# COLLABMEMBENCH: BENCHMARKING SHARED MEMORY UNDER ASYNCHRONOUS MULTI-AGENT WORK

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

## ABSTRACT

Asynchronous multi-agent systems need shared memory: decisions, evidence, and updated constraints from one agent must remain available to others beyond private histories. During concurrent work, this memory must preserve provenance and causal dependencies, distinguish parallel branches, and identify records superseded by later updates. Existing memory benchmarks focus on recall and retrieval, while multi-agent benchmarks measure overall team performance; neither isolates whether a shared-memory backend provides valid evidence during overlapping work. We introduce COLLABMEMBENCH, built from timestamped GitHub events collected through GH Archive from public open-source repositories. It organizes issues, pull requests, reviews, commits, and releases into controlled Sequential, Parallel, and Entangled workflows with non-blocking event release and distinct private agent histories. Holding the agent team, model, tools, schedule, and resource budgets fixed, we compare replaceable memory backends on answer quality, complete-case success, evidence sufficiency, and token and time costs. Across six external memory systems and a long-context baseline, even the best-performing memory system reaches only 47.6% Macro QA Score, 20.3% Complete Case Rate, and 37.4% Evidence Sufficiency Rate. Performance varies across workflow structures, with recurring failures to recover prerequisite evidence, separate branches, and reject superseded records. These results show that relevance-oriented retrieval alone does not reliably support shared memory for asynchronous multi-agent coordination. COLLABMEMBENCH lays the foundation for developing shared-memory systems that enable asynchronous agent teams to coordinate reliably as their work evolves. Code and benchmark artifacts are available at `https://anonymous.4open.science/r/CollabMemBench-475D`.

## 1 INTRODUCTION

Large language model (LLM) agents increasingly work in teams on long-running tasks in software engineering, customer support, scientific discovery, and infrastructure operations (Qian et al., 2024; Jiang et al., 2025; Schmidgall et al., 2025; Yu et al., 2025). Such teams are naturally asynchronous: agents begin from different observations, work concurrently, finish at different times, and hand off incomplete tasks without a global synchronization barrier (Gonzalez-Pumariega et al., 2025; Zhang et al., 2026b). Consequently, an agent's private history may lack the decisions, evidence, and updated constraints from others needed for its next action. This setting makes **shared memory** a coordination substrate rather than merely a larger store of past conversations (Nii, 1986; Zhang et al., 2026a). Its role is to retain information and make information produced by one agent usable by others. To support subsequent work, shared memory must preserve where a record came from, what earlier decisions it depends on, which concurrent branch it belongs to, and whether it remains valid after later updates (Lamport, 2019; Ren et al., 2026). These requirements are coupled. Retrieving an earlier decision without the update that superseded it can be harmful; retaining only the latest state can erase the rationale needed to interpret that state; and mixing records from unrelated concurrent tasks can create an answer that is individually plausible but globally inconsistent.

Open-source software development provides a concrete example and a source of repository-grounded tasks (Jimenez et al., 2024). As shown in Figure 1, a dependency agent upgrades a



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.

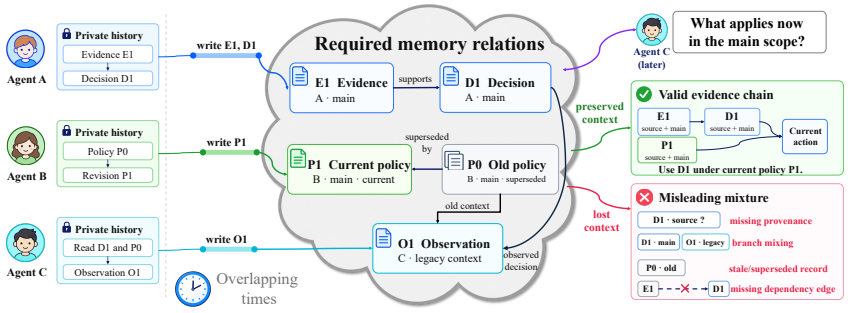


Figure 1: Shared memory for asynchronous collaboration. Agents with private histories write to memory as work overlaps. Decisions require valid evidence chains with provenance, branch scope, and dependencies; relevance-only retrieval may return an incomplete or inconsistent mixture.

