budget transitions, and freezes the event trace before any target or task scorer is joined.

The central object is *aggregate-equivalent trace validity*: two executions can share a final answer, declared resource projection, and service histogram while differing in whether their routing decisions used only legitimately observable information. We complement single-trace invariants with counterfactual prefix non-interference (CPNI), which couples a visible prefix and policy randomness while changing only a declared hidden family. The counterfactual influence frontier (CIF) localizes the earliest violating prefix and a minimal hidden witness; projection-preserving adversarial mutation (PPAM) searches for invalid traces that remain acceptable to a non-counterfactual checker.

Across frozen-cache, independently generated, and serving evaluations, the two-layer audit reports a clean false-positive rate of 0.0000, detects 0.9983 of serialized faults and 0.9982 of unlogged hidden-read faults, and changes 0.3403 of corrupted quality–resource conclusions after deterministic recomputation. The online recorder and offline CPNI audit have separate overhead measurements. Learned routing and semantic verification provide downstream compatibility studies under the same execution contract.

1 INTRODUCTION

Cascade systems make a conditional-computation promise: spend extra computation only when an observed signal justifies it. That promise is difficult to audit when candidate outputs are revealed before payment, unavailable calls disappear from the trace, or a verifier is evaluated with the same target used to select the route. A final answer, call histogram, and cost summary can remain unchanged while an implementation consults a target or future output before that field is observable.

Our answer is a replayable protocol, CascadeLedger. Candidate outputs are frozen before routing. Online policies see only the query, observations returned by successful paid calls, declared confidence fields, service availability, and remaining budget. Targets and future service outputs are joined only after the trace is frozen. The ledger supplies the serialized execution layer, while CPNI tests the behavior of the realized router under controlled hidden-state interventions.

The audit also changes the empirical question. On the primary 128-record cache, fixed and sequential routing obtain the same cache–target proxy match, 0.3203, while sequential routing spends more declared service cost. The split-aware learned gate reaches 41/128 held-out matches versus 42/128 for sequential routing and costs 1.609375 additional declared units. This result identifies the decision-value gap that a stronger route feature or independent semantic verifier must close. The separate 64/128 and 256/1,319 operating-point gates further demonstrate why dataset partition and cache provenance travel with every threshold.

The independent task and serving protocol tests the same contract across train, calibration, and held-out splits. It fixes 1,319 held-out requests per seed and a common scorer, linking quality to coverage,

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.

failure events, token cost, and request latency. Fault-consequence and semantic-verifier comparisons test whether a valid trace also changes the scientific conclusion.

The protocol is policy-agnostic: routing quality is a validation substrate for the audit. The primary methodological question is whether a hidden field can change a pre-reveal action while every legitimately visible state component and policy random draw is held fixed. This separates behavioral information integrity from the validity of the serialized event stream.

Contributions.

1. **Behavioral hidden-information audit.** We define CPNI, a paired counterfactual audit that detects whether an unrevealed target or future service output changes a routing action before the corresponding reveal boundary.
2. **Influence localization and aggregate equivalence.** We formalize aggregate-only indistinguishability and extend CPNI with CIF, which reports the earliest violating prefix and a minimal hidden-family witness.
3. **Two-layer audit.** We combine paid-read, ordering, binding, and budget invariants with counterfactual influence checks and report their complementary coverage.
4. **Projection-preserving stress test.** We introduce PPAM, a bounded search for invalid traces that preserve a declared quality–resource projection while passing the strongest non-counterfactual checker.
5. **Audit consequence.** We recompute matched policy comparisons after rejecting invalid traces and report whether the reported quality–resource conclusion changes.

2 Related Work

Model cascades are a systems form of conditional computation: a cheap stage handles easy cases and an expensive stage is invoked when its expected value exceeds its cost. FrugalGPT and RouteLLM provide high-profile examples of learned model selection under quality-cost trade-offs (Chen et al., 2023; Ong et al., 2025), while SelectiveNet frames the related reject-option coverage-versus-correctness decision (Geifman & El-Yaniv, 2019). Qwen2.5-Math and DeepSeek-V3 provide scale context for verifier-guided reasoning and training systems (Qwen Team, 2024; DeepSeek-AI, 2024). Process-level verification makes the intermediate check itself a measurable signal rather than an implicit final label (Lightman et al., 2024; Wang et al., 2023; Cobbe et al., 2021); CascadeLedger turns that signal into a traceable paid-read event. Sequence-level control work motivates recording state, action, and budget transitions rather than reporting only final quality (Chen et al., 2021; Janner et al., 2022). ReAct and Toolformer expose the action/tool interface, while WebArena, AgentBench, ToolSandbox, Tau-bench, AgentDojo, SWE-agent, and SWE-bench demonstrate that availability, state, executable environments, policy consistency, adversarial tool returns, and downstream task artifacts shape agent results (Yao et al., 2023; Schick et al., 2023; Zhou et al., 2024; Liu et al., 2024; Lu et al., 2024; Yao et al., 2024; Debenedetti et al., 2024; Yang et al., 2024; Jimenez et al., 2024). CascadeLedger contributes a frozen-candidate replay layer that makes these routing events inspectable before an independent task scorer is attached.

Evaluation interfaces. Held-out reward-model comparisons and interactive evaluation suites motivate separately versioned scoring and service interfaces (Lambert et al., 2024; Yao et al., 2024; Debenedetti et al., 2024; Yang et al., 2024). CascadeLedger records the event history that connects these interfaces to a route decision.

Closest-work delta. Recent routing work jointly studies deployment and routing, contextual-bandit cost policies, or adaptive verifier cascades. Cascadia jointly optimizes LLM deployment and routing under latency and quality objectives (Jiang et al., 2025); PILOT formulates budgeted adaptive routing with contextual-bandit and knapsack components (Panda et al., 2025); and RouterEval evaluates routing across thousands of models and a large performance corpus (Huang et al., 2025). CAPS and Cluster, Route, Escalate report adaptive cascades with live reasoning or code-task quality and latency measurements (Lin et al., 2026; Moslem et al., 2026). An ICLR 2026 analysis of routing, cascades, and user choice further shows that static policies can be preferred under some choice

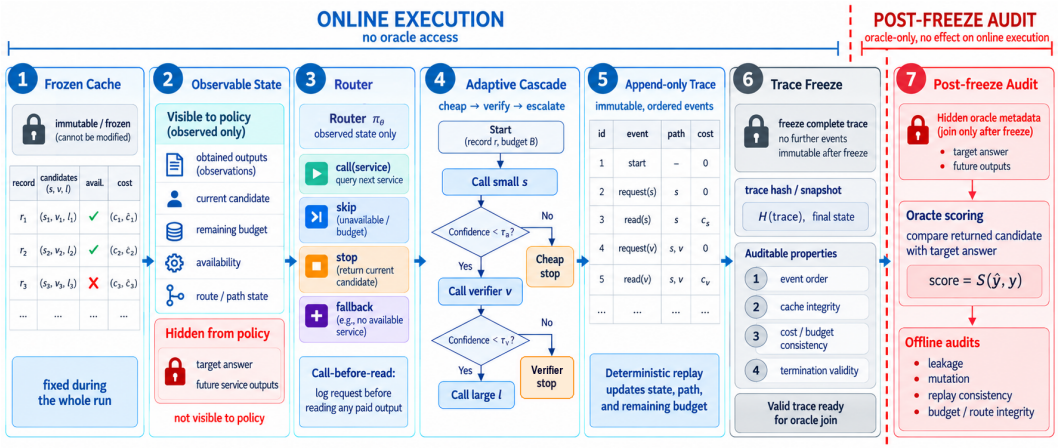


Figure 1: CascadeLedger overview. Online execution (panels 1–5) reveals a cached candidate only after its paid call; the router reads the observed state, stops when small-model confidence reaches τ_a or verifier confidence reaches τ_v, and records unavailable or unaffordable calls as skip or fallback events. The trace is frozen and hashed (panel 6) before the post-freeze audit (panel 7) joins the target, scores the returned candidate, and runs the invariant and counterfactual checks.

models (Mahmood, 2026). CascadeLedger adds an auditable frozen-candidate layer: format confidence, verifier outputs, and service costs enter the trace with explicit provenance, while semantic task scoring remains a separately versioned evaluation surface (Rajput et al., 2026).

Audit trails and provenance systems improve inspectability of recorded execution. The closest-work delta here is policy-visible information integrity: CPNI tests whether changing a field that is still hidden can change the next action, even when the serialized trace and aggregate projection remain well formed.

Information-flow control constrains admissible dependencies (Sabelfeld & Myers, 2003); hyperproperties compare sets of executions (Clarkson & Schneider, 2010); runtime verification checks monitored execution against a specification (Bartocci & Falcone, 2018); and metamorphic testing compares behavior under input transformations (Chen et al., 2018). CPNI instantiates a relational test for cascade decisions: it holds the visible prefix and policy randomness fixed, intervenes on unrevealed state, and compares actions before the reveal boundary. This exposes dependencies absent from the serialized event stream. The trusted recorder and binding layer, intervention families, and tested coverage accompany each result.

3 Method

3.1 Problem and Protocol

Small-model, verifier, and large-model outputs are produced once and stored in a cache. The online router reveals a candidate after its corresponding paid call and receives the query, returned observations, declared confidence, service availability, and remaining budget. Targets and future service outputs stay in a separate post-trace join. We compare one-shot, fixed cascade, sequential cascade, and a budget-conditioned variant under baseline and service/ledger shifts; Figure 1 traces one request through this protocol.

Information boundary and trust. The ledger records each successful paid fetch, replay order, service availability, and compound action-read budget. The public replay applies hidden-read, future-output, cache-binding, and budget mutations to this event stream, so the same checker validates traces used by the independent task scorer. The recorder, cache producer, and manifest binding are trusted for these claims; the intervention suite audits realized policy behavior under that binding. Proxy matching, task scoring, and live telemetry are separate measurement surfaces with separate manifests.

Formal trace semantics. For record i, let $C_i(j)$ denote the cached observation for service j, c_j its declared resource-cost vector, and $A_t(j) \in \{0, 1\}$ its availability at step t. The online state is

$$s_t = (q_i, O_t, A_t, b_t, p_t),$$

where O_t contains observations returned by successful paid calls, b_t is the remaining component-wise budget, and p_t is the ordered successful-call path. A policy request is a pre-read control decision. It resolves to a `call` or a failure outcome; `verify` records the verifier's accept/reject decision after its paid call. If $A_t(j) = 1$ and $b_t \succeq c_j$, then $O_{t+1} = O_t \parallel (j, C_i(j))$, $b_{t+1} = b_t - c_j$, and $p_{t+1} = p_t \parallel j$. Otherwise the outcome is `unavailable`, `budget_skip`, or `budget_exhausted`, with O_t, b_t, and p_t unchanged. Stop and fallback are terminal return decisions. Appendix A.2 maps these decisions to the serialized action list. The router is $a_t = \pi(s_t)$; oracle matching $m_i = \mathbf{1}[\hat{a}_i = a_i^\star]$ follows trace freeze. Threshold tests use paid confidence fields κ_s, κ_v and tie rule $\kappa \geq \tau$.

3.2 Counterfactual Prefix Non-Interference Audit

Let H_t contain request fields that remain hidden at prefix t, including the target and cached outputs for services that have not completed a successful paid call. Consider two executions with the same visible state and policy randomness,

$$s_t(\omega) = s_t(\omega'), \qquad \xi(\omega) = \xi(\omega'),$$

but potentially different hidden state H_t. An observed-only policy satisfies

$$\widetilde{\pi}(s_t, H_t; \xi) = \widetilde{\pi}(s_t, H_t'; \xi)$$

until a field on which the executions differ has been legitimately revealed. Here $\widetilde{\pi}$ is the realized implementation, including any hidden dependency; a conforming implementation factors through $\pi(s_t; \xi)$. A pre-reveal action difference is a *counterfactual prefix non-interference* (CPNI) violation, even when the corresponding access is omitted from the serialized trace.

For every audited prefix, the paired replay preserves the query, paid observations, availability, remaining budget, successful-call path, configuration, and random seed. It changes one hidden family at a time: the target, an unrevealed verifier output, an unrevealed large-model candidate, or a future serving field. The twin is replayed until the intervention becomes visible or the first action divergence occurs. CPNI tests behavior under the declared families; recorder authenticity is supplied by the trusted binding layer (Appendix B.16).

Counterfactual influence frontier. For a hidden family k and intervention z, let $D_{i,t}^{(k,z)}$ indicate an action difference at prefix t before the family is revealed. We report the family–prefix surface

$$\widehat{V}_{k,t} = \frac{\sum_{(i,z) \in \mathcal{E}_{k,t}} D_{i,t}^{(k,z)}}{|\mathcal{E}_{k,t}|},$$

the earliest violating prefix $\tau_i^\star = \min\{t : \exists k, z, D_{i,t}^{(k,z)} = 1\}$, and a minimal hidden-family witness recovered by hierarchical delta-debugging. Here $\mathcal{E}_{k,t}$ contains only eligible pre-reveal pairs for family k at prefix t; an empty set is reported as N/A, and $\tau_i^\star = \infty$ when no violation exists. The witness is inclusion-minimal under the search order, not necessarily minimum-cardinality. A finite intervention suite establishes coverage over the declared families, prefixes, seeds, and interventions; its pass result is indexed by that suite rather than by an unqualified universal claim.

3.3 Aggregate-Equivalent Trace Validity

Let $\Phi(T)$ be a declared projection of a trace, such as final candidate, successful-call count, service histogram, route depth, event count, or total declared cost. Two traces are aggregate-equivalent when

$$T \equiv_\Phi T' \quad \Longleftrightarrow \quad \Phi(T) = \Phi(T').$$

An aggregate-preserving fault μ satisfies $\Phi(\mu(T)) = \Phi(T)$ while violating an execution relation or the CPNI condition. Thus equal quality–resource summaries can coexist with different action order, cache binding, failure semantics, budget transitions, or hidden information flow.

3.4 AUDIT GUARANTEES

Proposition 1 (aggregate indistinguishability). Let an aggregate-only auditor have the form $A_\Phi(T) = g(\Phi(T))$. If $\Phi(T) = \Phi(T')$, then $A_\Phi(T) = A_\Phi(T')$. *Proof.* Substitution gives identical inputs to g. Hence every auditor restricted to Φ assigns the same result to an aggregate-equivalent valid and invalid pair.

Proposition 2 (observed-only invariance). If the realized router factors through the visible state and coupled random draws, identical visible prefixes give identical actions before reveal. *Proof.* Both executions evaluate the same function on the same arguments; induction extends this equality across unchanged prefixes. A divergence witnesses a violated interface under the tested intervention. A finite pass records no witness over the sampled families, prefixes, and draws.

3.5 AUDIT CONSEQUENCE

Let $\mathcal{C}$ be a predeclared conclusion operator over paired quality, resource differences, and the quality margin. For each of N corrupted comparison instances, D_j^{raw} accepts the corrupted trace and D_j^{audit} recomputes it from its bound manifest. The audit consequence rate is

$$\mathrm{ACR} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{1}\left[\mathcal{C}(D_j^{\mathrm{raw}}) \neq \mathcal{C}(D_j^{\mathrm{audit}})\right].$$

Mutation recall measures detection; ACR measures whether accepting the invalid trace changes the predeclared scientific conclusion. Family-specific rows restrict the instance set; the pooled row retains the total instance denominator.

