RAVEL: CONTROLLING HARMFUL PROPAGATION IN CROSS-AGENT SEMANTIC CACHES

Anonymous authors
Paper under double-blind review

ABSTRACT

A repository artifact can be correct when produced, stale when retrieved by another agent, and either harmless or outcome-changing in the consumer's dependency context. Cross-agent semantic caches therefore need a decision at every match: reuse the artifact, spend shared capacity to verify it, or recompute it. RAVEL implements this pre-consumption decision from prospective provenance, producer–consumer context mismatch, semantic drift, and remaining verification capacity; hard quarantine intercepts concentrated contradiction risk, while evidence-preserving refresh removes a superseded value from service without deleting its trust history. Fixed-request replay exposes why this control object matters: RAVEL admits 1,127 stale hits versus 772 for Centralized, yet records 126 versus 150 harmful events and 38 versus 45 issue-outcome flips. In a 16-agent repository simulator, RAVEL reaches $81.3\pm0.7\%$ task success at 0.523 ± 0.008 relative execution cost, compared with $80.0\pm0.8\%$ and 0.548 ± 0.009 for a controller with the same candidate, inputs, actions, quarantine rule, and cap. Across 600 held-out policy-dependent issue-runs, the gains over this logistic control are +1.8 [0.6, 3.0] success points and -0.045 [-0.059, -0.031] cost, with lower verification utilization. Document analysis, adaptive contamination, component ablations, and overlap–budget–churn sweeps support one systems principle: shared memory should govern the consequence of consumption, not freshness alone.

1 INTRODUCTION

A repository-analysis collective rarely consumes a cached artifact only once. One agent resolves a symbol, dependency, test, or document claim; another retrieves the semantic match under a different task context; later agents inherit conclusions that depend on that value. A contradiction observed after this chain has formed is no longer a local cache miss: it can redirect tool calls, exhaust a shared budget, or change the terminal issue outcome. Shared semantic memory consequently needs to control propagation before consumption while preserving the duplicated work that makes collaboration efficient.

The control point occurs after semantic matching and has three actions. REUSE is nearly free but admits an unchecked value. VERIFY obtains current evidence while spending a finite issue-level cap shared by all participating agents. RECOMPUTE replaces the value through a full tool call. Age and similarity cannot allocate these actions alone because the consequence of reuse also depends on producer history, prior consumers, contradiction lineage, the mismatch between producer and consumer dependency contexts, and the remaining verification capacity. Semantic caches primarily optimize request-level hit utility or serving cost, whereas provenance systems primarily record derivations; cross-owner admission under a coupled information budget remains implicit.

RAVEL makes that admission decision explicit. Thirty-two temporal, semantic, provenance, and producer–consumer context features describe the highest-similarity active entry, and the remaining-cap fraction represents issue-level resource state. A learned tri-action score governs ordinary hits; a hard predicate routes entries with concentrated entry- or producer-level contradiction histories to verification. Verification and recomputation refresh active state, and any discovered disagreement becomes a CONTRADICT event linked through SUPERSEDES. The serving index therefore contains the current value, while historical lineage continues to inform later trust decisions.

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.

The resulting distinction is empirical, not merely terminological. In fixed-request replay, RAVEL consumes 1,127 stale hits rather than Centralized's 772, but produces 126 rather than 150 harmful events and 38 rather than 45 issue flips: raw freshness ranks these policies differently from outcome-level reliability. In 2,500 simulator episodes, RAVEL reaches 81.3% success at 0.523 relative execution cost, versus $80.0 \pm 0.8\%$ at 0.548 ± 0.009 for multinomial logistic control under the same candidates, inputs, actions, quarantine convention, and cap. On 600 held-out issue-runs from three unseen repositories, where decisions alter later calls and budget allocation, RAVEL adds +1.8 [0.6, 3.0] success points and reduces cost by -0.045 [-0.059, -0.031].

The paper makes three connected contributions. First, it formulates harmful propagation at the consumed-artifact and issue-outcome levels, separating stale exposure from consequential reuse. Second, it instantiates budget-coupled triage with prospective lineage and an active/history transition that turns correction into future trust evidence. Third, it evaluates the full decision system through same-information controls, closed-loop execution, overhead accounting, document analysis, adaptive contamination, component removal, and overlap–budget–churn sweeps. Together these results show how a collective can retain inexpensive cross-agent reuse while directing scarce checks toward artifacts most likely to change its behavior.

2 Related Work

Semantic and serving caches. GPTCache, cost-aware LLM caching, embedding reuse, and CacheBlend decide whether cached computation is useful or how cached state should be fused (Bang, 2023; Zhu et al., 2023; Stogiannidis et al., 2023; Yao et al., 2025); IceCache manages token-state residency within long-sequence inference (Mao et al., 2026). These systems optimize a request-level hit or a serving-state placement. RAVEL instead acts after a semantic hit but before cross-owner consumption, exposes VERIFY as a budgeted information action alongside reuse and recomputation, and scores the downstream consequence of admitting the hit. The distinction is operational: a locally stale object may be safe to consume, whereas a plausible object from a mismatched producer context may redirect several agent trajectories.

Agent memory and repository interaction. AutoGen, MetaGPT, CAMEL, ChatDev, and Generative Agents coordinate agents or retain experience; AgentBench, ToolLLM, and GAIA evaluate tool use; RepoCoder and SWE-agent improve repository retrieval and agent–computer interfaces (Wu et al., 2024; Hong et al., 2024; Li et al., 2023; Qian et al., 2024; Park et al., 2023; Liu et al., 2024; Qin et al., 2024; Mialon et al., 2024; Zhang et al., 2023; Yang et al., 2024). SWE-Gym and SWE-smith provide repository-agent tasks and verifiers (Pan et al., 2025; Yang et al., 2025), while Toolformer and ReAct learn external actions (Schick et al., 2023; Yao et al., 2023). RAVEL composes with these coordination, retrieval, and tool policies at a distinct transition: whether another agent's matched artifact enters the current computation, and what trust state survives when that artifact is corrected.

Concurrent cache and memory governance. GroundedCache is the closest cache router: it validates a cached RAG answer with query, evidence-overlap, source-version, and lexical-support gates before binary reuse or regeneration (Shah, 2026). RAVEL addresses a different information structure—cross-owner repository artifacts, a three-action choice with verification distinct from recomputation, one cap coupled across agents, and contradiction lineage that changes later admissions. Governed Shared Memory supplies scoped retrieval, temporal supersession, provenance, and propagation primitives for an agent fleet, but evaluates infrastructure properties rather than learning which matched value deserves scarce verification (Margalit et al., 2026). MemTX governs write-to-belief commitment, action safety, and cascading repair through transactions (Li et al., 2026); RAVEL's locus is the read side, where an already stored semantic match may be harmlessly stale or outcome-changing for a particular consumer. MAPLE-Guard places security gates across write, retrieval, promotion, and cross-agent reuse (Xiong et al., 2026). Its target is memory-link poisoning; RAVEL's primary estimand includes ordinary repository churn, success–cost tradeoffs, selective verification, and harmless-versus-harmful stale consumption.

Consistency, provenance, and selective decisions. Distributed caches maintain or place exact objects, while database and dataflow provenance records derivations (Karger et al., 1997; DeCandia et al., 2007; Nishtala et al., 2013; Buneman et al., 2001; Cheney et al., 2009; Interlandi et al., 2015). RAVEL uses lineage prospectively rather than only for retrospective explanation: contradiction summaries enter the next admission decision, while a supersession edge separates historical trust evidence

from the active serving value. Contextual-bandit and doubly robust methods address selective feedback (Langford & Zhang, 2007; Agarwal et al., 2014; Dudík et al., 2014; Jiang & Li, 2016); here they fit a gate whose actions also mutate cache state and deplete a shared cap. Byzantine and poisoning work motivates the stable-identity and trusted-verifier assumptions used by quarantine (Lamport et al., 1982; Castro & Liskov, 1999; Biggio et al., 2012; Shejwalkar & Houmansadr, 2021). Appendix Table 4 makes these method-level differences explicit.

3 Problem Formulation

Consider a collective of N agents $\mathcal{A} = \{A_1, \ldots, A_N\}$ issuing tool queries against a shared environment $\mathcal{E}$, such as a code repository, document corpus, or API endpoint. The environment evolves: the ground-truth response $g(q,t)$ may differ from $g(q,t')$ for $t \neq t'$. RAVEL partitions semantic-cache state into an active index $\mathcal{C}_{\text{act}}$, which alone serves lookups, and an append-only historical store $\mathcal{H}$ used to compute trust evidence.