vulnerable library; later, a release agent changes the supported Python versions while a CI agent encounters a failure on an older version. The CI agent must connect the security rationale with the current release policy before deciding whether to revert the upgrade or update the test matrix. A rollback based on the failure alone could restore the vulnerability, while overlooking the policy change could leave the CI issue unresolved. Although the records appear together in shared memory, the decision still depends on their provenance, causal dependencies, and current validity, echoing difficulties with attribution and evolving decisions in multi-party memory (Hu et al., 2026). In human software collaboration, concurrent updates do not become consistent automatically: Git and its surrounding protocols help collaborators track versions and provenance, expose conflicts, and reconcile concurrent changes (Bird et al., 2009; Kaki, 2019). Asynchronous agents face a similar challenge when they contribute decisions, observations, and intermediate results to shared memory: it must separate parallel contexts, preserve dependencies, and identify conflicting or superseded records. We use repository workflows to turn this coordination requirement into a controlled setting, testing whether existing memory systems exhibit these capabilities and how their absence affects downstream evidence and decisions.

Existing memory benchmarks study recall, updates, and reuse across sessions (Maharana et al., 2024; Wu et al., 2024; Hu et al., 2026; He et al.), while multi-agent benchmarks assess communication and team outcomes (Zhu et al., 2025; Yu et al., 2026). Dynamic benchmarks add overlapping tasks and changing environments (Zhang et al., 2026b; Froger et al., 2026; Meng et al., 2026); GateMem further studies governance of a shared memory pool (Ren et al., 2026). Yet these evaluations do not jointly isolate a shared-memory backend and test whether cross-agent evidence remains valid as concurrent work unfolds. We argue that evaluating memory for collaborative multi-agent systems under asynchronous execution requires two complementary capabilities: *cross-agent information continuity*, recovering decisions, evidence, and constraints produced by collaborators but absent from the receiving agent's private history; and *concurrent consistency*, keeping evidence contributed by parallel collaborators correctly scoped and current as related branches evolve. Together, these capabilities determine whether shared memory supports reliable handoffs and decisions across the team. They also make asynchronous collaboration observable in the evaluation: events are released without waiting for earlier tasks to finish, agents maintain private histories, and shared-memory reads occur while multiple workflows remain active.

We introduce CollabMemBench, which uses timestamped GitHub events collected through GH Archive from public open-source repositories to construct controlled asynchronous workflows from issues, pull requests, reviews, commits, and releases. Three structures test distinct memory demands: **Sequential** tests causal evidence across agents and time; **Parallel** tests separation of interleaved contexts that touch shared resources; and **Entangled** tests validity and dependencies as related branches interact. Agents retain private histories while events are released without waiting for ongoing work. At checkpoints, a receiving agent uses shared memory to answer questions about earlier decisions, dependencies, superseded records, and next steps. We vary only the memory backend, holding the agent team, model, tools, events, schedule, and context and retrieval budgets fixed. An evaluator-only manifest specifies reference answers and valid evidence chains, allowing us to measure answer quality, complete-case success, evidence sufficiency, and token and time costs. We further assess memory-level behavior through Current-Record Precision (CRP) and Dependency-Chain Recall (DCR), measuring the validity of retrieved decision records and the recovery of required records and dependency links. Across six widely used external memory systems and a long-context baseline, GraphRAG leads overall with 47.6% MQS, 20.3% complete-case success, and 37.4% ESR,

while Mem0 follows closely in MQS at 46.5% but incurs substantially higher latency. Performance varies across Sequential, Parallel, and Entangled workflows. Serializing parallel branches helps some systems, while greater causal depth reduces evidence recovery for others. Case studies reveal missing prerequisite evidence and confusion between current, conflicting, or superseded records. Together, these findings suggest that relevance-oriented retrieval alone is insufficient and motivate explicit handling of provenance, branch scope, validity, and reconciliation.