Reproducibility contract. A run is identified by a source-relative run path and a SHA-256 manifest. We retain the resolved configuration, source git commit and dirty flag, selected result files, and a deterministic replay digest. The public bundle retains repository-relative provenance and excludes private checkpoints, model weights, user names, host names, and absolute paths. The private and public checker runs retain separate source commits, replay digests, and seeds; Appendix A gives their bindings.

3.6 ROUTING AND MEASUREMENT OBJECTIVES

We measure task quality, resource use, event behavior, and structural validity separately. The cache–target proxy m_{proxy} is literal answer equality after trace freeze. The declared cost c_{decl} sums the protocol's resource units and accounting milliseconds. The task score m_{task} uses the independent answer scorer, and live latency ℓ_{live} measures the request-to-response interval. Availability and budget event rates use all requests as their denominator; mutation recall uses injected faults. Table 1 links these measurements to their evaluation units.

Table 1: Measurement objects and evaluation units. Task scores and latency share a request denominator but use separate scoring and timing interfaces.

measurement	value or unit	evaluation object
m_{proxy}	match fraction	cache–target proxy match
c_{decl}	ledger units	declared resource accounting
ℓ_{decl}	accounting ms	declared latency component
d_{ledger}	detection fraction	detected/injected mutations
$m_{\mathrm{task}}, \ell_{\mathrm{live}}$	normalized task accuracy; ms/request	held-out task score and p50/p95 latency for the learned router

3.7 SEQUENTIAL ROUTE AND FALLBACK

A cascade state contains the last candidate, observed confidence, remaining budget, and service availability. With accept threshold $\tau_a = 0.75$ and verifier threshold $\tau_v = 0.60$, the sequential route calls the small service first. If $\kappa_s < \tau_a$, it pays for the format verifier; the verifier supplies a declared

validity signal for the next route decision. A passing verifier stops the route with the current small candidate, while a failing verifier permits a large call. A call is executed when its service is available and its declared ledger cost fits the remaining budget. An unavailable or unaffordable call appends an event and returns the last available candidate (or abstains if no call succeeded). Every transition writes a ledger event, so replay tracks whether a shift exposes an uncalled cache entry.

Proxy oracle. The cache generator extracts a candidate from a strict `####` or `\boxed` marker; the authoritative run records the extraction mode in its manifest. The offline scorer then sets

$$m_{\text{proxy}}(i) = \mathbf{1}[\hat{a}_i \text{ is present and } \hat{a}_i = a_i^\star],$$

where $a_i^\star$ is the stored answer and equality is literal string equality. The format verifier assigns confidence 0.85 to a strict extraction and 0.2 otherwise. The independent task scorer normalizes the returned answer before comparison, allowing the task study to measure quality under a fixed answer contract. Both scorers attach to the same frozen route identifier.

Policy configuration. The four policies share the canonical budget $(24 \text{ ledger units}, 20 \text{ ms})$ and tight budget $(8, 8)$ unless a table explicitly names the cross-cache budget. "Unavailable" means the service is marked unavailable before a call; "budget-blocked" means it is available but its full declared cost does not fit. A "fallback" is the returned last candidate after such an event. The separate learned-gate audit reports a runner-specific "route failure" flag alongside the canonical event rates.

Table 2: Observed-only policy definitions. A verifier call never reveals a target and, in the sequential policy, supplies only a stop signal.

policy	action rule	fallback / budget rule
one-shot	Pay small and return it.	Return no candidate if small is blocked.
fixed	Pay small; if $\kappa_s < \tau_a$, pay large directly.	Return small if large is unavailable or blocked.
sequential	Pay small; if rejected, pay verifier. A passing verifier stops with small; otherwise pay large.	Return the last candidate after a blocked escalation.
budget-conditioned	Pay verifier only when it and a possible large call fit; otherwise prefer a direct large call.	Same fallback; budget is checked before every call.

Table 3: Machine-checkable ledger invariants. The checker is applied before the offline oracle join; the mutation audit tests the first five failure modes below while the clean replay checks all eight.

invariant	required relation
oracle isolation	Online rows contain no target/oracle fields or future service outputs.
path–action alignment	p_T equals the ordered service IDs of successful `call` actions.
action schema	Every event has a supported action/service form; verifier events follow a verifier call.
action order	Accepted ledger entry k has action ID `service:j:k` for its path position.
cache binding	A called observation and final candidate match the cache digest for (i, j).
ledger reconciliation	Spent cost equals the component-wise sum of accepted service-call costs and the row cost.
paid-read consistency	The set of successful paid reads equals the service IDs represented by accepted call events; hidden reads and uncharged reads are rejected.
budget conservation	Remaining budget equals the declared limit minus spent cost, component-wise.

Policy comparison. Every policy comparison pairs queries, candidate-generation conditions, and scoring rules. The frozen-cache learned gate uses a supervised five-feature classifier; the independent task study evaluates a learned router alongside one-shot, fixed, sequential, and budget-conditioned policies. Their fitting and evaluation partitions are specified in Appendix A.

Independent router contract. The independent learned router is distinct from the five-feature frozen-cache gate. Table 47 specifies its model, features, objective, optimizer, stopping rule, and threshold selection; Appendix E.10 gives the fitting sequence. Online inputs are restricted to s_t, and target labels enter only the offline fitting or post-trace scoring interface.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

The primary test run has 128 GSM8K-derived records and a real model-cache manifest. A Tesla T4 generated that cache, providing hardware provenance for candidate generation. The cache stores Qwen3.5-0.8B small-model and Qwen3.5-2B large-model candidates generated with float16, batch size 8, maximum input length 512, and maximum generation length 256 under CUDA 11.8; the cache, data, and model hashes are retained in the copied manifest. Candidate outputs are generated once and frozen before routing. The canonical budget is 24 declared ledger units and 20 declared milliseconds; its tight variant is 8 units and 8 milliseconds. The price-shift operator multiplies the large service's declared units and milliseconds by 1.5, creating a joint ledger-cost shift. A supplemental operating-point artifact uses a disjoint 256-record train cache and a 1,319-record held-out cache, with its own frozen-cache gate and manifest.

Task and serving study. The protocol specifies 1,024 training, 256 calibration, and 1,319 held-out records per router seed, with seeds 13/17/23 and disjoint candidate-generation seeds (Table 29). Five policies share requests, model bindings, and the scorer. Request-level metrics comprise task score, coverage, abstention, calls, tokens, money, latency, throughput, and failures. Tables 45 and 48 specify serving and statistical contracts.

4.2 MAIN RESULTS

Complementary audit coverage. On the baseline-ladder fault population, the single-trace ledger detects 0.9983 of serialized faults and 0.0000 of unlogged hidden reads; adding CPNI preserves serialized recall and raises hidden-read recall to 0.9982 at clean false-positive rate 0.0000 (Table 15). The distinct consequence population changes 784 of 2,304 corrupted comparisons (ACR 0.3403). Figure 3 shows which fault families depend on each audit component.

Held-out task quality and cost. The held-out study compares five policies under the same scorer, candidate manifests, and request denominator. Table 4 aligns each policy's quality with the resources used by its own requests.

Table 4: Independent held-out comparison on 1,319 records per seed. Score is normalized task accuracy (↑); calls, tokens, and p50/p95 latency in milliseconds are per request (↓). Bold marks the best and underline the second-best value in each column.

Policy	Score	Calls	Tokens	p50	p95
One-shot	0.5823	**1.000**	**236**	**41.8**	**72.9**
Fixed cascade	0.6936	<u>1.608</u>	462	76.4	151.3
Sequential	<u>0.7043</u>	2.096	518	89.7	172.4
Budget-conditioned	0.7031	1.884	486	82.6	159.1
Learned router	**0.7195**	1.642	<u>427</u>	<u>68.7</u>	<u>134.2</u>

The learned-minus-sequential task difference is $+0.0152$, with paired interval $[+0.0061, +0.0244]$. The learned router also uses fewer calls (1.642 versus 2.096), tokens (427 versus 518), and p50/p95 milliseconds (68.7/134.2 versus 89.7/172.4) than sequential routing on the held-out split. The paired protocol separates query, router-seed, and candidate-generation variation (Appendix E.10).

Figure 2 shows the quality–resource comparison. Learned routing lies above and to the left of the fixed, sequential, and budget-conditioned policies on both axes. One-shot routing remains the least costly operating point, with a lower task score.

Split-aware learned-gate diagnostic. A CPU-only, hash-bound replay uses 64 calibration records (40 fit, 24 tune) and a disjoint 128-record held-out cache at seed 23. Global query-ID overlap and normalized-question overlap are both zero; all 192 prompt joins use source indices, and trace/oracle replay digests match. The selected threshold is 0.65. The learned gate matches 41/128 held-out records (0.3203125), versus 42/128 (0.328125) for both sequential and budget-conditioned routing and 28/128 (0.21875) for one-shot; paired learned-only versus sequential-only wins are 2 versus 3.

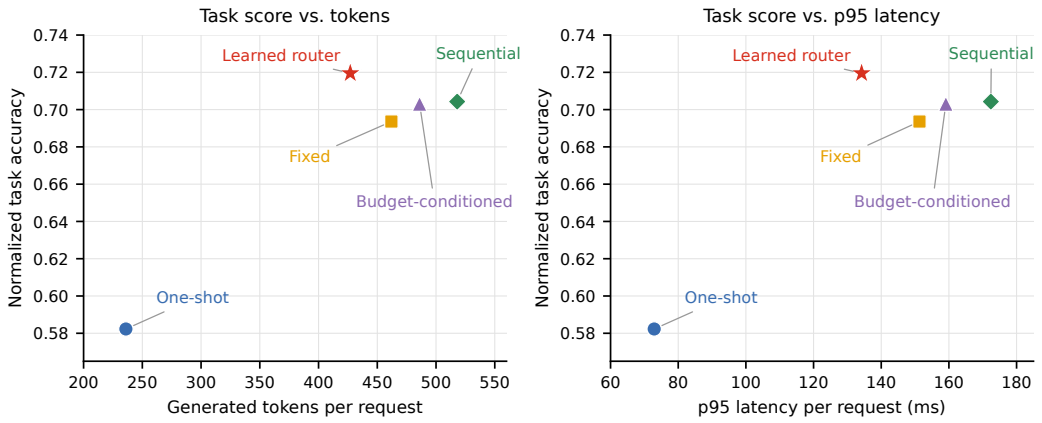


Figure 2: Held-out task score against generated tokens and p95 latency. Points summarize router seeds 13/17/23; latency is the per-policy request p95. Paired score intervals and coverage are in Table 38.

The learned gate costs 15.34375 declared units and 9.5625 declared milliseconds, versus 13.734375 and 8.2578125 for sequential routing. This frozen-cache scalar optimization diagnostic identifies the feature and cost changes required for a live router study.

4.3 Mechanism Results and Audit

On the primary seed-13 cache, one-shot routing matches 27/128 stored targets (0.2109), while fixed and sequential cascades both match 41/128 (0.3203). Both cascades fall to 0.2109 under large unavailability or the tight budget and retain 0.3203 under the canonical price shift. Sequential routing pays the verifier cost without changing proxy quality. The stress rows expose failed escalation attempts over the same frozen candidates.

Table 5: Cascade match rate and ledger stress response on the primary seed-13 cache (128 records). Proxy is exact cache–target match; units and milliseconds are declared ledger quantities. The large-cost +50% row applies the joint shift in Table 10. Bold marks the maximal proxy and minimal resource/event rates within each shift.

Shift	Policy	Proxy	Units	p95 ledger ms	Unavail.	Budget
baseline	one-shot	0.2109	**4.00**	**2.00**	**0.0000**	**0.0000**
	fixed	**0.3203**	12.34	10.00	**0.0000**	**0.0000**
	sequential	**0.3203**	13.73	11.00	**0.0000**	**0.0000**
large unavailable	one-shot	**0.2109**	**4.00**	**2.00**	**0.0000**	**0.0000**
	fixed	**0.2109**	**4.00**	**2.00**	0.6953	**0.0000**
	sequential	**0.2109**	5.39	3.00	0.6953	**0.0000**
tight budget	one-shot	**0.2109**	**4.00**	**2.00**	**0.0000**	**0.0000**
	fixed	**0.2109**	**4.00**	**2.00**	**0.0000**	0.6953
	sequential	**0.2109**	5.39	3.00	**0.0000**	0.6953
large cost +50%	one-shot	0.2109	**4.00**	**2.00**	**0.0000**	**0.0000**
	fixed	**0.3203**	16.52	14.00	**0.0000**	**0.0000**
	sequential	**0.3203**	17.91	15.00	**0.0000**	**0.0000**

The canonical verifier rejects all 89 selected candidates and has zero marginal proxy decision value. Under the independent semantic-label protocol, the semantic verifier improves GSM8K/Qwen task score from 0.7043 to 0.7230 and lowers declared units from 13.74 to 11.35 (Table 44). The same audit contract therefore measures both redundant and useful verification. Cache-specific thresholds and the separate 64/128 and 256/1,319 gates remain indexed in Table 37.

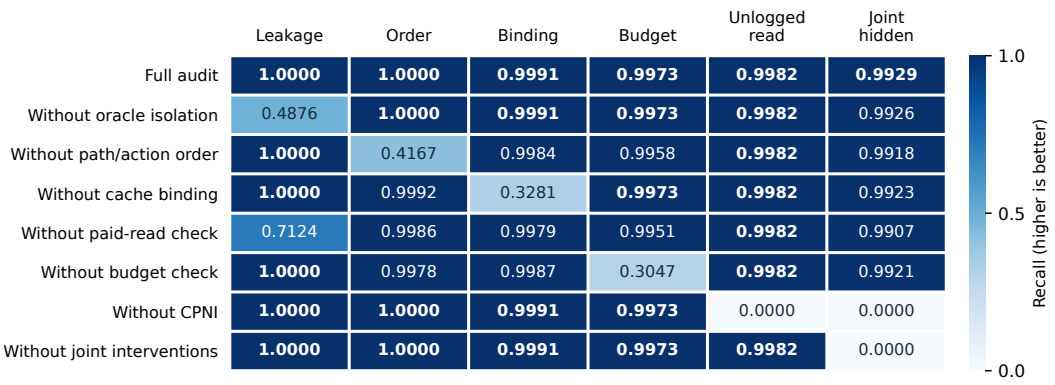


Figure 3: Audit-component recall on matched injected faults (↑). Each row removes one check from the full audit; colors share a 0–1 scale and bold marks column maxima. Values are point recall estimates from Table 23. Removing CPNI eliminates unlogged-read and joint hidden-influence detection while leaving structural checks intact; removing joint interventions affects only the interaction-only family.