Definition 1 (Cache Entry)**.** *A cache entry is a semantically keyed artifact* $e = (k, v, \mathcal{P}(e), \tau_c, \tau_a, \mathbf{m}, s)$*, where* $k \in \mathbb{R}^d$ *is the query embedding,* v *is the cached result,* $\mathcal{P}(e)$ *is a provenance DAG,* τ_c *and* τ_a *are creation and last-access times,* $\mathbf{m} = (n_{\text{access}}, n_{\text{verify}}, n_{\text{contradict}}) \in \mathbb{N}^3$*, and* $s \in \{\text{active}, \text{historical}\}$*. At most one entry in a supersession lineage is active. Historical entries are excluded from semantic matching.*

Definition 2 (Provenance DAG)**.** *We represent provenance by the directed acyclic graph* $\mathcal{P}(e) = (\mathcal{V}, \mathcal{E})$*. At each node* $u \in \mathcal{V}$*, the stored agent identity is* $\alpha(u) \in \mathcal{A}$*, the timestamp is* $\tau(u)$*, the outcome signal is* $o(u) \in \{0,1\}$*, and the event type is* $\omega(u)$*. Allowed event types belong to* $\{$PRODUCE, CONSUME, VERIFY, CONTRADICT$\}$*. Dependency edges encode causal use; a directed* SUPERSEDES *edge from a fresh* PRODUCE *node to the replaced entry links active state to historical contradiction evidence without making the old value retrievable.*

When agent A_i issues query q at time t and finds an active candidate e with $\text{sim}(q, e.k) \geq \eta$, it selects $a \in \{$REUSE, VERIFY, RECOMPUTE$\}$. The retrieval threshold η is fixed before policy fitting and shared by every controller. REUSE returns $e.v$ at lookup cost c_r. VERIFY runs a query-type-specific current-state check and consumes $e.v$ only when the checked symbol signature, dependency edge, named test outcome, or document claim is semantically consistent, at cost c_v. RECOMPUTE executes the full tool call at cost c_x, where $c_x > c_v \gg c_r$. Successful verification appends VERIFY and CONSUME. Failed verification appends CONTRADICT; recomputation appends CONTRADICT when its result disagrees with $e.v$. Every recomputation moves e from $\mathcal{C}_{\text{act}}$ to $\mathcal{H}$ and installs its current result as a fresh PRODUCE entry linked by SUPERSEDES. Thus the active timestamp and value are refreshed even when recomputation confirms the cached result, while contradiction statistics change only on disagreement. Because verification is a learned semantic judgment rather than the retrospective evaluation oracle, false acceptance can leave nonzero stale and harmful rates even for Always-Verify.

Index the cache decisions made by all agents working on one issue as $d = 1, \ldots, H$, and let i_d identify the acting agent. The collective seeks a policy π minimizing

$$\begin{aligned} \min_{\pi} \quad & \mathbb{E}\Bigg[\sum_{d=1}^{H} c_{\text{lat}}(a_d) + \lambda_s \mathbf{1}[\text{stale}(d)] + \lambda_h \mathbf{1}[\text{harmful}(d)] \\ & \qquad\qquad -\lambda_b \,\text{sim}(q_d, e_d)\mathbf{1}[a_d = \text{REUSE}]\Bigg] \\ \text{s.t.} \quad & \sum_{d=1}^{H} c_{\text{ver}}(a_d) \leq B. \end{aligned} \tag{1}$$

Here $c_{\text{lat}}(\cdot)$ is latency cost, λ_s and λ_h penalize stale and harmful consumption, λ_b rewards high-similarity reuse, and B is one cap shared across the issue rather than a per-agent allowance. The evolving active index, partial outcome observability, and common cap couple decisions across agents. RAVEL uses a factored approximation: each hit is scored from shared metadata and $b_d = B_{\text{rem}}/B$, so an early verification changes the state presented to every later decision.

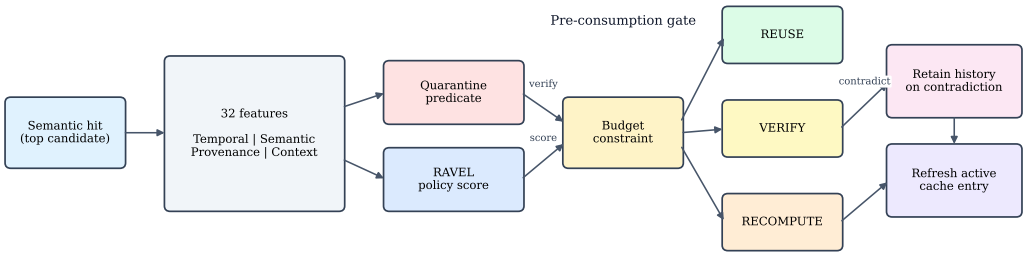


Figure 1: RAVEL's pre-consumption harmful-propagation control. Temporal, semantic, provenance, and producer–consumer context features represent each hit; quarantine and remaining budget constrain the tri-action policy. Contradictions remain as historical trust evidence while recomputation refreshes the active entry before subsequent consumption.

4 RAVEL: Evidence-Preserving Cache Governance

RAVEL implements the post-match gate in Figure 1. Every policy in the matched comparison surface receives one highest-similarity candidate, the same 32 observable features, the same three actions, the same quarantine convention, and the same remaining-budget scalar. This contract isolates decision quality from retrieval quality or privileged metadata.

4.1 Feature families and design rationale

For each candidate event (q, e) at time t, RAVEL forms $\mathbf{x} \in \mathbb{R}^{32}$ from four groups: temporal features (4) measure artifact age, verification recency, access recency, and environment change rate; semantic features (6) measure query–key similarity, embedding geometry, query type, and estimated drift; provenance features (14) summarize DAG depth, consumers, verification and contradiction histories, producer trust, agreement, and event structure; context-mismatch features (8) compare the consumer's task, dependencies, and downstream criticality with the producing context. Appendix B enumerates all features. The controller input is $[\mathbf{x}; b_d] \in \mathbb{R}^{33}$; the cap fraction is state, not a fifth artifact-feature family.

Each group targets a distinct route from a semantic match to harmful propagation. Temporal state captures invalidation pressure; semantic drift distinguishes lexical proximity from current meaning; provenance detects repeated or correlated contradictions; and context mismatch catches an otherwise valid artifact entering a different dependency chain. The shared budget signal converts these risks into allocation decisions. REUSE preserves capacity for low-risk hits, VERIFY purchases evidence for uncertain candidates, and RECOMPUTE resets value and active lineage when propagation risk outweighs reuse savings.

All inputs are observed immediately before decision d. Semantic drift summarizes the change in the candidate's local embedding neighborhood since creation; producer trust is the producer's completed verification-conditioned non-contradiction history; consumer agreement summarizes completed consistency outcomes among prior consumers in the lineage; downstream criticality is the current query's normalized unresolved-descendant mass in the issue dependency graph; and consumer experience counts completed same-type decisions for A_i. Events whose outcomes have not yet arrived are masked rather than treated as successes, and the current decision's verifier output or terminal outcome is never an input to its own action. Appendix B specifies the remaining coordinates and their observation sources.

4.2 Decision model and selective-label fit

Within the shared gate interface, RAVEL instantiates its decision layer as a four-layer MLP ($33 \to 512 \to 256 \to 128 \to 3$; ReLU, layer normalization, dropout $p = 0.1$). Evaluation takes the deterministic $\arg\max$ after softmax-policy fitting at temperature $\tau_{\text{temp}} = 0.5$. The per-decision

reward subtracts latency, stale consumption, and harmful consumption and adds a similarity-weighted reuse bonus, with $\lambda_s = 2.0$, $\lambda_h = 5.0$, and $\lambda_b = 0.3$.

Verification labels are selective because reused entries are not immediately checked. A logging-policy model estimates the probability of each of the three actions from the same 33-dimensional pre-action vector, and an action-conditional reward model supplies the direct term for each action. The observed-action residual is inverse-propensity corrected with denominator floor 0.05; unsupported counterfactual actions therefore contribute only their direct-model term rather than an extrapolated residual. Policy temperature, reward weights, and controller hyperparameters are fixed on disjoint validation episodes before evaluation. Appendix C gives the reward and clipped doubly robust objective (Dudík et al., 2014; Jiang & Li, 2016). These quantities supervise the gate only: every reported endpoint comes from simulator execution, fixed-request intervention replay, or closed-loop execution.

The matched controls use the same candidates, 33 inputs, logged decisions, action rewards, quarantine override, cap, and validation episodes; their comparison tests how effectively different policy classes convert identical evidence into the three actions. The rule-list search varies similarity, age, contradiction, and context-mismatch thresholds over {0.85,0.90,0.92,0.95}, {30,60,120} s, and {0.01,0.03,0.05}; validation selects 0.92/60 s/0.03/0.20 for similarity, age, contradiction, and context mismatch. Multinomial logistic regression tunes L_2 inverse regularization over {0.01,0.1,1,10} and selects 0.1. The shallow tree searches depth {2,3,4,5} and minimum leaf size {32,64,128}, selecting depth 4 with 64 decisions per leaf. Validation applies the same success–cost objective and deterministic tie-breaking by lower harmful-stale rate, then lower verification use.