Our contributions are:

- We formulate memory for collaborative agent teams under asynchronous execution as a shared coordination problem requiring both cross-agent information continuity and concurrent consistency, and operationalize the relevant failure modes through provenance, causal dependency, context separation, supersession, and reconciliation.
- We introduce COLLABMEMBENCH, a benchmark derived from public open-source development histories that constructs Sequential, Parallel, and Entangled workflows while preserving controlled causal order and asynchronous overlap.
- We provide an evidence-grounded evaluation protocol that isolates replaceable shared-memory backends and measures answer quality, complete-case success, evidence sufficiency, and resource cost under fixed agent and execution conditions.
- We evaluate representative memory systems and show that their ability to retrieve relevant information does not yet ensure reliable support for concurrent multi-agent work, motivating concurrency-aware designs for shared agent memory.

## 2 RELATED WORK

**Agent memory: persistence and retrieval.** Agent-memory systems retain and reuse experience through retrieval, compression, and structured representations (Chhikara et al., 2025; Edge et al., 2024; Zhang et al., 2026a). Existing benchmarks cover long-conversation recall and temporal reasoning, factual and reflective memory, trajectory-level inference and updating, cross-session task use, and evolving decisions in multi-party dialogue (Maharana et al., 2024; Wu et al., 2024; Tan et al., 2025; Zhao et al.; He et al.; Hu et al., 2026). They therefore go beyond factual recall, but do not jointly evaluate a replaceable shared memory used by agents with private histories during overlapping work.

**Shared memory and multi-agent coordination.** Shared workspaces, formalized by blackboard systems, let agents contribute to a common state (Nii, 1986). MultiAgentBench instead compares communication and planning strategies, EntCollabBench evaluates handoffs across permission-isolated roles, and AgentWebBench studies evidence acquisition from content agents (Zhu et al., 2025; Yu et al., 2026; Zhong et al., 2026). Their primary experimental object is the team or coordination policy rather than its memory backend. GateMem directly evaluates a multi-principal shared pool, but its episodes are ingested and queried in temporal order rather than while multiple workflow branches remain active (Ren et al., 2026). This distinction motivates isolating the shared-memory backend. Causal ordering and version control further motivate our treatment of dependencies, branches, and superseded information (Lamport, 2019; Kaki, 2019).

**Dynamic and asynchronous benchmarks.** Dynamic benchmarks test delayed or parallel actions, planning disruptions, event-driven environments, and long-running external state (Gonzalez-Pumariega et al., 2025; Zhang et al., 2026b; Geng & Chang, 2026; Froger et al., 2026; Meng et al., 2026; Sugiura et al., 2026). They make temporal dynamics observable, but do not isolate shared memory: success may come from communication, repeated observation, or recomputation. The missing evaluation combines private histories, observable memory operations, and checkpoint-level scoring of whether retrieved evidence remains valid and supports a downstream answer.

Taken together, these studies examine what memory retains, how teams coordinate, and how agents handle overlapping work. As Table 1 shows, none of the compared benchmarks brings these threads together to evaluate both *cross-agent information continuity* across handoffs and *concurrent consistency* as parallel branches change which evidence remains valid. Our COLLABMEMBENCH constructs controlled asynchronous workflows from timestamped events and linked artifacts in real multi-participant open-source development histories. These workflows capture provenance, causal dependencies, and resource overlap, allowing us to evaluate whether shared-memory systems maintain cross-agent information continuity and concurrent consistency.