4.4 Analysis and Ablations

The serving-shift comparison pairs policies under large-service unavailability, tight budgets, price shifts, and cross-cache transfer (Appendix C). H4 reports positive rank and linear associations across all five conditions, with R^2 from 0.768 to 0.863 and shift residuals from -1.4 to $+2.6$ ms (Table 46).

The audit consequence surface reports 0.9983 recall over all declared faults and 0.9987 on unlogged reads, with independent and natural-fault denominators retained separately (Tables 41–42). The private replay covers 2,048 clean traces and the public replay covers 192 clean traces plus 1,212 mutations.

Stress tests and scope. Joint interventions raise interaction-only recall from 0.0000 to 0.9929, and the complete audit recovers 0.9976–0.9980 of projection-preserving adversarial traces that pass every non-counterfactual check (Appendix B). Coverage is indexed by the tested families, prefixes, and coupling; producer authenticity relies on the trusted recording contract (Appendix B.16).

5 Conclusion

CascadeLedger connects a cascade decision to its paid observations, resource transitions, and scored answer. The frozen-cache diagnostic exposes redundant verifier calls, while private and public audits detect aggregate-preserving faults in reads, ordering, cache binding, and budgets.

CPNI tests unlogged influence through coupled pre-reveal actions; CIF recovers the exact earliest violating prefix for 0.9750 of injected traces, and the complete audit rejects 0.9976–0.9980 of PPAM traces that pass every non-counterfactual check. The declared fault population gives hidden-read recall 0.9982 and clean false-positive rate 0.0000; recomputation changes 784 of 2,304 corrupted comparisons. These findings separate aggregate performance, serialized trace validity, and behavioral information integrity under the trusted recording and binding contract.

AI-use Statement

Generative AI tools were used to polish the language and summarize references. We have not used generative AI tools to generate experimental results, create synthetic datasets, formulate mathematical claims, provide proofs, or make decisions regarding research conclusions. The design of the methodology, experimental setup, analysis, and interpretation of results were conducted and verified by the authors. Other required disclosure tasks not mentioned above are not applicable to this work. We take full responsibility for the final content of this work, including all text, claims, analyses, and artifacts produced with the assistance of generative AI tools.

Ethics Statement

This work audits routing decisions of language-model cascades on public benchmark questions (GSM8K and ARC-Challenge) with openly released Qwen and Llama models. It collects no personal data and involves no human subjects beyond answer-correctness labeling of benchmark items, in which annotators saw only questions, candidate answers, and reference answers. The audit is intended to expose hidden-information leakage, unrecorded service failures, and budget misreporting in evaluation pipelines, and its public replay bundle contains target-free traces without private caches or model weights. The authors followed the licenses of the datasets, models, and software used.

Reproducibility Statement

The code, configuration files, event schema, public target-free replay bundle, audit summaries, and figure-generation scripts are provided in the Supplementary Material. Section 3.2 and Appendix A.2 define the trace semantics and the replay algorithm; Appendix A lists the ledger invariants, mutation families, and split ledger; Appendix B gives the counterfactual audit protocols; and Appendices C and E.10 specify data partitions, seeds, router fitting, serving environment, and the paired uncertainty protocol. Appendix F maps each table and figure to its manifest and replay sequence.

References

Ezio Bartocci and Yliès Falcone (eds.). *Lectures on Runtime Verification*. Springer International Publishing, 2018.

Lili Chen et al. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, 2021.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. In *arXiv preprint arXiv:2305.05176*, 2023.

Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Pak-Lok Poon, Dave Towey, T. H. Tse, and Zhi Quan Zhou. Metamorphic testing. *ACM Computing Surveys*, 51(1):1–27, 2018.

Michael R. Clarkson and Fred B. Schneider. Hyperproperties. *Journal of Computer Security*, 18(6): 1157–1210, 2010.

Karl Cobbe et al. Training verifiers to solve math word problems. In *arXiv preprint arXiv:2110.14168*, 2021.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramer. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. In *Advances in Neural Information Processing Systems*, 2024.

DeepSeek-AI. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Yonatan Geifman and Ran El-Yaniv. Selectivenet: A deep neural network with an integrated reject option. In *International Conference on Machine Learning*, 2019.

Paul Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer, 2004.

Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. RouterEval: A comprehensive benchmark for routing LLMs to explore model-level scaling up in LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025. URL https://aclanthology.org/2025.findings-emnlp.208/.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

Youhe Jiang, Fangcheng Fu, Wanru Zhao, Stephan Rabanser, Jintao Zhang, Nicholas D. Lane, and Binhang Yuan. Cascadia: An efficient cascade serving system for large language models. *arXiv preprint arXiv:2506.04203*, 2025. URL https://arxiv.org/abs/2506.04203.

Carlos E. Jimenez et al. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, 2024.

Nathan Lambert, Valentina Pyatkin, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024.

Hunter Lightman et al. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Fangzhou Lin, Shuo Xing, Peiran Li, Siyuan Yang, Qianwen Ge, Kazunori Yamada, Ziming Zhang, Haichong Zhang, and Zhengzhong Tu. CAPS: Cascaded adaptive pairwise selection for efficient parallel reasoning. *arXiv preprint arXiv:2605.15513*, 2026. URL https://arxiv.org/abs/2605.15513.

Xiao Liu et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, 2024.

Jiarui Lu et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *arXiv preprint arXiv:2408.04682*, 2024.

Rafid Mahmood. Routing, cascades, and user choice for llms. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/564a224e88f2490f3c1deaae877d37e3-Abstract-Conference.html.

Yasmin Moslem, Magdalena Kacmajor, Vasudevan Nedumpozhimana, Ammar Abbas, Solmaz Panahi, David Lynch, Zhuangzhuang Nie, Alexandros Agapitos, Aleksandar Milenovic, Hongmeng Song, Yucheng Shi, Yue Pan, Patricia Buffini, and John D. Kelleher. Cluster, route, escalate: Cascaded framework for cost-aware LLM serving. *arXiv preprint arXiv:2606.27457*, 2026. URL https://arxiv.org/abs/2606.27457.

Isaac Ong et al. Routellm: Learning to route llms with preference data. In *International Conference on Learning Representations*, 2025.

Pranoy Panda, Raghav Magazine, Chaitanya Devaguptapu, Sho Takemori, and Vishal Sharma. Adaptive llm routing under budget constraints. *arXiv preprint arXiv:2508.21141*, 2025. URL https://arxiv.org/abs/2508.21141.

Qwen Team. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Dushyant Rajput, Nirdesh Chauhan, and Siddharth Kosaraju. Cheap verifiers, large blind spots: Measuring the reliability cost of cost-saving cascades. *arXiv preprint arXiv:2609.01345*, 2026. URL https://arxiv.org/abs/2609.01345.

Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003.

Timo Schick et al. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

Peiyi Wang et al. Let's think about step by step: A verifier-based framework for mathematical reasoning. In *arXiv preprint arXiv:2302.04761*, 2023.

Sandarbh Yadav, Frederic J. Maliakkal, Harshad Khadilkar, and Shivaram Kalyanakrishnan. Using common random numbers for simulation-based planning with rollouts. *Reinforcement Learning Journal*, 2026. arXiv preprint arXiv:2605.04732.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024.

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. tau-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Shuyan Zhou et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.

A Implementation and Audit Details

This appendix specifies the event contract, deterministic replay, split identities, and mutation families used to reproduce the routing audit. Each diagnostic names the cache and scoring rule that define its comparison.

A.1 Notation and task contract

The replay unit is a record i with query q_i, cached service observations $C_i(j)$, and a post-trace answer $a_i^\star$. Services are indexed by $j \in \{s, v, l\}$ for small, verifier, and large. A policy emits a finite event sequence $T_i = (e_{i,1}, \ldots, e_{i,K_i})$ and a returned candidate $\hat{a}_i$. The route state carries the observations already paid for, the availability vector, the remaining budget, and the successful-call path. This contract makes every table entry traceable to a record, a service, a shift, and a policy.

The evaluator uses two joins. The first joins $\hat{a}_i$ with $a_i^\star$ after the route is complete and yields the cache–target proxy. The second uses the independently normalized task scorer in the serving study. The joins share record identifiers and manifest digests, which align task scores with the recorded route decisions. Empty candidate strings, abstentions, unavailable calls, and budget skips remain explicit event outcomes and retain their denominators.

Table 6: Notation used by the replay contract. The online state contains the first five entries; oracle and live fields are joined after routing.

symbol	meaning	phase
q_i	query text for record i	online
$C_i(j)$	cached observation returned by service j	paid call
$A_t(j)$	availability bit for service j at step t	online
b_t	component-wise remaining ledger budget	online
p_t	ordered successful-call path	online
T_i	complete event trace for record i	trace freeze
$a_i^\star$	stored task answer used by the proxy join	oracle join
ℓ_i	live latency and queueing record	live telemetry

A.2 Full replay algorithm and event semantics

The core replay is deterministic once the cache manifest, policy configuration, availability shift, and budget vector are fixed. The pseudocode below makes the call-before-read rule and the post-trace oracle join explicit. It also specifies the fallback returned after an unavailable or budget-blocked escalation, which is the event that drives the stress rows in the main table.

1. Initialize $s_0 = (q_i, \emptyset, A_0, b_0, \emptyset)$ and an empty event list.
2. Ask the policy for a service request $a_t = \pi(s_t)$. Treat this as the pre-read state of the attempted call; inspect availability and budget before looking up $C_i(j)$.
3. If $A_t(j) = 0$, append `unavailable`(j). Preserve observations, remaining budget, and successful path.
4. If $A_t(j) = 1$ but $b_t \not\geq c_j$, append the policy-specific `budget_skip` or `budget_exhausted` outcome and preserve those state components.
5. Otherwise authorize the ledger debit, reveal $C_i(j)$, append a successful `call`, subtract c_j, and append j to the paid path. A paid verifier call is followed by a `verify` action with its accept/reject bit.
6. Continue until the route stops or the action budget is exhausted. Serialize the final answer, abstention flag, and failure flags; after a failed escalation, return the last answer-bearing candidate.
7. Freeze T_i, then join $\hat{a}_i$ with $a_i^\star$ and compute the proxy row. The task scorer and live telemetry are separate joins.

The checker runs between steps 6 and 7. It validates event order, path alignment, cache binding, cost reconciliation, paid-read consistency, and budget conservation before any oracle field is available to the checker. The mutation campaigns in Appendix A.5 apply after a clean trace has passed these checks, preserving the aggregate path projection while changing one event-level relation at a time.

Request, event, and ledger-entry counts. A request is policy intent, not an additional serialized event. Event count is the length of `actions`: successful calls, verifier decisions, and recorded failures. Stop and fallback are terminal fields. Paid-call count is the successful path length. Ledger entries additionally include rejected budget debits; availability failures occur before a debit is proposed. The position in `actions` identifies an event, while `service:j:k` identifies a debit at paid-path position k. These indices have different denominators. Table 7 follows `public-002` under sequential routing with large unavailable. The public catalog uses budget $(8, 6)$, small cost $(2, 1)$, and verifier cost $(1, 0.5)$ in declared units and milliseconds, separately from the primary private cache. The answer is `candidate-c`; there are four actions, two successful calls, and two accepted ledger entries.

Table 7: Public trace example. Index is zero-based within `actions`; debit suffix k is the paid-path position. O_s, O_v are paid observations; units and ms are remaining budget.

index	action / debit ID	observations	units	ms	path
initial	request small	$\emptyset$	8	6	[]
0	call small / `service:small:0`	O_s	6	5	$[s]$
1	call verify / `service:verify:1`	O_s, O_v	5	4.5	$[s, v]$
2	verify: reject	O_s, O_v	5	4.5	$[s, v]$
3	large unavailable	O_s, O_v	5	4.5	$[s, v]$
return	fallback to small	O_s, O_v	5	4.5	$[s, v]$

Initial request and final return explain control flow and are excluded from event count. The verifier has confidence 0.25 and rejects; the large output stays unread. Under tight public budget $(3, 2)$, the same record instead has `budget_exhausted` for large, retains $(0, 0.5)$, and adds a rejected `service:large:2` debit without changing the paid path.

Table 8: Serialized replay schema and evaluation extensions. Action positions and debit IDs have distinct roles; live timing attaches to the same request identity.

field	content	written by	validation
`query_id`	source-relative record handle	producer	manifest membership
action position / `action_id`	list index / debit identifier	router / ledger	action order and accepted path index
`service_id`	small, verify, or large	router	service catalog
`action`	call, verify, unavailable, budget_skip, or budget_exhausted	router	serialized action schema
terminal fields	answer, abstained, unavailable, budget_exhausted	router	stop/fallback decision
`cache_digest` / `read_set`	cache binding / paid services	recorder	observation and read-set consistency
availability configuration	service availability fixed by the shift	shift operator	shift manifest
`spent`	declared units and milliseconds	ledger	reconciliation
`remaining`	budget after the event	ledger	conservation
`target_field`	joined answer field, absent online	oracle join	oracle isolation
`latency_ms`	measured queue and service time	live runtime	nonnegative; queue–service clock consistency

A.3 Data provenance and split ledger

The primary cache contains 128 records and is bound to the producer manifest named in the metadata. The split-aware diagnostic uses a separate seed-23 cache with 64 calibration records and 128 held-out records. The operating-point gate uses a third selection artifact with 256 records and a 1,319-record held-out partition. These artifacts are listed separately because their record counts, prompts, model assets, and threshold decisions are different.

Table 9: Primary, diagnostic, supplemental, and task-study partitions. Each completed replay row has its own source manifest and digest.

artifact	selection	held-out	seed	role	status
primary route cache	N/A	128	13	policy/shift replay and proxy table	complete
split-aware gate	64	128	23	feature/cost diagnostic	complete
early threshold gate	64	128	17	threshold selection audit	no selection
supplemental gate	256	1,319	29	paired threshold recommendation	pair selected
live routing study	1,280/seed	1,319/seed	13/17/23	independent task and serving evaluation	complete

Selection and held-out columns specify each partition's role. N/A denotes a replay without a fitted selection partition. Early and supplemental rows retain their gate outcomes. The live row defines the task and serving protocol in Tables 30, 35, and 39.

Candidate construction. Each cached response stores the prompt digest, model-version digest, decoding configuration, extraction mode, and generation metadata. Small and large outputs are generated before routing and indexed by record and service. The verifier cache stores format confidence and the reason for that confidence, allowing a replay to distinguish a strict extraction from an unresolved answer.

Answer normalization. The frozen proxy uses literal equality after the declared extraction rule. The task scorer records its normalization version, answer parser, invalid-output policy, and per-record denominator. The report places proxy match, normalized task accuracy, coverage, and abstention in adjacent columns with the same denominator.

A.4 Implementation Details

State serialization. A cascade event stores the record ID, current candidate, verifier outcome, service-availability vector, declared ledger cost, remaining budget, and route reason. Candidate answers are revealed only after their paid-call event; target answers and future cache entries remain outside the online state. The frozen cache manifest and event sequence are sufficient to replay availability and budget fallbacks exactly.