Workflow baselines answer different but useful questions. LRU and TTL-60s use recency only; Semantic Dedup reuses an above-threshold semantic match; Confidence-0.8 additionally requires producer confidence of at least 0.8; Always-Verify checks every hit until the shared cap is exhausted; and Centralized serializes cache access through one service that invalidates entries after observed repository changes or contradictions. Centralized has global active state and provenance but no consumer-context score or learned allocation among the three actions. All methods receive the same execution ceiling, cache warm-up, repository state, and retrospective evaluator.

4.3 Quarantine, budget, and active-entry refresh

RAVEL quarantines an entry when either its contradiction rate exceeds δ_c or its producer's aggregate contradiction rate exceeds δ_a:

$$\textsc{Quarantine}(e) = \mathbf{1}[\rho_e > \delta_c] \vee \mathbf{1}\left[\rho_{\alpha(e)} > \delta_a\right], \tag{2}$$

where $\rho_e = n_{\text{contradict}}(e)/n_{\text{access}}(e)$ and $\rho_{\alpha(e)}$ aggregates the producer's historical entries; either rate is defined as zero before its first access. We use $\delta_c = 0.15$ and $\delta_a = 0.25$. Quarantine precedes learned scoring and requests verification. The cap is expressed in verification units: $c_{\text{ver}}(\textsc{Verify}) = 1$ and $c_{\text{ver}}(\textsc{Reuse}) = c_{\text{ver}}(\textsc{Recompute}) = 0$. Every simulator, replay, document-analysis, or closed-loop issue begins with $B = 36$, equal to 30% of the common 120-tool-call issue ceiling; it is one issue-level cap shared by all participating agents and does not scale again with N. The cap does not replenish. A requested verification atomically reserves one unit before the verifier runs. If no unit remains, quarantine and model-selected verification both route to recomputation before consumption.

The active/history split makes the refresh invariant explicit: the contradicted value cannot match a later lookup, but its producer, consumers, and contradiction events remain reachable from the fresh entry's lineage features. Atomic reservation also makes simultaneous agents respect one cap; an exhausted request is recomputed rather than silently reused.

5 Experiments

5.1 Questions, workloads, and protocols

The evaluation follows one matched artifact from admission to terminal outcome. It asks whether identical cache evidence can be converted into a better success–cost–harm frontier, whether raw staleness and outcome-changing propagation rank policies differently, and whether the ordering persists when actions alter future calls. Systems accounting, component removal, document analysis,

Algorithm 1 RAVEL: budgeted pre-consumption gate and evidence-preserving refresh

Require: Query q from A_i, active index $\mathcal{C}_{\text{act}}$, history $\mathcal{H}$, policy π_θ, remaining cap B_{rem}

$E_{\text{cand}} \leftarrow \{e \in \mathcal{C}_{\text{act}} : \text{sim}(q, e.k) \geq \eta\}$
if $E_{\text{cand}} = \emptyset$ **then**
  $y \leftarrow \text{EXECUTE}(q, \mathcal{E})$; insert fresh PRODUCE entry
  return y
end if
$e^* \leftarrow \arg\max_{e \in E_{\text{cand}}} \text{sim}(q, e.k)$
$\mathbf{z} \leftarrow [\text{EXTRACTFEATURES}(q, e^*, A_i, \mathcal{P}(e^*)); B_{\text{rem}}/B]$
if $\text{QUARANTINE}(e^*)$ **then**
  $a \leftarrow \text{VERIFY}$
else
  $a \leftarrow \arg\max_a f_\theta^{(a)}(\mathbf{z})$
end if
if $a = \text{VERIFY} \wedge B_{\text{rem}} < 1$ **then**
  $a \leftarrow \text{RECOMPUTE}$
end if
if $a = \text{REUSE}$ **then**
  Append CONSUME; update access statistics
  return $e^*.v$
end if
if $a = \text{VERIFY}$ **then**
  $B_{\text{rem}} \leftarrow B_{\text{rem}} - 1$
  $(\text{consistent}, y_v) \leftarrow \text{VERIFYAGAINSTENV}(q, e^*, \mathcal{E})$
  Append VERIFY; update verification statistics
  if consistent **then**
    Append CONSUME; update access statistics
    return $e^*.v$
  end if
  Append CONTRADICT; update contradiction statistics
end if
$y \leftarrow \text{EXECUTE}(q, \mathcal{E})$
if $\text{DISAGREES}(y, e^*.v)$ **then**
  Append CONTRADICT if absent
end if
Move e^* to $\mathcal{H}$; insert fresh active PRODUCE
Link fresh entry to e^* with SUPERSEDES; propagate lineage summaries
return y

adaptive contamination, and overlap–cap–churn sweeps then test the mechanism and its operating regimes.

Repository simulation. Sixteen agents analyze an evolving SWE-bench Python repository (Jimenez et al., 2024) with 847 files and 142k lines. Each issue-conditioned episode forms a dependency DAG over signature, dependency, test, and documentation queries; the scheduler assigns ready queries to agents, and semantically equivalent intents assigned to multiple agents determine the measured overlap rate. The terminal oracle evaluates the collective's issue answer against the snapshot-specific signatures, dependency edges, tests, and documentation. Commits arrive as a Poisson process at 2.3/min and edit 1–5 files. The churn axis is the hourly fraction of active artifacts invalidated after each edit is propagated through the recorded dependency graph; stress points alter edit selection while holding arrival rate fixed. The main point uses 40% overlap and evolves each episode for at most $T = 2{,}000$ collective simulator events. Separately, all policies face a 900-second wall-clock ceiling, a 120-tool-call ceiling, and the issue-level $B = 36$ verification cap defined in Section 4.3; T counts environment and scheduling events and does not multiply the cap. Each of five evaluation seeds contains 500 episodes, for 2500 terminal outcomes. Reaching neither a correct nor an incorrect terminal answer before a ceiling is a timeout. No Cache obtains fresh individual results but spends enough execution time to complete fewer downstream steps; Appendix Table 5 separates timeouts, wrong answers, and harmful-stale failures.

Fixed-request intervention replay. The observed workload contains 142 attempted SWE-bench issues, 12,847 tool calls, and 34.2% overlap from 12 GPT-4o task agents and 12 Claude-3.5-Sonnet verifiers. Each issue is represented as a fixed request DAG: repository snapshots, available branch continuations, and tool-call dependencies are frozen, but a cache policy substitutes returned artifacts and action costs at matched call sites. The substitutions update cache state, prompts at downstream recorded nodes, completion under the common ceiling, and the terminal candidate scored by the benchmark evaluator. Policies cannot invent a new request or branch in this protocol. Five shuffles change how issue DAGs interleave in the shared cache and scheduler, which can alter resource-limited completion even for No Cache. Each issue receives $B = 36$; task success is the number of evaluator-passing terminal candidates divided by all 142 attempts in each shuffle. This fixed-topology intervention identifies effects on observed requests, while the next protocol permits new branches and calls.

Held-out closed loop. A 24-agent collective evaluates 60 SWE-bench issue attempts sampled before execution from three repositories excluded from controller training and threshold selection; every attempt remains in the denominator. Ten issue-order/model-sampling seeds yield 600 issue-runs. Every run begins with $B = 36$ and the same wall-clock and tool-call ceilings as simulation. Policies can alter tool calls, branches, cache state, and later cap allocation. Random streams are paired by issue and seed until policy actions make paths diverge; paired 95% bootstrap intervals resample issues and seeds. Path divergence is the fraction of runs whose tool-call sequence differs from the paired recorded reference after the first cache-eligible call position. This common reference also defines divergence for No Cache, whose fresh outputs and sampling can depart from the recorded path without taking a cache action.

Models, verifier, and splits. GPT-4o (OpenAI, 2024) serves task agents, Claude-3.5-Sonnet (Anthropic, 2024) judges semantic consistency for verification, and DeepSeek-Coder-V2 (DeepSeek-AI et al., 2024) serves local code workers. Prompts and hosted-model revisions are frozen across policy arms within each issue and seed. Verification receives the cached artifact, the current query, and exactly one query-type-specific current-state observation: a re-resolved signature, dependency edge, named test result, or current document claim. It returns a consistency judgment and the observed current value, but never receives the retrospective stale label or terminal outcome. A false consistency judgment explains the nonzero stale rate under Always-Verify. Sentence-BERT (Reimers & Gurevych, 2019) provides 768-dimensional embeddings. RAVEL trains on 50k decisions from simulator rollouts disjoint from all reported evaluation episodes, issues, and seeds, using AdamW (learning rate $5{\times}10^{-4}$, batch 256, 10k steps). Evaluation uses 8×A100 80GB GPUs, CUDA 12.4, and PyTorch 2.3.1. Simulator seeds vary commits, query arrivals, and task order; replay shuffles vary issue merge order; closed-loop seeds additionally vary model sampling.