Table 1: Comparison of representative benchmarks. ✓: explicit design or evaluation target; △: partially covered; –: not an explicit target.

| Benchmark | Non-block. overlap | Private hist. | Shared-mem. target | Cross-agent prov. | Supers./ conflicts | Branch recon. | Evidence eval. | Cost |
|---|---|---|---|---|---|---|---|---|
| AMA-Bench (Zhao et al.) | – | – | △ | – | △ | – | △ | ✓ |
| MemoryArena (He et al.) | – | – | △ | – | – | – | – | ✓ |
| EverMemBench (Hu et al., 2026) | – | – | △ | △ | ✓ | △ | △ | △ |
| GateMem (Ren et al., 2026) | – | – | ✓ | △ | ✓ | – | △ | ✓ |
| MultiAgentBench (Zhu et al., 2025) | – | ✓ | – | – | – | – | – | ✓ |
| EntCollabBench (Yu et al., 2026) | – | ✓ | – | △ | – | – | ✓ | ✓ |
| AgentWebBench (Zhong et al., 2026) | – | – | – | ✓ | △ | – | ✓ | △ |
| Gaia2 (Froger et al., 2026) | ✓ | △ | – | – | △ | – | – | ✓ |
| ParaCook (Zhang et al., 2026b) | ✓ | – | – | – | – | – | – | ✓ |
| **COLLABMEMBENCH** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

# 3 THE COLLABMEMBENCH BENCHMARK

## 3.1 PROBLEM FORMULATION

Let $\Pi = \{\pi_k\}_{k=1}^{n}$ denote a fixed team of role-specialized agents and $M$ the shared-memory backend under evaluation. The dataset $\mathcal{D} = \{(\mathcal{W}_i, \mathcal{G}_i)\}_{i=1}^{|\mathcal{D}|}$ pairs each asynchronous workflow $\mathcal{W}_i = (X_{i,0}, \mathcal{E}_i)$ with an evaluator-only manifest $\mathcal{G}_i$. Here, $X_{i,0}$ is the initial state and $\mathcal{E}_i = (e_{ij})_{j=1}^{|\mathcal{E}_i|}$ contains events $e_{ij} = (t_{ij}, x_{ij})$ with release time $t_{ij}$ and visible content $x_{ij}$. The hidden manifest specifies the questions, reference answers, record dependencies, validity conditions, and supporting evidence required at each checkpoint. A valid evidence chain contains all required prerequisites, respects workflow scope, and distinguishes current records from conflicting or superseded ones. At time $t$, each agent retains a private history $H_k(t)$ and accesses $M$; a read returns the ranked view $V_k(t; M)$. The runner logs each memory operation as $o_\ell = (k_\ell, a_\ell, z_\ell, c_\ell, p_\ell, b_\ell, \tau_\ell), \quad a_\ell \in \{\mathsf{read}, \mathsf{write}\}$, recording the agent, read/write action, key or entity, content, provenance, workflow scope or branch, and logical time, respectively.

We test two properties: ① **Cross-agent information continuity**, requiring $V_k(t; M)$ to recover a complete valid chain containing prerequisites produced by another agent and absent from $H_k(t)$; and ② **Concurrent consistency**, requiring the view to distinguish independent scopes and preserve the dependencies and validity of related branches after supersession or reconciliation. Let $\widehat{\mathcal{A}}_i(M)$ denote the answers produced at the evaluation checkpoints, and let $s_i(M) = \mu_i(\mathcal{G}_i, \widehat{\mathcal{A}}_i(M))$ denote their evaluation score. The overall performance is $S(M) = |\mathcal{D}|^{-1} \sum_{i=1}^{|\mathcal{D}|} s_i(M)$. The benchmark observes the two properties through the retrieved views $V_k(t; M)$ and the resulting checkpoint answers; the backend's internal state remains opaque. We compare memory backends while holding the agent team, workflow dataset, models, tools, scheduling policy, and execution budgets fixed.

## 3.2 ASYNCHRONOUS WORKFLOW CONSTRUCTION

Public repository histories provide independent actors, naturally asynchronous and timestamped events, explicit provenance, and linked artifacts that expose causal dependencies, supersession, and branch interactions. Their availability supports auditable and reproducible construction. We construct 300 cases from 39,063 GH Archive events and payloads across seven permissively licensed public repositories collected from July to December 2025. Pinned repository clones provide replay states; released workflows replace real identities and dates with synthetic roles and logical times.