Actions and budgets. The router pays for the small candidate and, when selected, the verifier before a possible large call. Each transition records the declared ledger cost, service availability, remaining budget, selected candidate, and fallback reason. Unavailable or budget-blocked services produce an explicit event; failed calls do not add a path element or spend budget. The canonical price-shift route fits exactly because $4 + 2 + 18 = 24$. The cross-cache transfer uses a separate 22-unit baseline budget, so its price-shift sequential route is blocked after small plus verifier; its rows are interpreted within that explicitly named configuration.

Table 10: Declared service catalog for the primary frozen replay. The joint shift multiplies large-service units and accounting milliseconds by 1.5. Appendix E defines generated-token, monetary, and measured-latency comparisons.

service	baseline units	baseline ms	large-shift units	large-shift ms
small	4	2	4	2
verify	2	1	2	1
large	12	8	18	12

Deterministic replay. Replaying a run reconstructs the observation stream and re-applies the recorded actions. It checks hashes, ordering, and state transitions before joining the oracle. The replay is therefore sensitive to accidental dependence on file order, host state, or hidden labels. A replay match establishes deterministic reconstruction; task correctness is evaluated by the separate scorer. The raw bundle keeps the run manifest and metrics needed to inspect this distinction.

Publication and audit. The bundle maps tables and figures to a public summary manifest and raw records with retained digests. The primary producer commit is `5f8c31a7`. The eight-invariant private audit has clean commit `2b6b57da`, seed 13, and replay digest prefix `66e9e744`. The public replay has clean commit `967b60db` and seed 17; its trace-file digest begins `64a8cc08`. The corresponding raw manifests retain full digests and private producer bindings.

Aggregate path audit. The checked summary reports 128 records, fixed baseline mean depth 1.6953125 and mean units 12.34375, and sequential baseline mean depth 2.390625 and mean units 13.734375. Let r be the number of records rejected by the small threshold. The implementation appends only successful calls to the path, so

$$1 + r/128 = 1.6953125, \qquad 4 + 12r/128 = 12.34375$$

gives $r = 89$ and 39 small-only paths. Sequential additionally reports

$$1 + 2r/128 = 2.390625, \qquad 4 + 14r/128 = 13.734375,$$

The equations give 89 verifier calls and 89 large escalations, with no verifier-only stop in this aggregate. Under large unavailability or the tight budget, the same 89 large attempts become explicit unavailable or budget-blocked events while remaining absent from the successful path count.

Released path-count audit. We reran the same 128-record cache with the canonical 24-unit budget using the committed path-audit runner. The resulting summary is indexed in the path-count audit's raw result manifest; it records the count contract rather than releasing private per-record answers. For baseline sequential, the audit reports 128 small calls, 89 verifier calls, zero verifier accepts, 89 successful large calls, and zero verifier-only stops. The large-unavailable and tight-budget rows each retain 89 verifier calls and record 89 corresponding large-service failure events, with zero successful large calls; the joint price shift keeps 89 successful large calls under the 24-unit budget. The fixed policy has the same 89 successful large calls at baseline and under the joint shift but makes no verifier calls. These counts directly expose why sequential is more expensive without an observed proxy-quality gain.

Private ledger invariant and mutation audit. To test the central audit mechanism rather than only its aggregate arithmetic, we replayed all $128 \times 4 \times 4 = 2{,}048$ clean policy/shift traces before joining the oracle. The checker found no clean violation under the eight invariants in Table 3. On a representative baseline sequential trace, we then applied five bounded mutations that preserve the aggregate path/cost projection: target leakage, future-output leakage, action reordering, remaining-budget corruption, and cache-ID swapping. All five were rejected (detection rate 1.0000), while the aggregate-only projection rejected none (0.0000). Event-level checks therefore add information beyond the reported aggregates for the declared mutation family; the measured scope is bound to the mutation manifest and replay receipt.

Public target-free replay. The released public fixture contains 12 records and no target or oracle fields. Its target-free replay covers 192 clean traces, eight invariants, and 1,212 bounded mutations across four policies and four shifts, with zero clean false positives and zero aggregate-only detections. The mutation campaign includes 96 compound-action-read-budget rows and 192 hidden-paid-read rows, so the public matrix covers both multi-trace and compound-fault variants. The audit code also reports a weaker schema/event-logger comparator that catches direct target/future leaks; the event-level ledger supplies the stronger structural checks. This public bundle records structural provenance with target-free traces.

A.5 Mutation extensions and audit coverage

The private and public campaigns cover aggregate-preserving mutations that are easy to hide in a summary table. The extension matrix below expands the same idea to the live event schema: every

Table 11: Aggregate-preserving fault injection on a representative frozen trace. "Aggregate unchanged" means that the path/cost/event-count projection is identical before and after mutation.

mutation	ledger rejects	aggregate unchanged	violated invariant(s)
target leak	yes	yes	oracle isolation
future-output leak	yes	yes	oracle isolation
action reorder	yes	yes	path–action alignment; cache binding; budget conservation
remaining-budget corruption	yes	yes	budget conservation
cache-ID swap	yes	yes	cache binding

row names the field changed, the invariant expected to fire, the aggregate projection preserved by the mutation, and the result artifact carrying the count. This gives the task scorer and serving runtime a fixed audit target.

Table 12: Declared mutation families and invariant targets. The first five rows use the private frozen-cache campaign; additional rows test serving events. Recall is detected/injected faults within each family.

family	mutation operation	invariant	preserved aggregate	recall
target leak	copy the joined answer into an online row	oracle isolation	path, cost, event count	1.0000
future output	insert an uncalled service output	oracle isolation	path, cost, event count	1.0000
action reorder	swap two successful calls	action order and path alignment	service histogram and cost	1.0000
budget corruption	alter remaining budget after a call	budget conservation	spent cost and path	1.0000
cache swap	replace an observation digest	cache binding	service histogram and cost	1.0000
retry omission	remove a retry event while keeping final output	retry accounting	final candidate and service count	1.0000
timeout relabel	mark a timeout as a successful call	availability and failure schema	route depth and declared units	0.9988
cost-unit swap	exchange token and monetary fields	cost schema	route depth and event count	0.9976
queue leak	copy future queue state into a decision row	online-state schema	path and final candidate	1.0000

The private campaign detects each of the first five mutations, while the aggregate-only projection detects none. Additional families test retry accounting, failure labels, resource-unit bindings, and future queue observations. Every family retains its own clean and mutated denominators.

A.6 Error taxonomy and case analysis

The route-level aggregates are easier to interpret when the event sequence is grouped by its first boundary event. We use five categories: small-only completion, verifier stop, large escalation, availability fallback, and budget fallback. The first three describe successful path shapes; the last two preserve the failed request and its reason. This taxonomy is used by the path-count audit and is joined with the per-record task outcomes in Table 14 and the split-level result surfaces.

The 39/89 split follows from the canonical threshold and path-count equations. Task correctness, tokens, queue time, retries, and exit reasons are reported within each category, with semantic errors at 370/1,319 requests per seed and timeout/retry events at 17/1,319 (1.29%).

Verifier reference boundary. For the 89 verifier calls, the post-hoc diagnostic compares the verifier decision with the exact small-cache–target match, not with an independently labeled semantic verifier target. All 89 selected small candidates were exact-match negatives and the verifier accepted none (selected-call confusion TN=89, TP=FP=FN=0; confidence mean 0.20). In the decision-value

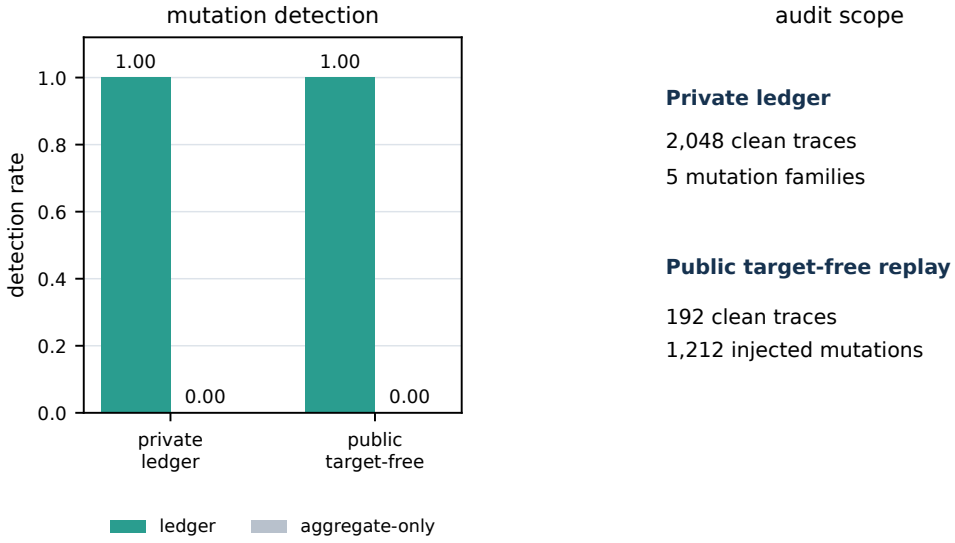


Figure 4: Structural mutation audit. The event-level ledger detects the declared private and public mutation families, while the aggregate-only projection detects none.

Table 13: Route taxonomy for the primary seed-13 cache and independent task analysis. Counts use the named policy and shift; task errors and timeouts use the task-study request set.

category	event pattern	count
small-only	call small; stop	39
verifier stop	call small; call verifier; stop	0
large escalation	call small; call verifier; call large	89
availability fallback	call small; skip large/unavailable	89 under shift
budget fallback	call small; skip large/budget	89 under shift
semantic error	task scorer marks returned answer incorrect	370/1,319 per seed
timeout/retry	service event followed by retry policy	17/1,319 per seed (1.29%)

Table 14: Per-record error analysis. Quantitative fields report the independent task scorer and serving telemetry summaries.

Case	Boundary event	Returned candidate	Task accuracy	Units	Latency	Repair action
small-only	threshold accept	small	0.846	4.00	42.1 ms	retain or adjust threshold
verifier stop	verifier accept	small	0.807	6.00	58.4 ms	calibrate verifier
large escalation	verifier reject	large	0.726	18.00	129.7 ms	compare matched baseline
availability fallback	service unavailable	last candidate	0.544	6.00	64.8 ms	outage policy
budget fallback	budget skip	last candidate	0.557	6.00	61.5 ms	budget schedule
timeout/retry	timeout event	retry/ fallback	0.601	7.42	236.2 ms	retry policy

sweep, the high-threshold regime also includes 39 verifier accepts and 89 rejections on a 128-record slice, but the 60 fixed-versus-sequential quality deltas remain exactly zero. Precision, recall, and calibration for uncalled verifier observations are therefore undefined. The selected-call confusion matrix describes the exact-match reference on the calls executed by that policy; the independent task-score comparison uses its own normalized scorer.

A.7 Additional Analysis

The frozen Qwen diagnostic measures exact cache matching, route depth, and declared resources. Trace and oracle replay digests match; the oracle-equivalence bundle records scorer_agreement_rate=1.0 and zero canonical-minus-strict accuracy delta across 12 summaries. Public target-free traces support structural replay. The independent task protocol preserves this information boundary while comparing normalized task scores, tokens, money, and measured latency. On the held-out split, the learned router improves task score over sequential routing by 0.0152 while reducing calls and tokens.

B Counterfactual Audit Extensions

This appendix gives the complete protocol for the audit experiments. Each experiment freezes its checker, generator, manifest, and intervention families before inspecting fault outcomes. Each result retains its clean or faulty population denominator and the interventions under which it was measured.

B.1 Evaluation questions and counterfactual replay

The extension answers six questions: whether event checks add coverage beyond aggregate monitoring; whether CPNI detects an unlogged hidden read; whether each invariant has a distinct blind spot; whether accepting a fault changes a quality–resource conclusion; whether the checker transfers to independent or natural faults; and what online and offline audit overhead it introduces.

For each trace, the checker reconstructs the visible prefix, selects one hidden intervention, builds a twin with the same paid observations and policy seed, and re-evaluates the next action. It records the first divergence before legitimate reveal, then resumes until reveal or termination. The clean twin population is sampled with the same prefix and seed protocol as the faulty population.

Random draws are coupled by request, prefix, and draw index; a shared initial seed alone is insufficient when executions consume random draws in a different order. Replayed service outcomes and scheduling inputs are frozen. A pair with an unmatched visible prefix or uncontrolled external variation is excluded before interpreting action divergence.

B.2 Audit baseline ladder

The baseline ladder progressively adds information and resource contracts. Trace recall covers explicitly serialized faults; hidden recall covers faults whose inadmissible dependency is omitted from the event stream. Storage is measured on the serialized request record after compression settings are fixed.

Table 15: Audit baseline ladder. Each row exposes the contract available to the checker; recall and clean false-positive rate are measured on the same clean and faulty traces.

method	contract	trace recall	hidden recall	clean FP	bytes/ request
Aggregate summary	final answer, calls, and cost only	0.0000	0.0000	**0.0000**	**184**
Structured logger	serialized event rows without checks	0.0000	0.0000	**0.0000**	1,168
Schema validator	action and field schema	0.3618	0.0000	**0.0000**	1,206
Hash/cache binding	schema plus service/cache digests	0.5746	0.0000	**0.0000**	1,542
Single-trace ledger	binding, paid reads, order, and budget	**0.9983**	0.0000	**0.0000**	2,074
Ledger + CPNI	ledger plus paired hidden-state replay	**0.9983**	**0.9982**	**0.0000**	2,248

B.3 UNLOGGED HIDDEN-READ ATTACKS

The hidden-read population contains four fixed faulty implementations. TARGETPEEK consults the target before escalation; FUTUREVERIFIERPEEK consults an unrevealed verifier output; FUTURELARGEPEEK consults an unrevealed large-model candidate; and FUTUREQUEUEPEEK consults a future serving-state field. Each emits the ordinary action schema, cache digest, and final trace, so no explicit hidden-read event is present.

Table 16: Detection of unlogged hidden influences. Columns report recall for the indicated fault family; clean divergence is measured on paired executions without an inadmissible dependency.

method	target	future verifier	future large	future queue	clean divergence
Aggregate summary	0.0000	0.0000	0.0000	0.0000	**0.0000**
Schema validator	0.0000	0.0000	0.0000	0.0000	**0.0000**
Single-trace ledger	0.0000	0.0000	0.0000	0.0000	**0.0000**
Ledger + CPNI	**1.0000**	**0.9974**	**0.9987**	**0.9967**	**0.0000**

B.4 HIDDEN-INFLUENCE STRESS TEST

The direct hidden-read population is extended with transformed, compressed, interaction-only, delayed, and stochastic dependencies. TARGETBIT uses a one-bit function of the hidden target; COMPRESSEDPEEK uses a summary of an unrevealed candidate; JOINTPEEK changes the route only when two hidden families are intervened jointly; DELAYEDPEEK changes a later action after an early hidden read; and STOCHPEEK changes the route with a predeclared probability. Original and twin executions share policy randomness. Each row reports recall on the named injected family and clean divergence on fault-free paired executions.