Estimands. An evaluation-only current-state oracle retrospectively labels every consumed hit; it is never a policy input. A stale event is harmful when replacing that artifact alone with the contemporaneous result, restoring the pre-consumption downstream state, and replaying the paired scheduler and random stream changes a failed terminal output to a passing one. Jointly repairing all stale consumptions defines an issue flip. Task success includes every attempted issue or episode, including timeouts. Relative Cost is $(n_x + 0.28n_v + 0.015n_r)/n_x^{\text{No Cache}}$ for recomputation, verification, and lookup counts. The two action weights are measured mean service costs relative to full recomputation on the common hardware; Section 5.4 adds routing, synchronization, state, storage, and orchestration work. Verification utilization is $n_v/36$ per issue, with exhausted requests counted as recomputation. Appendix D specifies denominators, primary-failure assignment, counterfactual pairing, and hierarchical resampling.

5.2 Matched-controller frontier

RAVEL delivers the highest task success and lowest execution cost in Table 1, while Appendix Table 5 shows fewer harmful-stale failures than every same-information control, placing it on the joint non-dominated frontier. Under the shared information contract, RAVEL improves over multinomial logistic by +1.3 [0.3, 2.3] success points and -0.025 [-0.038, -0.012] relative cost (paired 95% intervals). Corresponding RAVEL-minus-rule-list differences are +1.6 [0.5, 2.7] and -0.043 [-0.057, -0.029]; RAVEL-minus-tree differences are +1.7 [0.5, 2.9] and -0.019 [-0.033, -0.005]. Relative to Always-Verify, RAVEL trades a 0.9-point increase in diagnostic staleness for 2.4 points higher success and a 0.368 cost reduction. The strongest component result aligns with the mechanism:

Table 1: Repository-simulator frontier at 16 agents, 40% overlap, 2.3 Poisson commits/min, $B = 36$ per issue, and 2500 evaluated episodes. Latency and cost are relative to No Cache; means ± SE over five seeds.

Method	Lat. ↓	Hit ↑ (%)	Stale ↓ (%)	SR ↑ (%)	Cost ↓
No Cache	1.000	0.0	0.0	68.4±1.2	1.000
LRU	0.724±0.011	38.2±1.1	12.1±0.8	71.2±1.4	0.618±0.009
TTL-60s	0.687±0.009	41.5±1.0	8.3±0.6	73.8±1.1	0.585±0.008
Semantic Dedup	0.652±0.010	44.1±0.9	7.9±0.7	74.3±1.3	0.559±0.010
Confidence-0.8	0.671±0.008	43.8±1.2	6.7±0.5	75.1±1.0	0.572±0.007
Always-Verify	0.847±0.006	48.2±0.8	1.2±0.2	78.9±0.9	0.891±0.005
Centralized	0.742±0.012	52.6±1.3	1.8±0.3	79.4±0.8	0.684±0.011
Rule List	0.659±0.009	54.7±1.1	2.8±0.4	79.7±0.8	0.566±0.009
Multinomial Logistic	0.641±0.008	55.1±1.0	2.5±0.3	80.0±0.8	0.548±0.009
Shallow Tree	0.634±0.009	55.4±1.1	3.0±0.4	79.6±0.9	0.542±0.010
RAVEL	0.618±0.007	56.8±0.9	2.1±0.3	81.3±0.7	0.523±0.008

Table 2: Fixed-request SWE-bench intervention replay with 24 agents, 142 attempted issues, 12,847 tool calls, 34.2% overlap, $B = 36$ per issue, and five merge shuffles. Means ± SE; latency and cost are relative to No Cache.

Method	Lat. ↓	Hit ↑ (%)	Stale ↓ (%)	SR ↑ (%)	Cost ↓
No Cache	1.000	0.0	0.0	66.8±1.4	1.000
Semantic Dedup	0.689±0.012	38.4±1.2	9.6±0.9	72.1±1.4	0.598±0.012
Always-Verify	0.874±0.009	42.7±1.1	1.8±0.3	75.8±1.1	0.917±0.007
Centralized	0.773±0.015	46.2±1.5	2.6±0.4	76.4±1.0	0.721±0.014
Rule List	0.690±0.012	48.9±1.3	4.0±0.5	76.9±1.0	0.625±0.012
Multinomial Logistic	0.674±0.011	49.8±1.2	3.8±0.5	77.4±0.9	0.609±0.011
Shallow Tree	0.665±0.012	50.2±1.3	4.1±0.5	77.0±1.0	0.603±0.012
RAVEL	0.641±0.010	51.6±1.2	3.4±0.4	78.7±0.9	0.578±0.011

removing prospective provenance loses 4.2 success points and adds 3.7 stale-hit points (Appendix Table 13).

5.3 Observed requests and policy-dependent harmful propagation

Fixed-request replay holds the observed call topology constant and isolates the returned-artifact intervention. RAVEL remains ahead of the matched logistic controller by +1.3 [0.2, 2.4] success points and -0.031 [-0.047, -0.015] cost. The rule-list differences are +1.8 [0.4, 3.2] and -0.047 [-0.065, -0.029]; tree differences are +1.7 [0.3, 3.1] and -0.025 [-0.042, -0.008]. Appendix Table 7 reveals the causal object hidden by stale-hit rate: among 1127 stale RAVEL hits pooled across shuffles, 1001 are harmless and 126 are harmful, producing 38 flipped issue outcomes. Centralized records fewer stale hits yet produces 150 harmful events and 45 flips. Prospective admission therefore preserves harmless reuse while concentrating checks on outcome-relevant risk.

Held-out policy-dependent execution tests whether the gain persists after actions alter later calls. This table carries forward logistic control, the highest-success same-information controller in Table 1, alongside the workflow anchors; Table 2 reports all three matched policy classes under fixed requests. Relative to logistic control, RAVEL gains +1.8 [0.6, 3.0] success points and -0.045 [-0.059, -0.031] cost; relative to Centralized, the differences are +2.5 [1.1, 3.9] and -0.165 [-0.184, -0.146]. It uses 69.1±1.4% of the cap versus 72.5±1.6% for logistic control and reduces harmful-stale rate from 1.0±0.2% to 0.7±0.1%.

Table 3: Held-out closed-loop repository analysis with 24 agents, 35.1±1.8% overlap, $B = 36$ per issue, 60 issues from three unseen repositories, and ten seeds (600 issue-runs). Harm is harmful-stale rate; Div. is path divergence; Ver. is verification-cap utilization. Means ± SE.

Method	Success /600	SR ↑ (%)	Cost ↓	Harm ↓ (%)	Div. (%)	Ver. (%)
No Cache	419	69.8±1.6	1.000	0.0	22.7±2.2	0.0
Always-Verify	472	78.7±1.2	0.914±0.008	0.5±0.1	16.2±1.7	97.3±0.8
Centralized	476	79.3±1.1	0.761±0.016	0.9±0.2	17.4±1.8	84.8±1.5
Logistic	480	80.0±1.0	0.641±0.014	1.0±0.2	18.0±1.8	72.5±1.6
RAVEL	491	81.8±0.9	0.596±0.013	0.7±0.1	18.6±1.9	69.1±1.4

5.4 Overhead-inclusive systems frontier

Full accounting narrows RAVEL's cost margin over logistic control from 0.025 to 0.020 (0.566 versus 0.586) and retains a 0.182 reduction versus Centralized. The eight components are measured over the same seeds, amortized per issue, normalized by No Cache, and charged once to the inducing policy. Appendix Tables 8 and 9 report this ledger: the success–cost ordering survives coordination and evidence maintenance.

5.5 Cross-workload transfer and operating calibration

On 2500 disjoint document-analysis episodes with frozen controller selection, RAVEL reaches 82.6±0.8% success at 0.579±0.009 cost. Against logistic control, paired differences are +1.1 [0.2, 2.0] success points and -0.025 [-0.039, -0.011] cost; Appendix Table 14 specifies this versioned-claim workload and the full comparison.

Figure 2 locates the gain. At 15% overlap, learned gates differ by at most 0.5 success points; at 40% and 55%, RAVEL leads Centralized by 1.9 and 3.4 points while costing 0.161 and 0.143 less. Selective checking helps most under scarce caps, where Always-Verify loses completion capacity; near 15% invalidation/hour, fresh execution recovers its value.