Our benchmark evaluates whether shared memory can support asynchronous multi-agent workflows along two dimensions: preserving causal continuity when related work spans agents and time, and maintaining consistency when multiple workflows progress concurrently. We instantiate these requirements with three workflow structures: ① **Sequential**, a long dependency chain targeting long-range causal continuity; ② **Parallel**, causally independent events that overlap in shared development resources and are replayed concurrently to test context separation; and ③ **Entangled**, a connected, branched dependency subgraph with possible joins or cross-branch dependencies that tests consistency across related workflows.

We consolidate collected issues, pull requests, reviews, commits, and releases concerning the same development change into a source event. Each source event records its natural-language request, affected files or components, repository evidence, and original temporal metadata. We assess source-

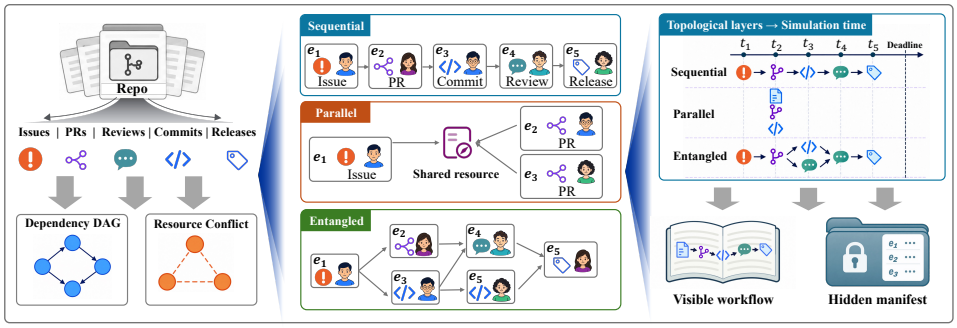


Figure 2: Workflow-construction pipeline. Repository artifacts are consolidated into source events and organized as a dependency DAG and resource-overlap graph. These structures generate Sequential, Parallel, and Entangled workflows; topological layers define simulation times, yielding an agent-visible workflow $\mathcal{W}$ and an evaluator-only manifest $\mathcal{G}$.

event pairs using their descriptions, affected files, timestamps, repository links, and observed state transitions. The annotation taxonomy distinguishes: *resource overlap* without an implied semantic conflict; *competing updates* to the same state; *supersession*; *contradictory observations* of the same revision or state; *cross-branch dependencies*; and explicit *join/reconciliation* events. A relation is retained only when its shared entity or resource and its supporting artifacts can be identified; directed relations must also agree with temporal order. A dependency $e_a \to e_b$ means that handling $e_b$ requires a decision, artifact, or state established by $e_a$. Temporally valid dependencies form a DAG used for Sequential and Entangled construction, while resource-overlap edges remain in a separate undirected graph used for Parallel construction. The remaining relation labels determine record validity and acceptable evidence chains in the hidden manifest.

We construct the three workflow types as follows. For **Sequential** cases, we enumerate paths containing at least five events in the dependency DAG and greedily retain a path only if it contributes at least two events absent from previously selected paths, avoiding near-duplicates in braided DAGs. For **Entangled** cases, we traverse the dependency DAG in breadth-first order using a sliding window; a window becomes a case when it contains six to nine nodes and its induced subgraph is connected (ignoring edge direction) with a branching point, while older nodes are discarded as needed to keep the window within nine nodes. Finally, for **Parallel** cases, we exclude causally dependent event pairs and take each connected component of the remaining resource-overlap graph as one case without traversing the dependency DAG.

After extraction, we organize each dependency graph into topological layers to construct its replay schedule rather than reproducing the concurrency or time intervals in the source history. Events in the same layer receive the same simulation time and are injected simultaneously, whereas events in deeper layers are released later. Timestamps support relation assessment and temporal validation but do not determine the replay schedule. The resulting workflows are controlled asynchronous reconstructions from real histories: their artifacts, actors, provenance, and causal evidence are authentic, while concurrency, compressed simulation times, and response deadlines are synthesized with causal order preserved. We map these layers to compressed simulation times, assign response deadlines, and interleave unused real events as distractors, whose identities are recorded only in the hidden manifest. Before inclusion, each case must pass structural, temporal, graph-consistency, and repository-replay checks, and its visible timeline and hidden manifest are stored separately.