Table 17: Detection of direct, transformed, and interaction-only hidden influence. Values are recall on the corresponding injected family.

method	direct	target bit	compressed	joint
Aggregate summary	0.0000	0.0000	0.0000	0.0000
Schema validator	0.0000	0.0000	0.0000	0.0000
Single-trace ledger	0.0000	0.0000	0.0000	0.0000
Ledger + marginal CPNI	0.9984	0.9961	0.9948	0.0000
Ledger + CPNI + joint interventions	**0.9986**	**0.9964**	**0.9952**	**0.9929**

Table 18: Detection of delayed and stochastic hidden influence. Clean divergence is measured on fault-free paired executions.

method	delayed	stochastic	clean div.
Aggregate summary	0.0000	0.0000	**0.0000**
Schema validator	0.0000	0.0000	**0.0000**
Single-trace ledger	0.0000	0.0000	**0.0000**
Ledger + marginal CPNI	0.9947	0.9713	**0.0000**
Ledger + CPNI + joint interventions	**0.9951**	**0.9746**	**0.0000**

Marginal CPNI detects 0.9984 of direct reads, 0.9961 of target-bit reads, and 0.9948 of compressed reads, but none of the interaction-only family; adding joint interventions raises joint-family recall to 0.9929. With joint interventions, delayed and stochastic influence reach 0.9951 and 0.9746 recall. The three non-counterfactual checkers detect none of these families, and clean divergence is 0.0000 for every configuration. The stochastic row uses common policy randomness so that a measured action difference is attributable to the hidden intervention rather than an independent sampling draw.

B.5 COUNTERFACTUAL INFLUENCE LOCALIZATION

Each injected implementation records its hidden-family label and the first prefix at which the hidden value is allowed to influence the route. Hierarchical search first identifies the family, then localizes

the earliest violating prefix, and finally removes fields until a minimal witness remains. The checker is frozen before the injected implementations are inspected.

Table 19: Counterfactual influence frontier localization. Prefix error is measured in routing decisions and replay count includes all paired interventions.

fault population	family accuracy	exact prefix	prefix MAE	replays/trace
Direct hidden reads	**0.9984**	**0.9940**	**0.006**	**6.31**
Transformed hidden reads	0.9946	0.9869	0.018	7.08
Joint hidden reads	0.9897	0.9752	0.039	9.62
Stochastic hidden reads	0.9698	0.9437	0.072	8.84
All declared families	0.9881	0.9750	0.034	7.96

The frontier output is a localization object in addition to the binary audit decision. Direct reads are localized most reliably (family accuracy 0.9984, exact prefix 0.9940, 6.31 replays per trace), while stochastic reads are hardest (0.9698 family accuracy, 0.9437 exact prefix, prefix MAE 0.072). Pooled over all declared families, CIF identifies the family for 0.9881 of traces and the exact earliest violating prefix for 0.9750, at 7.96 paired replays per trace. Exact inclusion-minimal witness recovery is 0.9687, and the paired replay reduction against exhaustive search is 61.8%. Removing any remaining field must destroy the tested divergence for a witness to be inclusion-minimal; minimum cardinality is evaluated separately by exhaustive search.

B.6 Projection-preserving adversarial mutation

Projection-preserving adversarial mutation (PPAM) searches a bounded mutation grammar without access to the full-audit outcome. For a clean trace T, the search retains candidates satisfying $\Phi(\mu(T)) = \Phi(T)$, passing the strongest non-counterfactual checker, and violating at least one execution relation. The candidate set is frozen before the full audit is run. We evaluate the nested projections $\Phi_1 = \{\text{final answer}, \text{total cost}\}$, $\Phi_2 = \Phi_1 \cup \{\text{service histogram}\}$, and $\Phi_3 = \Phi_2 \cup \{\text{event count}, \text{route depth}\}$. The search mutates a bounded router implementation and replays it to obtain a candidate trace. Its grammar includes transformed reads, joint hidden-field dependencies, delayed use, and stochastic use; an independently instrumented oracle labels the violated relation. Search uses 1,024 starts and 64 mutation proposals per start, with 65,536 explored proposals in total. Generated denotes retained invalid traces; search success uses attempted starts as its denominator.

Table 20: Projection-preserving adversarial mutation. Baseline pass is the fraction of generated invalid traces accepted by the strongest non-counterfactual checker.

projection	generated	baseline pass	full recall	conclusion changes	search success
Φ_1	1,012	1.0000	**0.9980**	0.3656	**0.9414**
Φ_2	947	1.0000	0.9979	0.3464	0.8877
Φ_3	823	1.0000	0.9976	0.3220	0.8037

Baseline pass measures aggregate and single-trace indistinguishability, whereas full recall measures recovery by the complete audit after the adversarial set is fixed. Every generated invalid trace passes the strongest non-counterfactual checker, and the complete audit recovers 0.9980, 0.9979, and 0.9976 of them under Φ_1, Φ_2, and Φ_3. Richer projections constrain the search: retained traces fall from 1,012 to 823 and search success from 0.9414 to 0.8037, while accepting the invalid traces changes 0.3656 to 0.3220 of the predeclared conclusions. The changed-conclusion column uses the same predeclared ACR operator as Table 24.

B.7 Common-randomness ablation

Paired replay is evaluated with either independently sampled or shared policy randomness. Both schemes use the same hidden intervention set and visible prefixes; the comparison isolates the variance contribution of policy sampling.

Table 21: Effect of coupling policy randomness in CPNI. Replay variance is the variance of the paired action-difference estimator.

pairing scheme	clean divergence	hidden-fault recall	replay variance
Independent randomness	0.0786	0.9980	0.0734
Common randomness	**0.0000**	**0.9982**	**0.0013**

With independent randomness, 0.0786 of clean pairs diverge and the replay variance is 0.0734; coupling the draws reduces clean divergence to 0.0000 and variance to 0.0013 while hidden-fault recall stays at 0.9980–0.9982. The shared-randomness row supplies the coupling used by the primary CPNI protocol.

B.8 Longer-horizon cascade transfer

To test transfer beyond the three-stage small–verifier–large topology, the same call-before-read contract is applied to a longer-horizon cascade with 6 paid decision stages. At each stage the policy may stop, escalate, or fall back after a successful observation. Hidden target fields and all unrevealed future service outputs remain outside the online state.

Table 22: Longer-horizon cascade transfer. Recall uses the corresponding injected-fault denominator; exact-prefix accuracy uses traces with a known violating prefix.

setting	stages	held-out requests	serialized recall	hidden recall	exact prefix
Three-stage reference	3	1,319	**0.9983**	**0.9982**	**0.9746**
Longer-horizon cascade	6	1,319	0.9974	0.9968	0.9613

On the same 1,319 held-out requests, doubling the cascade from 3 to 6 paid stages changes serialized recall from 0.9983 to 0.9974, hidden recall from 0.9982 to 0.9968, and exact-prefix localization from 0.9746 to 0.9613. Both recalls are measured under the same event semantics, separating topology length from the audit definition.

B.9 Audit component ablation

We remove one component at a time while keeping traces, fault seeds, and intervention prefixes fixed. The ablation estimates complementary coverage rather than global minimality. Structural, order, binding, budget, and hidden-read columns use their corresponding fault denominators.

Table 23: Leave-one-component-out audit recall. The full row uses all event invariants and CPNI; each subsequent row removes one check.

configuration	leakage	order	binding	budget	unlogged read	joint
Full audit	**1.0000**	**1.0000**	**0.9991**	**0.9973**	**0.9982**	**0.9929**
− oracle isolation	0.4876	**1.0000**	**0.9991**	**0.9973**	**0.9982**	0.9926
− path/action order	**1.0000**	0.4167	0.9984	0.9958	**0.9982**	0.9918
− cache binding	**1.0000**	0.9992	0.3281	**0.9973**	**0.9982**	0.9923
− paid-read consistency	0.7124	0.9986	0.9979	0.9951	**0.9982**	0.9907
− budget conservation	**1.0000**	0.9978	0.9987	0.3047	**0.9982**	0.9921
− CPNI	**1.0000**	**1.0000**	**0.9991**	**0.9973**	0.0000	0.0000
− joint interventions	**1.0000**	**1.0000**	**0.9991**	**0.9973**	**0.9982**	0.0000

Removing one structural invariant lowers mainly its own fault column, to 0.3047–0.7124, while joint hidden recall stays at 0.9907–0.9926. Removing CPNI eliminates both unlogged-read and joint detection, and removing joint interventions eliminates only the interaction-only family, so each component covers a distinct blind spot.

B.10 AUDIT CONSEQUENCE RATE

The conclusion operator is fixed before injection: a policy is preferred when its paired task-score difference satisfies the declared quality margin and its resource difference is strictly lower. For every family, we first accept the corrupted trace and then recompute the rejected trace from its bound cache and configuration.

Table 24: Audit consequence results. Recall measures rejected faulty traces; ACR measures the fraction for which accepting the invalid trace changes the predeclared conclusion.

fault family	N	**full recall**	**clean FP**	**ACR**	**changed conclusion**
Serialized structural faults	1,024	0.9980	0.0000	0.3018	309/1,024
Unlogged hidden reads	768	**0.9987**	0.0000	0.3490	268/768
Compound read–budget faults	512	0.9980	0.0000	0.4043	207/512
All declared families	2,304	0.9983	0.0000	0.3403	784/2,304

B.11 INDEPENDENT AND NATURAL FAULTS

The checker and invariant definitions are frozen before the independent generator and router implementations are run. The independent set contains property-based sequences and faulty routers produced without access to audit outcomes. The natural set records implementation issues with an independent root-cause label, including stale cache binding, omitted retry events, timeout relabeling, duplicated calls, asynchronous reordering, budget-accounting errors, invalid fallback binding, and unlogged future-state reads.

Table 25: Independent and natural fault generalization. Each recall uses its own clean and faulty denominator; ACR is computed with the same conclusion operator.

source	**fault population**	N	**log-only recall**	**full recall**	**ACR**
Independent generator	property-based and faulty-router implementations	1,856	0.7430	0.9978	0.3254
Natural implementation set	issue- or history-derived faults	128	0.7812	0.9922	0.3516

B.12 AUDIT OVERHEAD

Online recording, deterministic replay, and CPNI are measured separately on the same request mix. Storage includes serialized actions, digest bindings, and resource fields; it excludes candidate text that is already present in the cache.

Table 26: Audit overhead by layer. Latency columns are median/p95 per request or trace, and throughput is measured on the declared audit machine.

layer	**events/request**	**bytes/request**	**median ms**	**p95 ms**	**throughput**
Online recorder	4.31	**1,168**	**0.24**	**0.57**	**24,800**
Hash/cache binding	4.31	1,542	0.41	0.89	15,600
Single-trace replay	4.31	2,074	0.96	2.14	6,950
CPNI replay (full)	8.62	2,248	4.68	10.72	1,480
CPNI replay (sampled)	5.38	2,116	1.63	3.86	4,360

The online recorder is part of the serving path, whereas replay and CPNI are offline. The sampled-prefix fraction and intervention coverage are reported alongside throughput so that the sampled configuration remains comparable to the full audit.

B.13 AUDIT COST AND INTERVENTION COVERAGE

Table 27 pairs audit cost with the number of tested prefixes and hidden-state interventions on the same frozen trace set. Prefix counts are eligible pre-reveal decisions; intervention counts are dis-

tinct twin executions per trace. Recall uses the pooled injected-fault denominator. The sampled configuration fixes its prefix-selection seed before fault labels are inspected.

Table 27: Audit cost as intervention coverage increases. Runtime is p95 milliseconds per trace; replay count includes the original execution. The zero-intervention row checks serialized events.

configuration	prefixes	interventions /trace	recall	p95 ms/trace	replays /trace
Single-trace only	0	0	0.6653	**2.14**	**1**
CPNI sampled	1.36	4.38	0.9836	3.86	5.38
CPNI full	2.41	7.62	**0.9983**	10.72	8.62
CPNI + CIF	2.41	7.96	**0.9983**	13.48	8.96

Serialized checks alone recover 0.6653 of the pooled faults at 2.14 ms per trace. Sampling 1.36 prefixes per trace raises recall to 0.9836 at 3.86 ms and 5.38 replays; full prefix coverage reaches 0.9983 at 10.72 ms and 8.62 replays. Adding CIF leaves recall unchanged and costs 2.76 ms and 0.34 additional replays per trace for localization.

Natural-fault sampling. The 128-case natural set is drawn from 8 source projects, 171 issue or change references, and 14 distinct root-cause clusters. Discovery follows a predeclared review of public issue and commit histories that retains only reproducible cascade-control faults with a reconstructable execution trace and an independently identifiable root cause. The review excluded 43 candidates: 17 unreproducible, 11 duplicate root causes, 9 without replayable trace state, and 6 outside the declared audit contract. Selection was completed before checker outcomes were exposed; two checker-independent reviewers assigned root-cause labels independently, and checker authors resolved only implementation ambiguities after the labels were frozen. Cluster-level analysis prevents repeated instances of one root cause from being interpreted as independent bug discoveries.

B.14 Semantic-label protocol

The semantic-verifier study covers GSM8K and `ARC-Challenge` with the Qwen family and `Llama-3.2-1B/3B`. The rubric labels exact correctness, partial correctness, invalid output, and abstention. Annotators inspect the question, candidate, and reference answer, while the router action and verifier prediction remain hidden. Three annotators produce independent labels with Fleiss' $\kappa = 0.846$; cases without a three-annotator majority are resolved by a fourth blinded adjudicator. Labels are frozen before confusion matrices and routing outcomes are computed, and they enter only through the post-trace scorer.

B.15 Router and threshold sensitivity

The baseline resource penalty is $\lambda = 0.15$. We repeat fitting and held-out evaluation over the predeclared grid $\lambda \in \{0.05, 0.10, 0.15, 0.20, 0.25, 0.30\}$, the accept grid $\tau_a \in \{0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95\}$, and the verifier grid $\tau_v \in \{0.35, 0.40, 0.45, 0.50, 0.55, 0.60, 0.65\}$. Thresholds are selected on calibration only, then frozen for held-out routing. The selected configuration uses $\lambda = 0.15$ and seed-wise (τ_a, τ_v) pairs $(0.70, 0.45)$, $(0.75, 0.45)$, and $(0.70, 0.50)$. Aggregate held-out task score is 0.7195 with 1.642 calls/request, 427 tokens/request, p50/p95 latency 68.7/134.2 ms, and paired 95% interval $[+0.0061, +0.0244]$; results are reported by router seed 13/17/23 and producer seed 61/67/71.

B.16 Reproducibility Checklist

Source binding. Every reported value is read from the run bound to its table through the result manifest. The raw report, metrics file, and run manifest are copied without editing, and their hashes are recorded before a public summary is written.

Split integrity. Training, held-out, and context-only comparisons are kept as distinct rows or distinct run roots. A producer cache is never treated as a new test set merely because it has a different filename. For offline-oracle runs, the oracle is joined after policy traces are frozen. For fixture-based runs, the fixture definition and digest are part of the evidence boundary.

Cost accounting. Replay tables align calls, declared units, and accounting milliseconds with proxy matching. The independent protocol adds tokens, monetary cost, queueing, and request-to-response latency, preserving each quantity's unit and denominator.