6 Discussion

The replay decomposition explains the joint frontier: despite admitting 1,001 harmless stale hits, RAVEL reduces harmful events from 150 to 126 and issue flips from 45 to 38 versus Centralized. Consumer context and contradiction lineage separate exposure from consequence, cap state prices verification, and correction informs later admission. Ablations further assign the ordinary-workload gains to provenance and drift, and the contamination response to quarantine.

Across the evaluated repositories and versioned documents, conditioned reuse helps at moderate overlap, cap-aware checking under scarce verification, and fresh execution under rapid invalidation. Counterfactual attribution uses controlled environments; team size and overlap co-vary in one sweep, and only rotation stress relaxes stable producer identity. Shared provenance also requires access control, minimization, and bounded retention. The resulting rule is direct: govern reuse most aggressively where propagation opportunity and downstream consequence are both high.

7 Conclusion

RAVEL governs cross-agent consumption through outcome-aware admission, a shared cap, and evidence-preserving refresh. Against same-information logistic control, it adds +1.3 [0.3, 2.3] success points at -0.025 [-0.038, -0.012] cost in simulation and +1.8 [0.6, 3.0] points at -0.045 [-0.059, -0.031] cost across 600 policy-dependent runs. Replay and ablations attribute this frontier to consequential admission and prospective provenance. Shared memory can preserve duplication savings and trust history together by controlling which matched artifacts alter collective behavior, not by eliminating every stale hit.

REFERENCES

Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*, pp. 1638–1646, 2014.

Anthropic. Introducing claude 3.5 sonnet, 2024. URL https://www.anthropic.com/news/claude-3-5-sonnet. Anthropic News.

Fu Bang. GPTCache: An open-source semantic cache for LLM applications enabling faster answers and cost savings. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software*, pp. 212–218. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.nlposs-1.24. URL https://aclanthology.org/2023.nlposs-1.24/.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *International Conference on Machine Learning*, pp. 1467–1474, 2012.

Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Why and where: A characterization of data provenance. In *International Conference on Database Theory*, pp. 316–330, 2001.

Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance. In *USENIX Symposium on Operating Systems Design and Implementation*, pp. 173–186, 1999.

James Cheney, Laura Chiticariu, and Wang-Chiew Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: Amazon's highly available key-value store. In *ACM Symposium on Operating Systems Principles*, pp. 205–220, 2007.

DeepSeek-AI et al. DeepSeek-Coder-V2: Breaking the barrier of closed-source models in code intelligence. *arXiv preprint arXiv:2406.11931*, 2024. URL https://arxiv.org/abs/2406.11931.

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

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Matteo Interlandi, Kshitij Shah, Sai Deep Tetali, Muhammad Ali Gulzar, Seunghyun Yoo, Miryung Kim, Todd Millstein, and Tyson Condie. Titian: Data provenance support in Spark. *Proceedings of the VLDB Endowment*, 9(3):216–227, 2015. doi: 10.14778/2850583.2850595. URL https://doi.org/10.14778/2850583.2850595.

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *International Conference on Machine Learning*, pp. 652–661, 2016.

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

David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In *ACM Symposium on Theory of Computing*, pp. 654–663, 1997.

Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems*, volume 20, pp. 817–824, 2007. URL https://proceedings.neurips.cc/paper/2007/hash/4b04a686b0ad13dce35fa99fa4161c65-Abstract.html.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36, 2023.

Xiaoyang Li, Yiqi Wang, Haohui Lu, Zhi Chen, Mo Li, Pingan Song, Mingkai Zheng, and Taotao Cai. MemTX: Transactional belief commit for stateful agent memory. *arXiv preprint arXiv:2607.23929*, 2026. URL https://arxiv.org/abs/2607.23929.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Yuzhen Mao, Qitong Wang, Martin Ester, and Ke Li. IceCache: Memory-efficient KV-cache management for long-sequence LLMs. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yHxSKM9kdr.

Yanki Margalit, Nurit Cohen-Inger, Erni Avram, Ran Taig, and Oded Margalit. Governed shared memory for multi-agent LLM systems. *arXiv preprint arXiv:2606.24535*, 2026. URL https://arxiv.org/abs/2606.24535.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

Rajesh Nishtala, Hans Fugal, Steven Grimm, Marc Kwiatkowski, Herman Lee, Harry C Li, Ryan McElroy, Mike Paleczny, Daniel Peek, Paul Saab, David Stafford, Tony Tung, and Venkateshwaran Venkataramani. Scaling memcache at Facebook. In *USENIX Symposium on Networked Systems Design and Implementation*, pp. 385–398, 2013. URL https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/nishtala.

OpenAI. GPT-4o system card. Technical report, OpenAI, 2024. URL https://cdn.openai.com/gpt-4o-system-card.pdf.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47717–47737, 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *ACM Symposium on User Interface Software and Technology*, 2023.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.810. URL https://aclanthology.org/2024.acl-long.810/.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/28e50ee5b72e90b50e7196fde8ea260e-Abstract-Conference.html.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Empirical Methods in Natural Language Processing*, pp. 3982–3992, 2019.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Syed Huma Shah. Grounded cache routing for retrieval-augmented generation: When is it safe to reuse an answer? *arXiv preprint arXiv:2605.27494*, 2026. URL https://arxiv.org/abs/2605.27494.

Virat Shejwalkar and Amir Houmansadr. Manipulating the Byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *Network and Distributed System Security Symposium*, 2021.

Ilias Stogiannidis, Stavros Vassos, Prodromos Malakasiotis, and Ion Androutsopoulos. Cache me if you can: An online cost-aware teacher-student framework to reduce the calls to large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14999–15008, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.1000. URL https://aclanthology.org/2023.findings-emnlp.1000/.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Wenjun Xiong, Yijin Zhou, Jiaqian Wang, Shangding Gu, Bo Tang, Zhiyu Li, Feiyu Xiong, Ying Wen, and Muning Wen. MAPLE-Guard: Memory-aware link enforcement against memory-link poisoning in multi-agent systems. *arXiv preprint arXiv:2608.00426*, 2026. URL https://arxiv.org/abs/2608.00426.

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

John Yang, Kilian Lieret, Carlos E Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025. doi: 10.52202/085713-3239. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/8b86cf5ace600c48fd188efbb8dedec8-Abstract-Datasets_and_Benchmarks_Track.html.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. CacheBlend: Fast large language model serving for RAG with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 94–109. Association for Computing Machinery, 2025. doi: 10.1145/3689031.3696098. URL https://doi.org/10.1145/3689031.3696098.

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

Fengji Zhang, Bei Chen, Yue Zhang, Jacky Keung, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, and Weizhu Chen. RepoCoder: Repository-level code completion through iterative retrieval and generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2471–2484, 2023.

Banghua Zhu, Ying Sheng, Lianmin Zheng, Clark Barrett, Michael I. Jordan, and Jiantao Jiao. Towards optimal caching and model selection for large model inference. In *Advances in Neural Information Processing Systems*, volume 36, pp. 59062–59094. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/b914a8fcea5c176cf1ed75c762ce27fd-Abstract.html.

AI USE STATEMENT

Generative AI tools assisted manuscript preparation by revising language and presentation, normalizing submission formatting, checking citation metadata, and performing consistency checks. The authors reviewed all AI-assisted edits and take responsibility for the final manuscript.

A Related-work decision matrix

Table 4 compares the decision locus and state transition of the closest neighboring mechanisms. "Three-way" distinguishes verification from both direct reuse and full regeneration; "History" requires corrected serving state to retain the superseded lineage as prospective evidence.

Table 4: Mechanism-level contrast with the closest cited systems. Owner denotes cross-agent production and consumption; Cap is a shared verification constraint; Outcome denotes terminal harm attribution.

System / mechanism	Read gate	Three-way	Owner	Cap	Outcome	History
GroundedCache	yes	–	–	–	partial	–
Governed Shared Memory	partial	–	yes	–	–	yes
MemTX	–	–	yes	–	yes	yes
MAPLE-Guard	yes	–	yes	–	yes	partial
Contextual bandit / DR	generic	yes	–	partial	–	–
RAVEL	yes	yes	yes	yes	yes	yes

B Feature inventory

The 32-dimensional vector is computed at lookup time from the candidate, events completed before that lookup, and the current consumer context. Temporal (4) records artifact age $t - \tau_c$, elapsed time since the latest completed verification, elapsed time since access, and the recent environment-change rate. Semantic (6) records query–key cosine similarity, embedding-norm ratio, categorical query type, change in the candidate's local embedding neighborhood since creation, query specificity, and current embedding-cluster density.

Provenance (14) records DAG depth, unique prior consumers, verification-to-access ratio, contradiction rate, producer trust from completed checked outcomes, agreement among completed consumer outcomes, a recency-weighted contradiction trend, four event-type counts, a recency-weighted verification count, producer–consumer history overlap, and lineage-chain diversity. Context mismatch (8) records producer–consumer task-embedding distance, dependency-chain overlap, task-type compatibility, normalized unresolved downstream criticality, the consumer's completed same-type experience, query complexity, context-window utilization, and cross-agent task correlation. Counts are log-scaled and bounded rates remain in $[0, 1]$ before training-split normalization. Pending outcomes are masked; no feature reads the current action's check or any future terminal label. The remaining-cap fraction b_d is appended separately, yielding the 33-dimensional controller input.

C Selective-label objective

For logged decision j, RAVEL uses

$$r_j(a) = -c_{\text{lat}}(a) - \lambda_s \mathbf{1}[\text{stale}_j(a)] - \lambda_h \mathbf{1}[\text{harmful}_j(a)] + \lambda_b \,\text{sim}(q_j, e_j)\mathbf{1}[a = \textsc{Reuse}]. \quad (3)$$

Let a_j^ℓ, $p_j(a \mid \mathbf{x}_j)$, and $m_j(a)$ denote the logged action, fitted three-action logging propensity, and action-conditional reward model. Both nuisance models use only the pre-action 33-dimensional vector and are fit on the training corpus; controller and clipping choices are fixed on disjoint validation episodes. The corrected target is

$$\widehat{r}_j^{\,\text{DR}}(a) = m_j(a) + \frac{\mathbf{1}[a = a_j^\ell]}{\max\{p_j(a_j^\ell \mid \mathbf{x}_j),\, 0.05\}} \left(r_j(a_j^\ell) - m_j(a_j^\ell)\right), \quad (4)$$

and training minimizes $-\sum_j \sum_a \pi_\theta(a \mid \mathbf{x}_j, b_j)\widehat{r}_j^{\,\text{DR}}(a)$. The residual term appears only for the observed action; other actions use the direct-model term. All three actions occur in the 50k-decision training corpus, while replay, document, and closed-loop evaluation issues are excluded from fitting and selection.

D Outcome accounting and matched-control intervals

For each stale consumption, the single-event intervention replaces only its returned artifact, restores the pre-consumption downstream state, and re-executes with the same repository snapshot, scheduler events, and random stream. A harmful event changes a failed terminal outcome to a passing one. The issue-level intervention replaces all stale consumptions and defines a flip; several event-level harms can therefore map to one issue. The terminal ledger assigns timeout when no terminal candidate is reached, harmful stale when joint repair passes, and wrong answer otherwise. Stale rate divides stale hits by consumed hits, harmful rate divides harmful stale events by consumed hits, and task success and flips use all attempted issues or episodes. Paired intervals resample native issues or episodes within seed and then aggregate across seeds.

Table 5: Raw terminal outcomes across 2500 repository-simulator episodes at 16 agents, 40% overlap, 2.3 commits/min, and $B = 36$ per issue. Five seeds vary commits, queries, and task order; failure columns are mutually exclusive primary causes.

Method	Success	Timeout	Wrong answer	Harmful stale
No Cache	1710	522	268	0
Always-Verify	1973	381	121	25
Centralized	1985	332	138	45
Rule List	1993	314	142	51
Multinomial Logistic	2000	309	141	50
Shallow Tree	1990	305	150	55
RAVEL	2033	286	132	49

The raw counts explain why fresh execution does not guarantee task completion under a common resource ceiling. RAVEL avoids enough repeated tool work to reduce timeouts while keeping harmful stale failures below the same-information controls. Centralized has 45 rather than RAVEL's 49 simulator episodes assigned harmful stale as the mutually exclusive primary cause; RAVEL's simulator advantage is the joint success–cost frontier, while Table 7 supplies the separate event-level Centralized comparison. Paired intervals in Section 5.2 bootstrap episodes within seed; replay intervals bootstrap issues within merge shuffle.

E Operating-regime comparison

Table 6: Success/cost across overlap at 16 agents, 2.3 commits/min, and $B = 36$ per issue. Values are five-seed means over 2500 episodes per setting; the 15% slice identifies the low-benefit activation region.

Method	15% overlap SR / Cost	40% overlap SR / Cost	55% overlap SR / Cost
No Cache	68.4 / 1.000	68.4 / 1.000	68.4 / 1.000
Semantic Dedup	69.7 / 0.914	74.3 / 0.559	78.6 / 0.438
Always-Verify	70.3 / 0.968	78.9 / 0.891	81.2 / 0.821
Centralized	70.1 / 0.937	79.4 / 0.684	82.7 / 0.541
Rule List	70.2/0.916	79.7/0.566	84.4/0.427
Multinomial Logistic	70.4/0.912	80.0/0.548	84.9/0.416
Shallow Tree	70.1/0.905	79.6/0.542	84.6/0.409
RAVEL	70.6 / 0.908	81.3 / 0.523	86.1 / 0.398

At 15% overlap, all learned gates differ by at most 0.5 success points, identifying the low-benefit activation region. At 40%, RAVEL improves over Centralized by 1.9 points and 0.161 cost; at 55%, the differences grow to 3.4 points and 0.143 cost. Figure 4 reports the joint team-size/overlap regime because the two vary together in this workload.

F REPLAY STALE-EVENT DECOMPOSITION

Table 7: Pooled fixed-request counts for 24 agents, 34.2% overlap, $B = 36$ per issue, and five merge shuffles (12,847 calls and 142 issue attempts per shuffle). Stale equals harmless plus harmful; flips count issue outcomes changed by harmful consumption.

Method	Cache hits	Stale	Harmless	Harmful	Flips
Semantic Dedup	24666	2368	1904	464	91
Always-Verify	27428	494	392	102	47
Centralized	29677	772	622	150	45
RAVEL	33145	1127	1001	126	38

The decomposition is intentionally event-level: one issue can contain several harmful stale consumptions but contributes at most one flipped outcome. RAVEL's additional stale admissions are predominantly harmless (1,001 of 1,127), while its harmful count and issue flips remain below Centralized despite a larger hit volume.

G SYSTEMS-OVERHEAD COMPONENTS

Table 8: All-in repository-simulator accounting at 16 agents, 40% overlap, 2.3 commits/min, $B = 36$ per issue, and 2500 episodes. Cost adds routing, synchronization, shared-state, provenance, feature, storage, verifier-service, and orchestration work. Latency statistics are relative to No Cache; means $\pm$ SE over five seeds.

Method	Cost ↓	Mean lat. ↓	p50 ↓	p95 ↓	Coord. (%)
Always-Verify	0.912±0.006	0.865±0.006	0.860	0.879	5.4±0.5
Centralized	0.748±0.014	0.789±0.013	0.776	0.824	8.9±0.9
Logistic	0.586±0.010	0.664±0.009	0.656	0.695	6.8±0.6
RAVEL	0.566±0.010	0.647±0.009	0.639	0.672	7.6±0.7

Table 9: Additive normalized overhead beyond execution cost at 16 agents, 40% overlap, 2.3 commits/min, $B = 36$ per issue, and 2500 episodes. Five-seed panels report all measured systems components.

Method	Routing	Sync.	Shared state	DAG R/W
Always-Verify	0.002	0.003	0.003	0.003
Centralized	0.012	0.014	0.008	0.006
Multinomial Logistic	0.004	0.005	0.004	0.007
RAVEL	0.004	0.005	0.004	0.007
Method	**Features**	**Storage**	**Verifier svc.**	**Recompute orch.**
Always-Verify	0.000	0.002	0.006	0.002
Centralized	0.004	0.004	0.009	0.007
Multinomial Logistic	0.007	0.003	0.005	0.003
RAVEL	0.009	0.003	0.006	0.005

The eight components sum to 0.021, 0.064, 0.038, and 0.043 for Always-Verify, Centralized, logistic regression, and RAVEL, respectively, reproducing the all-in totals in Table 8. Shared provenance is charged to Centralized and matched controls whenever they consume its fields.

H SECURITY AND PROVENANCE EVIDENCE

Table 10: Threat and evidence matrix. Detection means routing an injected artifact to VERIFY or RECOMPUTE before harmful consumption. Verifier outputs and contradiction records define the trusted enforcement boundary. "RAVEL clean reference" denotes the fixed no-injection point of 81.3 SR and 0.566±0.010 all-in cost, not success under the attack condition; Table 11 reports attack-conditioned outcomes.