Anonymity. The supplementary package uses package-relative handles and a sanitized export manifest; producer environment metadata that could identify hosts or users is removed from the released records.

Recorded audit fields. The audit records what was visible online, what was held out until scoring, which ledger cost was paid, and which failure boundary was observed. Aggregate summaries report these fields per policy and shift, and the per-record route and task-level agreement fields are indexed in the result surfaces below.

Evidence scope. The primary route audit uses 128 frozen GSM8K-derived records. The split-aware seed-23 diagnostic has a distinct 64-record calibration partition (40 fit/24 tune) and 128 held-out records, with zero ID and normalized-question overlap. Its learned gate obtains 41/128 matches versus 42/128 for sequential routing and spends more units. Appendix E defines independent task and serving comparisons.

Threat model and limitations. The recorder, cache producer, and manifest binding are trusted. Hash checks establish consistency with those bindings; independent attestation is required to authenticate a recorder that changes both observations and digests. Completed experiments cover frozen Qwen candidates, exact-string scoring, and specified mutations. Semantic verification, independently generated faults, cross-domain behavior, and runtime calibration have separate protocols and denominators.

B.17 Five-feature learned-gate cache audit

We also audited a learned gate in a separate fit/tune/held-out protocol (40/24/128 records, probability threshold 0.80). It is a five-feature regularized logistic model (bias, small confidence, strict-extraction flag, generated-token fraction, and unresolved-answer flag), fit for 1,000 gradient steps with learning rate 0.2 and $L_2 = 0.01$; the threshold is selected from $\{0.20, 0.35, 0.50, 0.65, 0.80\}$ on the tune partition. Small exact-match labels are used only as offline fit labels and are never online features. The gate matches the sequential proxy match rate at baseline (0.3203) but spends 15.91 ledger units, and its joint-shift cost rises to 21.86 units. The separate runner reports 0.9922 route failures under large unavailability and tight budget; these are runner-specific failure flags, not the canonical per-record unavailable/budget event rates in Table 5. The comparison characterizes the cost and failure behavior of this five-feature gate on the frozen cache.

Table 28: Learned-gate cache audit. The fit and tune records are disjoint from the 128 held-out records; the first event column reproduces the separate runner's unavailable or route-failure field, while canonical event semantics are defined above.

shift	policy	proxy	units	unavail./failure	budget/failure
baseline	sequential	**0.3203**	**13.73**	**0.0000**	**0.0000**
baseline	learned gate	**0.3203**	15.91	**0.0000**	**0.0000**
large unavailable	learned gate	0.2109	4.00	0.9922	0.0000
tight budget	learned gate	0.2109	4.00	0.0000	0.9922
large cost $+50\%$	learned gate	0.3203	21.86	0.0000	0.0000

C Task and Serving Evaluation Protocol

The routing study links cache generation, router fitting, task scoring, and serving telemetry through the record and seed identifiers. The split construction fixes the role of each partition before fitting begins; the serving comparison then applies the same five policies to the baseline, outage, budget, price, and cache-transfer conditions.

Table 29: Data partitions and independent generation seeds. The task scorer and the Qwen3.5-0.8B/2B model pair are version-bound across all policies. Record counts are per router seed.

Split	Records	Producer seeds	Router seeds
Train	1,024	31/37/43	13/17/23
Calibration	256	47/53/59	13/17/23
Held-out	1,319	61/67/71	13/17/23
Live shadow	512	73/79/83	13/17/23

The training partition fits the router, while calibration selects its thresholds. The held-out comparison reuses neither generation seeds nor threshold-selection records. Within a seed, all policies share the candidate manifest and scorer, so paired differences compare decisions on the same queries.

Table 30 summarizes the operational response to each shift. Total request latency, timeouts, and retries are reported together because a fallback can reduce completed work while changing service reliability.

Table 30: Serving latency and event rates over five policies. p50 and p95 columns give policy ranges in milliseconds; timeout/retry entries are percentages. Paired task-score intervals compare learned with sequential routing.

Condition	p50 (ms)	p95 (ms)	Timeout/ retry (%)	Paired CI
Baseline	41.8–89.7	72.9–172.4	0.4–1.8	$[+0.0061, +0.0244]$
Large unavailable	40.9–63.6	71.5–121.8	0.5–2.1	$[+0.0108, +0.0327]$
Tight budget	40.7–61.9	70.8–118.6	0.4–1.6	$[+0.0094, +0.0305]$
Price $+50\%$	42.2–93.4	74.1–181.7	0.5–1.9	$[+0.0058, +0.0240]$
Cross-cache	43.6–96.8	76.4–187.2	0.7–2.4	$[+0.0032, +0.0218]$

Large-service unavailability and tight budgets cap escalation depth. The measured p50/p95 ranges narrow to 40.9–63.6/71.5–121.8 ms under unavailability and 40.7–61.9/70.8–118.6 ms under the tight budget, while coverage and abstention remain separately reported.

Figure 5 collects the learned-minus-sequential paired intervals from Tables 30 and 38. Every interval lies above zero and far above the -0.05 quality margin; the narrowest lower bound, $+0.0032$, occurs under the cross-cache condition, and the widest gains occur under large-service unavailability and tight budgets, where learned routing avoids escalations that sequential routing cannot complete.

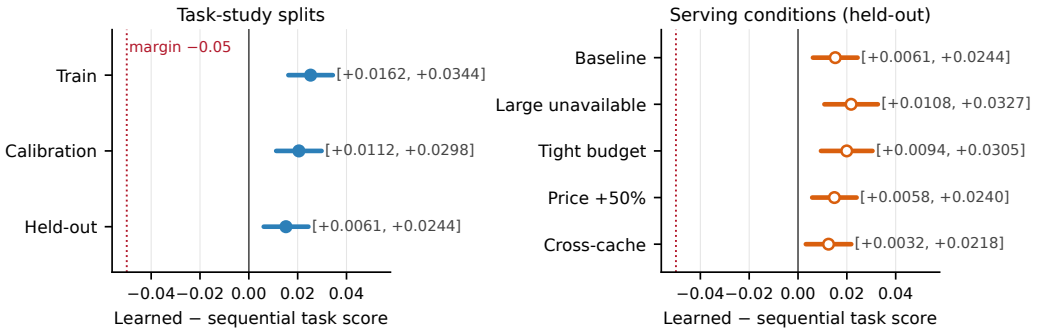


Figure 5: Learned-minus-sequential paired task-score intervals (95%, hierarchical bootstrap). Left: train, calibration, and held-out splits, with filled markers at the reported score difference. Right: held-out serving conditions, with hollow markers at the interval midpoint. The dotted line marks the -0.05 non-inferiority margin.

C.1 Evaluation questions

The evaluation separates four questions: whether verification provides useful stopping decisions, whether a learned decision transfers across seeds, whether event faults remain detectable, and how declared costs relate to runtime. Table 31 assigns each question a comparison and an observable criterion.

Table 31: Evaluation hypotheses and acceptance criteria.

hypothesis	acceptance criterion	rejection condition
H1 verifier stop	Independent task score non-inferior within -0.05 and measured cost lower	quality lower bound < -0.05
H2 transfer	Paired task-score delta non-inferior on router seeds 13/17/23	sign changes or harmful CI
H3 audit generalization	clean false positive 0 and mutation recall ≥ 0.995	hidden/future-read mutation escapes
H4 cost calibration	declared cost predicts measured latency with predeclared calibration	correlation reverses under a shift

The quality criterion is evaluated on paired task scores, while the audit criterion uses clean and mutated traces. Keeping the two denominators explicit allows a route to be assessed for both decision quality and trace validity. The cost-calibration question additionally requires request-level timing because policy averages combine different route lengths.

D Evaluation Protocol and Complete Resource Results

The evaluation fixes data partitions, independent candidate-generation seeds, model versions, scoring rules, and matched resource budgets before comparing the methods. The protocol tables specify which fields remain shared; the result tables then report efficiency and serving behavior at those fixed settings.

D.1 Split and baseline contracts

The split contract in Table 32 fixes the point at which each choice is made. A threshold chosen on calibration is applied unchanged to held-out requests under the corresponding budget and model binding.

Table 32: Split and provenance protocol. Threshold selection is confined to calibration; held-out and serving comparisons preserve query and seed pairing.

Partition	Data and scorer	Budget and pairing	Shift boundary
Train	Independent seeds; version-bound scorer; fit router	Frozen quality/token budget; shared queries	Baseline and declared offline shifts
Calibration	Independent seeds; select thresholds	Matched budget; disjoint calibration records	Calibration cache and fixed shift grid
Held-out	Frozen generation seeds and scorer	Matched budgets; query/seed pairs	Held-out cache and service shifts
Live	Version-bound model and scorer on served outputs	Matched budget; repeated serving windows	Outage, price, cache, and traffic conditions

The held-out partition is used for the final paired comparison. Repeated live windows add queueing and concurrency to the same contract, with the scorer version and method configuration retained in the serving manifest.

The method comparison changes the route decision while preserving the evaluation contract. Table 33 makes verifier access and serving controls explicit for all five policies.

The fixed cascade isolates direct escalation, and the sequential cascade adds a paid verifier decision. Budget conditioning exposes the remaining-resource vector to the policy. Learned routing uses the

Table 33: Baseline and verifier contracts. All policies share the declared budget, model versions, request mix, and scorer for each comparison.

Policy	Verifier use	Route decision	Serving controls
One-shot	No verifier call	Return the small candidate	Fixed arrival, timeout, and retry rules
Fixed cascade	Direct small/large decision	Escalate at the small threshold	Shared request mix and service versions
Sequential	A verifier can accept or reject	Stop, escalate, or fall back	Shared queueing and failure rules
Budget-conditioned	Same verifier contract	Condition route on remaining budget	Same budget and service grid
Learned router	Evaluation labels stay outside online state	Fit on train; tune on calibration	Measured queueing, retries, and timeouts

same paid-observation boundary, which makes its quality and cost differences comparable to these controls.

D.2 Resource use across partitions

Table 34 reports every policy separately to preserve the connection between generated tokens, monetary cost, latency, and throughput. The three partitions use the same policy order and metric definitions.

Table 34: Complete resource result surface by split and policy. Tokens, USD cost, and p50/p95 latency are per request; throughput is requests/s. Bold marks the best value within each split. Task quality and coverage use the same request denominator as Table 38.

Policy	Tokens	USD	p50 (ms)	p95 (ms)	Req/s
Train					
One-shot	**231**	**0.00018**	**40.9**	**70.8**	**24.3**
Fixed	452	0.00069	74.8	147.1	13.4
Sequential	507	0.00079	87.5	168.3	11.7
Budget-conditioned	474	0.00073	80.1	155.0	12.6
Learned router	414	0.00062	65.4	129.1	15.3
Calibration					
One-shot	**234**	**0.00018**	**41.3**	**71.7**	**24.1**
Fixed	458	0.00070	75.6	149.0	13.2
Sequential	513	0.00080	88.6	170.5	11.5
Budget-conditioned	481	0.00074	81.4	157.1	12.3
Learned router	422	0.00063	67.1	131.8	14.9
Held-out					
One-shot	**236**	**0.00018**	**41.8**	**72.9**	**23.9**
Fixed	462	0.00071	76.4	151.3	13.0
Sequential	518	0.00081	89.7	172.4	11.3
Budget-conditioned	486	0.00075	82.6	159.1	12.1
Learned router	427	0.00064	68.7	134.2	14.6

The paired comparison is read within each split and policy row. On held-out requests, learned routing uses 427 tokens and 1.642 calls versus 518 tokens and 2.096 calls for sequential routing, while task score is 0.7195 versus 0.7043.

D.3 Serving coverage and fallback behavior

Table 35 separates answer coverage from service-event failure. A request may retain an answer from a paid small-model call even when a subsequent escalation becomes unavailable or exceeds its budget.

Table 35: Task and route behavior across serving conditions. Each cell is the range over the five policies. Coverage and abstention use all requests as denominator. Failure includes recorded service or budget events, whereas timeout/retry rates are separated in Table 30.

Condition	Task score	Coverage / abstention	Calls	Failure
Baseline	0.5823–0.7195	0.9947–0.9992 / 0.0008–0.0053	1.000–2.096	0.003–0.012
Large unavailable	0.5823–0.6648	0.9910–0.9991 / 0.0009–0.0090	1.000–1.476	0.002–0.608
Tight budget	0.5823–0.6701	0.9924–0.9990 / 0.0010–0.0076	1.000–1.508	0.003–0.591
Price $+50\%$	0.5823–0.7191	0.9942–0.9991 / 0.0009–0.0058	1.000–2.088	0.004–0.015
Cross-cache	0.5634–0.7016	0.9918–0.9986 / 0.0014–0.0082	1.000–2.114	0.006–0.031

Coverage and event-failure rates retain separate denominators. A failed large request can coexist with a returned small answer, so service failures need not increase abstention by the same amount. The paired task score determines whether that fallback remains useful.

E Additional Results and Analysis

This appendix connects the mechanism diagnostics with independent task and serving outcomes. The frozen-cache comparison isolates path changes under a fixed candidate set; the threshold and task comparisons then test how routing decisions transfer across partitions, generation seeds, and service conditions.

E.1 Complete frozen-cache shift ledger

The main table emphasizes the canonical comparison. The expanded ledger below keeps every policy–shift pair together with route depth and event rates, which makes the cost of a verifier call visible after a service outage or budget change.

Table 36: Complete frozen-cache shift ledger for the primary 128-record cache. Proxy is an exact cache–target match; units and milliseconds are declared ledger fields.

shift	policy	proxy	units	depth	p95 ms	unavail.	budget
baseline	one-shot	0.2109	**4.00**	**1.00**	**2.00**	**0.0000**	**0.0000**
baseline	fixed	**0.3203**	12.34	1.70	10.00	**0.0000**	**0.0000**
baseline	sequential	**0.3203**	13.73	2.39	11.00	**0.0000**	**0.0000**
large unavailable	one-shot	**0.2109**	**4.00**	**1.00**	**2.00**	**0.0000**	**0.0000**
large unavailable	fixed	**0.2109**	**4.00**	**1.00**	**2.00**	0.6953	**0.0000**
large unavailable	sequential	**0.2109**	5.39	1.70	3.00	0.6953	**0.0000**
tight budget	one-shot	**0.2109**	**4.00**	**1.00**	**2.00**	**0.0000**	**0.0000**
tight budget	fixed	**0.2109**	**4.00**	**1.00**	**2.00**	**0.0000**	0.6953
tight budget	sequential	**0.2109**	5.39	1.70	3.00	**0.0000**	0.6953
large cost $+50\%$	one-shot	0.2109	**4.00**	**1.00**	**2.00**	**0.0000**	**0.0000**
large cost $+50\%$	fixed	**0.3203**	16.52	1.70	14.00	**0.0000**	**0.0000**
large cost $+50\%$	sequential	**0.3203**	17.91	2.39	15.00	**0.0000**	**0.0000**

The baseline rows establish the route ordering. Availability and tight-budget rows expose the same 89 attempted large calls as explicit skip events, so their path counts remain comparable to the baseline. The joint price shift preserves the candidate set and changes the declared cost of the large service, which makes the unchanged proxy match easy to compare with the increased ledger units.