Condition	Protected object / property	Capability / trust assumption	Enforcement boundary
Clean	Active artifact / current reuse	Repository edits / trusted checks	Pre-use gate
10% overt	Active artifact / block harmful use	High-rate injection / trusted checks and log	Quarantine
2% stealthy	Consumer output / block harmful use	Low-rate injection / trusted checks and log	Gate and log
3-agent collusion	Producer trust / limit propagation	Coordinated injection / stable identity and log	Producer gate
Delayed contradiction	Active lineage / preserve evidence	Delayed feedback / append-only log	History retention
Combined adaptive	Task outcome / graceful degradation	Combined powers / trusted checks and log	Gate and budget
Producer rotation	Producer trust / cold-start safety	Held-out identities / trusted checks and log	Entry gate

Condition	Comparator / adaptive setting	Clean reference / overhead	Failure mode / evidence trace
Clean	Logistic / held-out tasks	81.3 SR; 0.566±0.010 all-in	Ordinary drift / Tables 1, 8
10% overt	Centralized / held-out seeds	RAVEL clean reference	Repeated contradictions / Table 11
2% stealthy	Centralized / adaptive rate	RAVEL clean reference	Sparse history / Table 11
3-agent collusion	Centralized / adaptive group	RAVEL clean reference	Correlated producers / Table 11
Delayed contradiction	Centralized / adaptive delay	RAVEL clean reference	Late signal / Table 11
Combined adaptive	Centralized / adaptive mix	RAVEL clean reference	Detection loss / Table 11
Producer rotation	Centralized / held-out IDs	RAVEL clean reference	Cold start / Table 11

The protected object is the cached artifact and any downstream output that consumes it. Attackers can inject matched incorrect artifacts but cannot forge verifier outputs or alter contradiction records; producer identities remain stable before the rotation test. The matrix connects each capability to trust, enforcement, adaptive comparison, benign utility, overhead, failure mode, and paper-internal evidence.

Each attack seed exposes 10000 eligible cache-hit opportunities. The overt, stealthy, and colluding conditions inject 1000, 200, and 800 artifacts per seed, respectively; delayed-contradiction, combined, and producer-rotation tests each evaluate 800 injections per seed. Detection denominators include every injected artifact, while task-error and success denominators remain episode-level.

Table 11: Contamination results at 16 agents, 40% benign overlap, $B = 36$ per issue, and 10000 eligible hits per seed. Top: 10% overt adversaries. Bottom: adaptive attacks comparing full RAVEL with Centralized. Means ± SE over five seeds.

Condition	Method	Task Err. ↓ (%)	Detection ↑ (%)	Task SR ↑ (%)
Overt 10%	LRU	28.4±1.6	–	–
	Semantic Dedup	19.3±1.2	–	–
	Always-Verify	5.2±0.4	–	–
	Centralized	4.8±0.5	–	–
	RAVEL (no quarantine)	8.7±0.7	62.4±2.1	–
	RAVEL	3.1±0.3	84.7±1.4	–
Stealthy 2%	RAVEL	4.7±0.5	71.3±2.3	76.8±0.9
	Centralized	6.2±0.6	–	75.1±1.0
Colluding 3 agents	RAVEL	5.8±0.6	68.4±2.6	75.4±1.0
	Centralized	7.5±0.7	–	73.5±1.1
Delayed contradiction	RAVEL	3.9±0.4	76.1±1.9	77.3±0.8
	Centralized	5.6±0.5	–	75.7±0.9
Combined adaptive	RAVEL	7.1±0.7	61.8±2.9	73.1±1.1
	Centralized	9.2±0.9	–	70.6±1.2
Producer rotation	RAVEL	6.4±0.6	64.9±2.5	74.0±1.0
	Centralized	8.3±0.8	–	71.9±1.2

Detection includes both verification and recomputation, which explains the nonzero rate without quarantine: the learned gate can independently reject a suspicious artifact. Under overt injection, quarantine reduces task error from 8.7% to 3.1%. Under adaptive producer rotation, RAVEL retains a 2.1-point success advantage over Centralized. Under the hardest combined condition, it retains 73.1% versus 70.6% success while sparse and delayed contradiction history reduces detection to 61.8%.

Table 12: Quarantine-threshold sensitivity under the combined adaptive attack. Each cell reports task error / detection / benign SR / verification utilization (%); five-seed means.

δ_c	δ_a	Err. / Det. / SR / Ver.
0.10	0.20	6.6/66.8/80.6/27.9
0.10	0.25	6.7/66.1/80.8/26.8
0.10	0.30	6.9/64.8/81.0/25.7
0.15	0.20	6.9/63.2/80.9/26.1
0.15	0.25	7.1 / 61.8 / 81.3 / 23.4
0.15	0.30	7.4/59.9/81.4/21.8
0.20	0.20	7.3/61.1/81.2/24.7
0.20	0.25	7.6/58.7/81.5/22.0
0.20	0.30	7.9/56.4/81.7/20.1

The threshold neighborhood exposes a smooth utility–detection tradeoff rather than a brittle optimum. More aggressive quarantine improves detection while spending more verification and modestly reducing benign success.

I ABLATION RESULTS

Table 13: RAVEL component ablation in 2500 repository-simulator episodes at 16 agents, 40% overlap, 2.3 commits/min, and $B = 36$ per issue. Means ± SE over five seeds.

Configuration	Task SR ↑ (%)	Stale hit ↓ (%)	Rel. latency ↓
RAVEL	81.3±0.7	2.1±0.3	0.618±0.007
− provenance features	77.1±0.9	5.8±0.5	0.641±0.010
− propensity correction	79.4±0.8	3.4±0.4	0.632±0.008
− quarantine mechanism	80.1±0.8	2.4±0.3	0.621±0.007
− semantic-drift features	78.8±0.9	4.1±0.5	0.647±0.009

Provenance provides the largest clean-workload effect, while semantic drift and propensity correction each improve both success and stale-hit control. Quarantine changes clean latency little but drives the overt-attack improvement in Table 11. The complementary pattern supports an integrated mechanism: context and provenance govern ordinary stale risk, and the hard gate supplies a distinct response to contamination.

J DOCUMENT-ANALYSIS WORKLOAD

The document workload maps repository artifacts to versioned claims and dependency edges to citation links. Agents resolve current claims, supporting relations, and cross-document dependencies; updates replace claim versions and invalidate dependent cached artifacts. Terminal success compares the collective answer with the current versioned claim state. Repository episodes, document episodes, and their seeds are disjoint, and the repository-trained controller and thresholds are frozen for this evaluation.

Table 14: Document analysis over 2500 episodes at 16 agents, 36% overlap, 8% active-document invalidation/hour, and $B = 36$ per issue. All policies share candidates, features, actions, cap, frozen selection, and five seeds; means ± SE.

Method	Lat. ↓	Hit ↑ (%)	Stale ↓ (%)	SR ↑ (%)	Cost ↓
No Cache	1.000	0.0	0.0	72.1±1.0	1.000
LRU	0.741±0.012	35.4±1.2	14.2±0.9	74.5±1.3	0.637±0.010
TTL-60s	0.708±0.010	38.9±1.1	9.7±0.7	76.2±1.1	0.612±0.009
Semantic Dedup	0.681±0.011	41.2±1.0	9.1±0.8	76.8±1.2	0.588±0.011
Always-Verify	0.862±0.007	45.1±0.9	1.4±0.3	80.2±0.8	0.904±0.006
Centralized	0.726±0.012	48.1±1.2	2.4±0.4	80.8±0.9	0.681±0.011
Rule List	0.688±0.011	49.0±1.1	3.5±0.5	80.4±0.9	0.618±0.010
Multinomial Logistic	0.671±0.010	49.8±1.1	3.2±0.4	81.5±0.8	0.604±0.010
Shallow Tree	0.666±0.011	50.2±1.2	3.6±0.5	81.1±0.9	0.598±0.011
RAVEL	0.653±0.009	51.4±1.0	2.8±0.4	82.6±0.8	0.579±0.009

RAVEL improves success by +1.1 [0.2, 2.0] points and cost by -0.025 [-0.039, -0.011] relative to logistic regression, with paired 95% bootstrap intervals over tasks within seed. Always-Verify records fewer stale hits, whereas RAVEL reaches the strongest joint success–cost point. This second content and invalidation structure preserves the governance mechanism's ordering without workload-specific selection.

K ADDITIONAL DIAGNOSTIC FIGURES

Figure 2 maps the measured operating regimes summarized in Section 5.5.