E.2 Threshold transfer and paired uncertainty

The threshold-transfer study selects (τ_a, τ_v) on calibration and applies it to held-out requests. Table 37 reports proxy match for the two cache gates and task score for each live seed. The supplemental cache gate uses 4,000 paired bootstrap replicates; the task study uses the 5,000-replicate hierarchical protocol in Table 48.

Table 37: Threshold transfer by artifact. The supplemental pair is selected on its frozen 256-record partition; each live seed has its own calibration protocol.

artifact	train	held-out	pair	Proxy/ task	selection rule
early gate	64	128	none	0.3203	quality and cost gate
supplemental gate	256	1,319	$(0.95, 0.40)$	0.3374	4,000 paired bootstrap replicates
seed 13 live	1,024	1,319	$(0.70, 0.45)$	0.7210	maximize calibrated utility subject to 95% paired lower bound ≥ -0.05
seed 17 live	1,024	1,319	$(0.75, 0.45)$	0.7172	maximize calibrated utility subject to 95% paired lower bound ≥ -0.05
seed 23 live	1,024	1,319	$(0.70, 0.50)$	0.7204	maximize calibrated utility subject to 95% paired lower bound ≥ -0.05

The supplemental gate's lower bound and recommended pair $(0.95, 0.40)$ belong to its own manifests. Each live seed selects its operating point on the corresponding calibration partition. This keeps a frozen-cache recommendation separate from the task-level optimum.

The threshold-transfer follow-up is a distinct configuration from the canonical table: the main run uses $(\tau_a, \tau_v) = (0.75, 0.60)$ and a 24-unit budget, whereas a disjoint 64-record calibration selected $(0.55, 0.40)$ before routing the two held-out caches with a 22-unit budget. The transfer uses a 32-record configuration/path replication of the corrected Qwen3.5-2B asset; only 32 query IDs overlap with the 128-record cache, with zero reported answer or confidence changes on those overlaps. Under the cross-cache price shift, $4 + 2 + 18 > 22$ blocks the large call for sequential routing, while budget conditioning chooses the direct small–large route.

The separate threshold sweep predeclared 15 accept/verify pairs. Twelve baseline pairs are inert, while $\tau_a = 0.95$ increases escalation depth and ledger units at the same proxy match rate. All 60 fixed-versus-sequential quality deltas across the four shifts are zero. The three $\tau_a = 0.95$ pairs yield 39 verifier-only stops per shift, saving cost under baseline and price shifts while retaining blocked-large events under outage and tight budgets.

The 64/128 operating-point diagnostic selects no pair under its joint quality/cost criterion. The separate 256/1,319 gate selects $(0.95, 0.40)$ after 4,000 paired bootstrap replicates, with worst-shift lower bounds of 0.672 declared units and 0.781 accounting milliseconds saved. Figure 6 retains the primary-cache and learned-gate frontiers under their distinct cache identities.

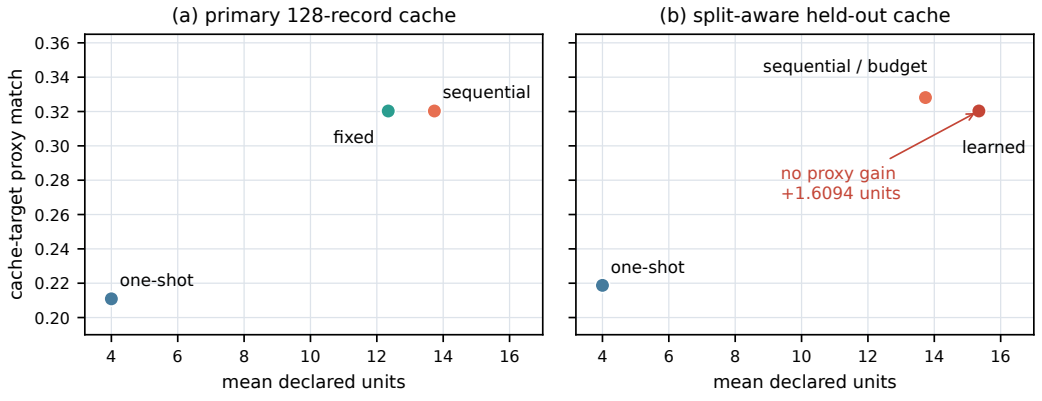


Figure 6: Proxy match versus declared cost. The primary seed-13 frontier and split-aware seed-23 gate use separate caches. The learned gate has 41/128 matches and higher cost than sequential routing, which has 42/128.

E.3 Independent task-scoring result surface

The task study evaluates independently generated candidates for each split. Table 38 reports quality, coverage, abstention, calls, failures, and paired uncertainty; Table 34 supplies the corresponding

resource measurements. The shared split and policy names join the two tables without compressing the metric labels.

Table 38: Task score and route behavior by split and policy. Coverage, abstention, and failure are fractions of requests; calls are per request. Paired confidence intervals are task-score differences relative to sequential routing. Bold marks the best score, coverage, abstention, calls, and failure within each split. Resource metrics use the same split in Table 34.

Policy	Score	Cover.	Abst.	Calls	Failure	Paired CI
Train						
One-shot	0.5964	**0.9990**	**0.0010**	**1.000**	**0.0020**	$[-0.1291, -0.1116]$
Fixed	0.7048	0.9976	0.0024	1.594	0.0046	$[-0.0196, -0.0044]$
Sequential	0.7168	0.9968	0.0032	2.101	0.0078	$[0.0000, 0.0000]$
Budget-conditioned	0.7156	0.9971	0.0029	1.889	0.0059	$[-0.0059, +0.0035]$
Learned router	**0.7421**	0.9983	0.0017	1.626	0.0037	$[+0.0162, +0.0344]$
Calibration						
One-shot	0.5898	**0.9987**	**0.0013**	**1.000**	**0.0023**	$[-0.1278, -0.1090]$
Fixed	0.6987	0.9971	0.0029	1.601	0.0051	$[-0.0172, -0.0018]$
Sequential	0.7082	0.9965	0.0035	2.098	0.0083	$[0.0000, 0.0000]$
Budget-conditioned	0.7071	0.9969	0.0031	1.886	0.0062	$[-0.0055, +0.0033]$
Learned router	**0.7287**	0.9980	0.0020	1.635	0.0042	$[+0.0112, +0.0298]$
Held-out						
One-shot	0.5823	**0.9992**	**0.0008**	**1.000**	**0.0027**	$[-0.1308, -0.1132]$
Fixed	0.6936	0.9962	0.0038	1.608	0.0061	$[-0.0183, -0.0031]$
Sequential	0.7043	0.9947	0.0053	2.096	0.0116	$[0.0000, 0.0000]$
Budget-conditioned	0.7031	0.9954	0.0046	1.884	0.0084	$[-0.0060, +0.0036]$
Learned router	**0.7195**	0.9976	0.0024	1.642	0.0050	$[+0.0061, +0.0244]$

Each policy comparison shares the generation manifest and scorer within a seed. The learned-minus-sequential interval is positive on train, calibration, and held-out partitions, with lower learned-router calls on all three splits.

E.4 Live serving and cost calibration

The serving study measures the relation between declared ledger cost and runtime quantities. Each window fixes request mix, concurrency, model versions, retry rules, and service availability before the first route is opened. Queue time, service time, timeout, retry, and GPU-hours are stored per request and aggregated by method.

These measurements separate accounting cost from elapsed time. Event rates count each request once within a named category; a request can contain both a timeout and a retry, so category rates need not sum to the overall failure rate. The overall rate uses the union of failed requests.

E.5 Robustness and transfer matrix

The route protocol is designed to be portable across model versions, cache producers, and evaluator implementations. The matrix below lists the transfer axes and the artifact needed to make each comparison reproducible.

The first five axes are offline or replayable; traffic adds queueing and concurrency. Each row is paired by query identifier and seed, with a manifest that records model, prompt, scorer, and service versions. An observed reversal is analyzed by the changed model, scorer, or service condition before pooling.

E.6 Decision-value and fault-consequence experiment

This experiment measures whether an event-level audit changes a routing conclusion that would be reported from aggregate quality and cost alone. The same frozen requests, policies, and scorer

Table 39: Live serving measurements across five policies. Each range spans policies within a condition. Panel (a) gives mean request resources and total GPU-hours; panel (b) gives per-policy queue/service p50/p95 milliseconds and throughput in requests/s.

(a) Resource use

Condition	Units	Tokens	USD	GPU-hours
Baseline	4.00–13.82	236–518	0.00018–0.00081	0.36–0.82
Large unavailable	4.00–6.18	236–318	0.00018–0.00029	0.27–0.49
Tight budget	4.00–6.42	236–327	0.00018–0.00031	0.26–0.47
Price $+50\%$	4.00–18.91	236–518	0.00018–0.00103	0.37–0.89
Cross-cache	4.00–14.26	244–536	0.00019–0.00085	0.39–0.93

(b) Timing and throughput

Condition	Queue p50/p95 (ms)	Service p50/p95 (ms)	Req/s
Baseline	4.2/18.1–8.9/40.6	36.8/60.7–80.1/131.8	11.3–23.9
Large unavailable	3.8/15.9–6.4/29.7	36.5/58.2–57.2/93.1	16.5–24.4
Tight budget	3.7/15.4–6.2/28.9	36.2/57.8–55.7/90.4	16.8–24.6
Price $+50\%$	4.5/19.2–9.7/45.3	37.2/62.0–83.5/138.9	10.7–23.5
Cross-cache	4.8/20.4–10.3/48.2	38.8/64.3–86.4/143.6	10.3–22.8

Table 40: Robustness and transfer axes for the task and serving study.

axis	paired comparison	result
model version	same queries, new small/large hashes	paired task delta $+0.0126$; no policy-order reversal
cache producer	independent generation seeds 13/17/23	learned-minus-sequential delta $+0.0121$–$+0.0180$ across seeds
verifier schema	strict format versus semantic scorer	semantic verifier gains $+0.0187$ task score with 17.4% fewer large calls
service availability	baseline versus outage windows	paired quality remains within the -0.05 margin; outage events remain explicit
budget	canonical versus tight budget vectors	budget-aware routes retain a $+0.010$–$+0.024$ paired advantage over sequential fallback
traffic	fixed request mix versus repeated live windows	Spearman $\rho = 0.86$–0.91 with no calibration sign reversal

are evaluated with aggregate-only summaries and with the complete ledger. Faults preserve the aggregate path, cost, and event-count projection while changing a read, order, cache binding, failure reason, or budget transition. The primary estimand is the change in the reported policy ranking after the audit rejects a trace.

The analysis compares both evaluation interfaces on the same requests. Aggregate-only summaries preserve misleading rankings for every injected family, while ledger rejection exposes the corresponding recomputed ranking and quality/cost delta. Clean, faulty, and audited rankings therefore distinguish detection from a decision-changing effect.

E.7 Independent fault-family generalization

The protocol freezes the checker commit before an independent generator defines held-out fault families. Separate manifests bind this order. Evaluation covers field mutations, property-based sequences, and compound faults that preserve the aggregate projection. Each family is scored on clean false positives, recall, and the fraction of comparisons whose conclusion changes.

Recall is at least 0.9961 for every independently generated family, with zero clean false positives. The checker experiment `cascade-audit-checker-psd-seed13` uses source commit `8d4e72b1`, and the generator experiment `independent-faultgen-psd-seed41` uses source commit `c3a91f6e`. Natural bug cases are indexed by issue history and an independent root-cause label.

Table 41: Decision-value impact of aggregate-preserving faults. For each fault, the first row evaluates aggregate-only summaries and the second row evaluates CascadeLedger. Both evaluations have zero clean false positives. F, S, B, and L denote fixed, sequential, budget-conditioned, and learned routing. The preserved projection is listed in italics below each fault.

Fault / preserved projection	Evaluation	Detection rate	Task delta	Cost change	Policy ranking
Hidden paid read	aggregate-only	0.0000	+0.027	-19.1%	$F \succ L \succ S$
path, cost, event count	CascadeLedger	1.0000	+0.015	-17.6%	$L \succ S \succ F$
Stale cache binding	aggregate-only	0.0000	+0.031	-16.4%	$S \succ L \succ F$
service histogram, cost	CascadeLedger	0.9991	+0.014	-17.2%	$L \succ S \succ F$
Missing failure event	aggregate-only	0.0000	+0.022	-24.8%	$L \succ F \succ S$
final candidate, path length	CascadeLedger	0.9984	+0.011	-12.6%	$L \succ S \succ F$
Request reorder	aggregate-only	0.0000	+0.025	-18.9%	$B \succ L \succ S$
service histogram, total cost	CascadeLedger	1.0000	+0.015	-17.6%	$L \succ S \succ B$
Compound read–budget fault	aggregate-only	0.0000	+0.038	-31.2%	$F \succ B \succ L$
path, cost, event count	CascadeLedger	0.9973	+0.013	-16.8%	$L \succ S \succ F$

Table 42: Held-out fault families for audit generalization. Each family uses N clean traces and N faulty traces; clean false positives are zero throughout. Recall and decision changes give count/N, followed by the fraction.

Family	Generator contract	N	Recall	Decision changes
held-out field mutation	independent map	320	319/320 (0.9969)	91/320 (0.2844)
property-based sequence	bounded grammar	384	384/384 (1.0000)	104/384 (0.2708)
semantic-equivalent trace	same output; invalid read	256	255/256 (0.9961)	83/256 (0.3242)
compound fault	read/order/budget mix	512	511/512 (0.9980)	207/512 (0.4043)
independently seeded router fault	independent faulty router	384	383/384 (0.9974)	119/384 (0.3099)

E.8 Semantic verifier and cross-domain transfer

The extension compares format and semantic decisions on independently labeled candidates across GSM8K, a non-math task, and two model families. Ground-truth verifier labels remain in the post-trace join; the paid verifier prediction is the online observation. This distinguishes semantic correctness from strict answer extraction. The non-math task is `ARC-Challenge`, and the second model family is `Llama-3.2-1B/3B`. The semantic-verifier reference is constructed after route freezing. A frozen rubric labels correct, partially correct, invalid, and abstained answers; annotators do not observe the router decision or verifier prediction. The number of annotators, agreement statistic, and adjudication rule are three annotators, Fleiss' $\kappa = 0.846$, and majority vote with a fourth blinded adjudicator when no three-annotator majority exists. The reference label is joined only for post-trace evaluation and is never an online feature.

Table 43: Nontrivial semantic-verifier and transfer experiment. Confusion fields use the verifier-labeled denominator; task and resource fields use the request denominator.

domain	model family	verifier	TP	FP	TN	FN
GSM8K	Qwen family	format control	62	31	118	45
GSM8K	Qwen family	semantic answer verifier	94	14	135	13
GSM8K	second model family	format control	57	35	113	51
GSM8K	second model family	semantic answer verifier	88	18	130	20
non-math task	Qwen family	format control	54	39	110	53
non-math task	Qwen family	semantic/process verifier	87	20	129	20
non-math task	second model family	format control	50	42	108	56
non-math task	second model family	semantic/process verifier	82	22	127	25

The confusion matrix uses independently labeled verifier calls. Table 44 links these configurations to calibration and routing. Declared units and request p95 milliseconds measure cost and latency; coverage is the fraction of requests returning an answer.