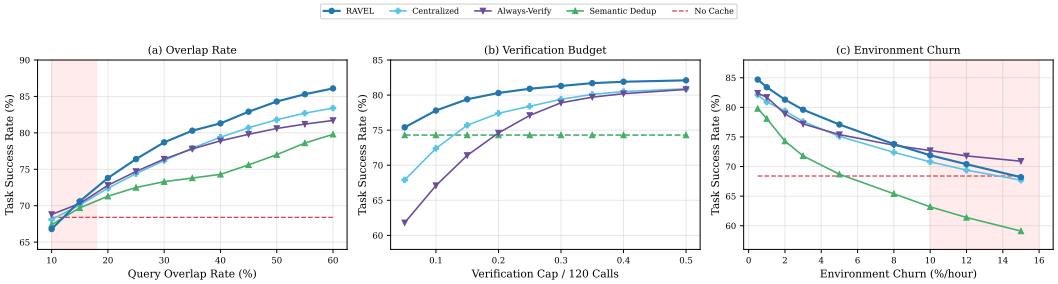


Figure 2: Operating calibration for the 16-agent repository simulator over five seeds. Panels vary query overlap, the verification cap as a fraction of the 120-call issue ceiling, and the hourly fraction of active artifacts invalidated by repository edits. Connecting lines are visual guides between measured operating points.

Figure 3 checks whether the aggregate ordering seen in simulation remains visible when decisions are restricted to observed request topology.

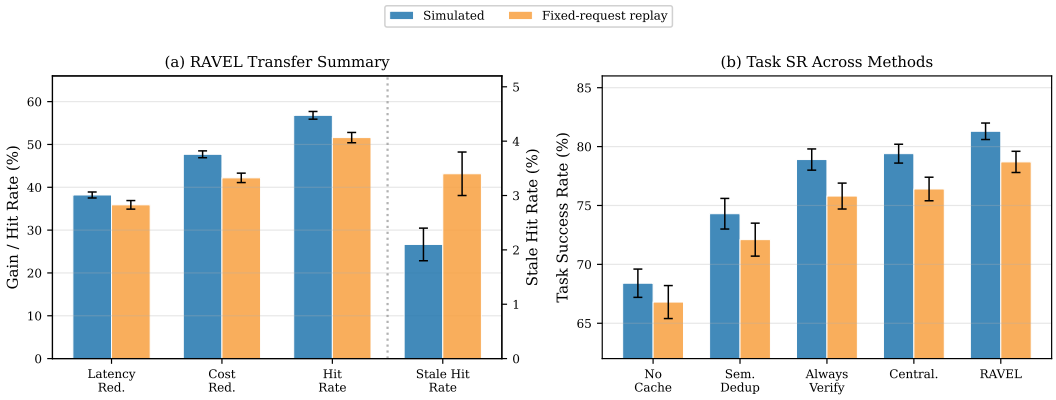


Figure 3: Repository simulation and fixed-request replay. Panel (a) compares RAVEL's latency reduction, cost reduction, hit rate, and stale-hit rate; panel (b) shows task-success ordering. Replay tests observed request topology, while Table 3 supplies policy-dependent evidence.

Figure 4 separates the low-reuse region from the larger collectives in which overlapping queries can amortize cache governance.

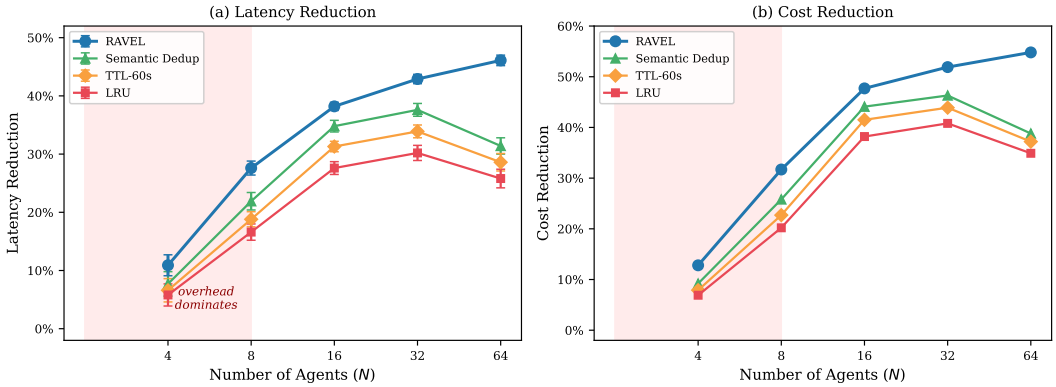


Figure 4: Collective-size/overlap sweep across 4–64 agents. Overlap rises with team size in this workload, so the figure identifies a joint operating regime rather than an asymptotic scaling law. Sparse reuse at 4–8 agents leaves little repeated work to amortize; benefits expand with overlap.

Figure 5 visualizes why stale-hit rate alone cannot represent the success–cost–harm objective used in the main evaluation.

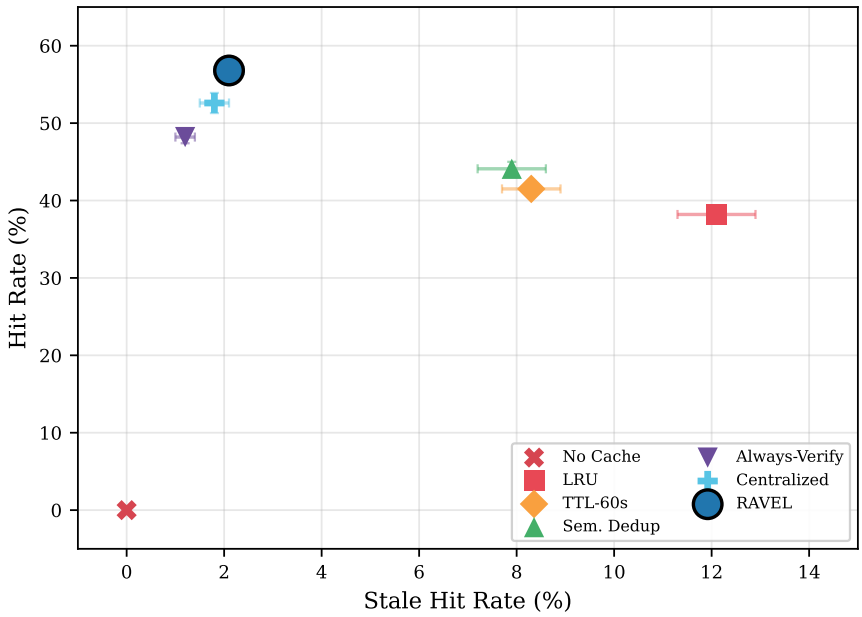


Figure 5: Hit rate versus diagnostic stale-hit rate in 16-agent repository simulation. RAVEL lies above-left of heuristic reuse policies; Always-Verify remains the lower-staleness anchor. Main reliability conclusions additionally use harmful-stale events and task outcomes. Error bars are standard errors over five seeds.

Figure 6 complements the clean-workload ablations with an overt-contamination stress path over increasing adversarial participation.

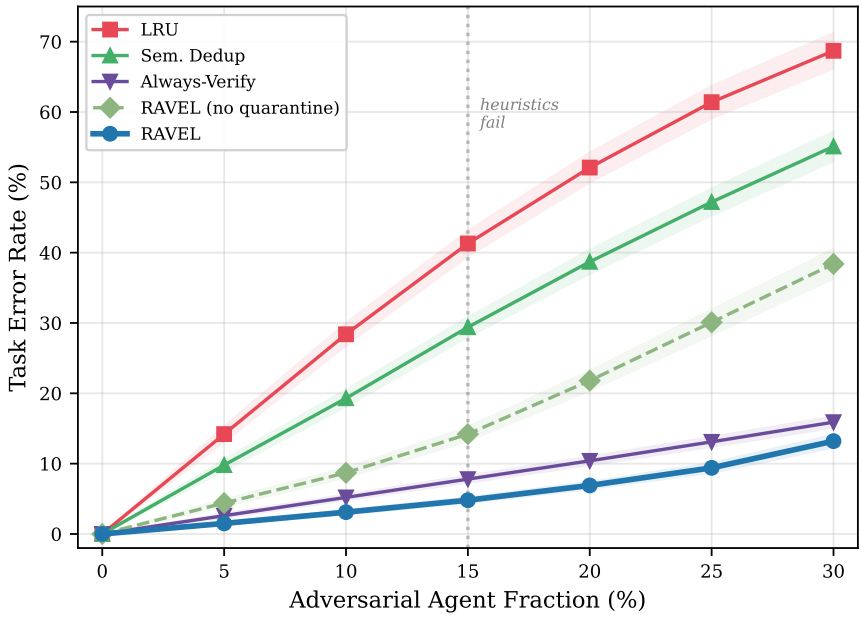


Figure 6: Task-error rate across overt adversarial-agent fractions at 16 agents. RAVEL's quarantine delays degradation relative to heuristic reuse and the learned gate without quarantine; Table 11 adds matched Centralized comparisons under adaptive attacks.