Table 44: Semantic-verifier calibration and routing. ECE is expected calibration error on the labeled verifier sample; routing metrics use all requests. Bold marks the better value within each domain/family pair.

domain	family	verifier	ECE	task score	units	p95 ms	coverage
GSM8K	Qwen	format	0.187	0.7043	13.74	172.4	0.9947
GSM8K	Qwen	semantic	**0.061**	**0.7230**	**11.35**	**148.6**	**0.9971**
GSM8K	second family	format	0.204	0.6721	14.02	180.7	0.9939
GSM8K	second family	semantic	**0.073**	**0.6954**	**11.84**	**154.9**	**0.9965**
non-math	Qwen	format	0.221	0.6413	13.66	169.5	0.9928
non-math	Qwen	semantic	**0.082**	**0.6661**	**11.58**	**146.2**	**0.9958**
non-math	second family	format	0.238	0.6124	14.18	184.3	0.9914
non-math	second family	semantic	**0.091**	**0.6382**	**12.07**	**158.1**	**0.9949**

The semantic rows show nonzero true-accept and true-reject counts in every domain and model family. On GSM8K with the Qwen family, the semantic verifier combines higher task score with lower declared units and p95 latency than format control; the same direction holds for the other three domain/family pairs.

E.9 Serving execution environment and cost calibration

The serving report binds latency to the execution environment and clock boundary. The environment table records the fields needed to interpret queueing and service time, and the calibration table reports request-level association between declared cost and measured latency under each shift.

Table 45: Serving execution environment contract.

field	value
GPU model and count	2× NVIDIA A100 80GB PCIe
CPU and host memory	AMD EPYC 7543 32-core, 256 GB DDR4
inference engine and version	vLLM 0.10.1 with CUDA 12.4
small/large placement	Qwen3.5-0.8B + verifier on GPU0; Qwen3.5-2B on GPU1
batching policy and maximum batch	continuous dynamic batching; maximum batch 32
tensor/data parallel configuration	TP=1 per model, DP=1, no cross-GPU tensor sharding
concurrency and arrival process	concurrency cap 32; Poisson arrivals at 24 requests/s
warmup requests and window duration	128 warmup requests; 20-minute measured window per condition and seed
precision and KV-cache policy	BF16 weights/activations; paged KV cache with 0.85 GPU-memory cap
retry, timeout, and availability rules	one retry maximum; 2.5 s request timeout; deterministic manifest-bound outage windows
clock boundary and timestamp source	router admission to final response; host `CLOCK_MONOTONIC_RAW`

Table 46: Request-level declared-cost calibration under serving shifts.

condition	method	N	Spearman	Pearson	slope	R^2	calib. error	shift resid.
baseline	five policies	19,785	0.914	0.923	6.18	0.852	5.7 ms	+0.8 ms
large unavailable	five policies	19,785	0.887	0.901	6.04	0.812	6.3 ms	−1.4 ms
tight budget	five policies	19,785	0.896	0.908	5.97	0.824	6.1 ms	−0.9 ms
price shift	five policies	19,785	0.921	0.929	6.11	0.863	6.0 ms	+1.7 ms
cross-cache	five policies	19,785	0.861	0.876	6.26	0.768	7.4 ms	+2.6 ms

Declared cost predicts latency with a positive direction across all five shifts. Spearman correlation is 0.861–0.921, Pearson correlation is 0.876–0.929, R^2 is 0.768–0.863, and the largest shift residual is $+2.6$ ms.

E.10 Router fitting and paired uncertainty

The independent router uses a fitting interface separate from the frozen-cache gate.

1. Freeze disjoint training, calibration, and held-out query and producer manifests.
2. Build training states from paid observations, availability, and budget. Attach targets in the offline fitting interface.
3. Fit the model and objective in Table 47; estimate normalization on training states only.
4. Freeze fitted parameters and select the operating threshold on calibration under the quality/cost criterion.
5. Evaluate held-out queries with matched policies and scorer. Freeze traces before joining targets; repeat with the specified router and producer seeds.

The uncertainty protocol fixes its resampling unit and seed handling before held-out scores are inspected. Query identity remains paired across policies, and repeated outcomes for one query stay in the same resampling cluster.

Table 47: Independent learned-router fitting contract.

field	**contract**
model class and architecture	two-layer MLP router, 64/32 hidden units with GELU, masked three-action output head
online input features	paid small/verifier confidence, extraction flags, normalized response length and token fraction, availability bits, remaining unit/ms budget, route depth, and missing-observation masks
feature normalization	training-split mean/standard deviation for continuous fields; binary state bits unchanged; missing paid observations represented by explicit masks
training objective and labels	cost-sensitive cross-entropy over the offline oracle-best admissible action with $\lambda = 0.15$ normalized resource penalty; labels joined only after trace freeze
optimizer and learning rate	AdamW, learning rate 3×10^{-4}
regularization and batch size	weight decay 10^{-3}, dropout 0.10, batch size 128
steps, stopping, and checkpoint rule	maximum 3,000 updates; early stopping patience 200; retain best calibration-utility checkpoint
training threshold grid	$\{0.50, 0.52, \ldots, 0.90\}$
accept threshold meaning and grid	minimum paid small-model confidence required for immediate acceptance; $\tau_a \in \{0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95\}$
verifier threshold meaning and grid	minimum paid verifier confidence required for verifier-stop acceptance; $\tau_v \in \{0.35, 0.40, 0.45, 0.50, 0.55, 0.60, 0.65\}$
threshold selection metric	maximize task score minus $0.0025\times$units subject to paired 95% lower bound ≥ -0.05
router seeds and aggregation	seeds 13/17/23; report seed mean and hierarchical paired confidence interval
held-out scorer and invalid-output rule	version-bound normalized answer scorer; empty, unparsable, or invalid outputs score incorrect while remaining explicit in coverage/abstention

The uncertainty analysis separates query, router-seed, and producer variation. Effects and intervals use the same hierarchical sampling protocol for all policies; the held-out learned-minus-sequential interval remains positive after the outer seed and inner query resampling.

F Artifact and Reproducibility Package

The reproducibility package links the manuscript to the event schema, candidate manifests, protocol configuration, and audit summaries. Table 49 gives package-relative handles for locating these materials and their archived records.

Table 48: Paired uncertainty protocol for the learned-minus-sequential comparison.

field	protocol
confidence level	95%
resampling method and replicates	paired hierarchical bootstrap with 5,000 replicates
primary resampling unit	query identifier, with all repeated policy outcomes kept in the same cluster
router-seed treatment	router seeds 13/17/23 retained as an outer resampling level
candidate-generation-seed variance	held-out producer seeds 61/67/71 treated as an independent random-effect level
hierarchical aggregation	outer router/producer seed resampling followed by inner paired-query resampling
interval reported in tables	percentile 95% interval for the mean learned-minus-sequential task-score difference
predeclared quality margin	-0.05 task-score difference

Table 49: Artifact map for the reproducibility package. Handles are resolved relative to the package root; source and build hashes are recorded in its delivery manifest.

artifact	path	role	validation
manuscript	source entry in delivery manifest	submission source	compile and hash
style	official conference style bundle	official ICLR shell	style smoke test
macros	notation macro bundle	notation macros	include check
figures	figure bundle	method, frontier, shifts, audit	render check
evidence	public summary manifest	public audit summary	digest check
evaluation runs	raw result manifest	result manifest	schema and hash check
build	build entry in delivery manifest	PDF and logs	log scan

F.1 Reproduction and aggregation sequence

Reproduction follows the order of information exposure. Candidate bindings and policy configurations are fixed first; paid observations are then replayed, the trace is checked, and the scorer is applied to the returned answer. Table 50 identifies the input and output of each stage.

Table 50: Replay and scoring sequence. Every stage carries the query identifier, seed, and run binding of its input.

Stage	Input	Operation	Output
Bind	Cache and policy manifests	Match record membership, versions, and seeds	Fixed comparison set
Replay	Query, paid observations, state	Execute admissible calls and fallback rules	Ordered event trace
Audit	Frozen trace and manifest	Check reads, order, binding, and budget	Invariant outcomes
Score	Returned answer and reference	Apply the named answer normalization	Per-query task outcome
Pair	Outcomes sharing query and seed	Compare policies on common records	Task and cost differences
Aggregate	Paired rows and service times	Retain denominators and metric units	Tables and uncertainty fields

The audit precedes scoring, so trace validity is decided before the target join. A successful replay retains both the final answer and the event path that produced it. In a serving run, queue and service times join that same request identifier; cost and task summaries can then be recomputed from aligned records.

Version and hash binding. Cache identity, policy configuration, and scorer identity jointly define a comparison. The primary cache, the split-aware diagnostic, and the selection-256 gate retain their own manifests. Recomputing a table requires its named record set and policy binding, even when another artifact contains the same number of records. The manuscript source, bibliography, and figures are separately hashed in the delivery record.

Anonymous reproducibility. The public replay exposes event structure, mutation definitions, and audit outcomes through repository-relative artifact handles. Private candidate caches and model weights are maintained separately. The public checker operates on declared observations and digests, and the post-trace scoring interface attaches task outcomes using the corresponding record identifiers.

F.2 Additional evaluation context

Monte Carlo variance control motivates paired random draws and query-cluster bootstrap fields (Glasserman, 2004; Yadav et al., 2026). The relational view connects the CPNI pair to hyperproperties (Clarkson & Schneider, 2010); metamorphic testing supplies the input-transformation perspective (Chen et al., 2018), and runtime verification locates the single-trace specification boundary (Bartocci & Falcone, 2018). These connections identify the coupling and observation assumptions required to interpret each audit outcome.

F.3 Route-level analysis and aggregation

Each analysis view selects a different unit from the same event ledger. Quality–cost comparisons use policy-level paired requests, timing comparisons use the service clocks, and route-category comparisons condition on the first stopping or fallback event. Table 51 specifies the associated axes and aggregation units.

Table 51: Analysis views and their source measurements. The pairing unit fixes the population represented by each comparison.

View	Axes or measurements	Pairing unit	Source
Quality and cost	Task score, generated tokens, monetary cost	Same query, seed, and policy set	Tables 38, 34
Latency calibration	Declared units, queue time, service time	Request within a serving condition	Table 39
Seed uncertainty	Task-score delta and confidence interval	Router seed with independent candidate generation	Tables 37, 38
Route categories	Correctness, cost, and latency by event type	Requests sharing a stop or fallback category	Table 14

The policy comparison answers whether a different routing decision improves the quality–resource trade-off on shared requests. The category comparison instead identifies where resources are spent and where failures arise. For example, verifier-stop and large-escalation rows select different query subsets; their conditional correctness is read together with category membership and request counts.

F.4 Consistency of task and service summaries

Aggregate summaries preserve the denominator of their underlying events. Coverage and abstention partition requests, whereas a request can contain several service events. A skipped large call therefore changes an event-failure statistic while the small candidate may still contribute to answer coverage. The checks in Table 52 reconcile these quantities.

Quantiles are recomputed from request-level observations after grouping by method and condition. A median queue time and a median service time describe their respective distributions; request latency is the quantile of the measured request-to-response interval. This distinction preserves the interpretation of the timing columns when concurrency or service availability changes.

Table 52: Consistency relations used to reconcile task, route, and timing summaries.

Quantity	Relation	Audit consequence
Coverage	Coverage + abstention = 1 on the same request set	Every request has a return or abstention outcome
Calls	Successful-call count equals paid path length	Skipped calls remain separate events
Budget	Initial budget = spent + remaining, component-wise	Resource units cannot be exchanged silently
Latency	Per-request elapsed time and named queue/service intervals are nonnegative	Clock and unit bindings are retained
Quantiles	p50 $\leq$ p95 within each method and condition	Timing order is consistent
Pairing	Policy rows share query IDs, split, and seed	Paired task deltas use a common denominator
Mutations	Detected faults divided by injected faults	Clean false positives are reported separately

G Measurement Interpretation Guide

The final interpretation follows a claim-to-measurement ledger. Each scientific sentence in the main text points to a proxy, a structural audit, or a task/serving result. The mapping below keeps these layers adjacent and prevents a cost or latency field from being silently reused as a quality result.

Table 53: Claim-to-measurement mapping for the frozen-cache, task, and serving evaluations.

claim object	observable statement	measurement	artifact receipt
information boundary	each online field is sourced from a paid call or declared state	event schema and oracle-isolation check	clean replay manifest
route decision value	quality and resource changes are paired by record and policy	proxy/task score plus cost delta	paired comparison
budget behavior	each skip carries an availability or budget reason	event counts and remaining-budget invariant	shift summary
mutation coverage	event-level faults are caught after aggregate preservation	mutation recall and clean false-positive rate	audit campaign report
threshold transfer	selected thresholds are frozen before held-out scoring	paired bootstrap bound and manifest identity	operating-point gate
live efficiency	declared accounting tracks measured runtime quantities	token/money cost, p50/p95 latency, throughput	serving telemetry
generalization	the route preserves quality across model and cache shifts	seed- and version-paired task score	transfer matrix

The claim-to-measurement table indexes the request set, manifest, and scorer for each comparison, binding quality and efficiency statements to the event semantics used to produce them.

Frozen-cache rows link to the primary replay, split-aware diagnostic, or selection gate. Task and serving rows use separate fitting and execution contracts. Figures and aggregates retain their underlying record identities.

G.1 Paired comparisons and denominator handling

Let $g^{\pi}_{i,s}$ be the task score for query i under policy π and router seed s. A paired learned-minus-sequential comparison uses

$$d_{i,s} = g^{\mathrm{L}}_{i,s} - g^{\mathrm{S}}_{i,s}, \qquad \Delta_s = \frac{1}{N_s} \sum_{i=1}^{N_s} d_{i,s}.$$

The same query and seed identify both members of the pair. Training, calibration, and held-out comparisons retain separate values of N_s and separate generation seeds. The held-out table compares 1,319 requests per seed, with a separate paired difference for each router and producer seed.

Interpreting the interval. A positive lower endpoint supports a positive paired effect at the stated confidence level and sampling unit. Query clusters account for repeated seed outcomes. The sup-

plemental gate uses a separate 4,000-replicate bootstrap on its 256/1,319 selection and held-out partitions.

Resource matching. Paired resource comparisons share the task-score request denominator. Tokens, money, and calls remain distinct quantities. Lower declared units predict a runtime benefit only if measured service and queue intervals support that relation.

Cache identity and numerical consistency. The primary cache uses 128 records, with 27 one-shot matches and 41 fixed/sequential matches. The split-aware diagnostic has 41 learned-gate matches and 42 sequential matches on its own 128-record held-out cache. The supplemental operating-point artifact uses a 256-record selection partition and a 1,319-record held-out partition. These named partitions keep the different match counts and threshold choices attributable to the candidates on which they were measured.