### 3.3 Asynchronous Multi-Agent Evaluation Protocol

For each workflow, the runner releases the event timeline to agents in a live multi-agent environment; it does not replay prewritten memory records. It creates an isolated sandbox at pinned state $X_0$, launches the fixed agent team $\Pi$, and attaches the memory backend $M$ through a common adapter. Held-out tests and evaluator-only artifacts remain outside the agent-accessible workspace.

Figure 3 summarizes the runtime data flow, the separation between agent-visible inputs and evaluator-only artifacts, and the question-level evaluation of retrieved evidence and answers. A global simulation clock releases each event $e_j$ at $t_j$ without waiting for previously released events to finish or imposing a global synchronization barrier, allowing multiple tasks to remain active simultaneously. Agents receive role-specific observations and interact with the environment through a common observe–decide–act loop; a later observation may therefore supersede a record before an-

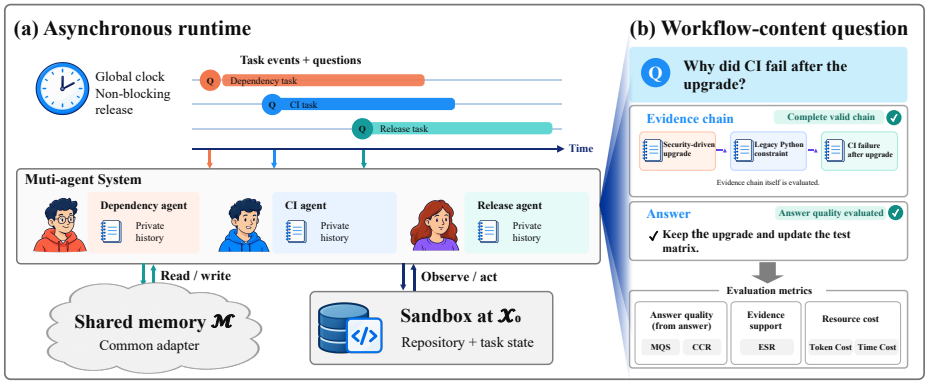


Figure 3: Asynchronous multi-agent evaluation protocol. A global clock non-blockingly releases events and questions to agents with private histories. Agents interact through a sandbox and a replaceable shared-memory backend accessed through a common adapter. A manifest evaluates retrieved evidence, answers, and resource costs without being exposed to agents or memory systems.

other active task consumes it. Each agent retains a private local history; information crossing agent boundaries must pass through $M$ or an explicitly logged communication action. Agents generate memory writes from their observations and actions, and all reads and writes target the same backend through the common adapter. The adapter gives every backend the same agent-visible content and metadata, while runner-side validity, dependency, and reference-evidence annotations remain evaluator-only. The release schedule is fixed, while memory operations are logged in execution order; the protocol therefore evaluates consistency under interleaved asynchronous histories rather than low-level write races. Backends return retrieved records rather than a required resolved state, so the receiving agent must interpret any conflicting evidence when producing its answer.

At designated task points, the timeline injects workflow-content questions that are routed to the corresponding agents. The runner records agent actions, memory reads and writes, retrieved records, token and time costs, and question answers. Evaluation questions and answers are logged but are not written back to $M$. The answers serve as the primary criterion for evaluating the memory backend: for each question, the evaluator checks both answer correctness and whether the retrieved information contains a complete valid evidence chain specified by the hidden manifest.

### 3.4 Evaluation Metrics

For case $i$ with questions $\mathcal{Q}_i$, let $r_{ij} \in [0,1]$ score answer agreement with the evaluator-only reference and $\bar{r}_i = |\mathcal{Q}_i|^{-1} \sum_{j \in \mathcal{Q}_i} r_{ij}$ be the case mean. The *Macro QA Score* is $\mathrm{MQS} = |\mathcal{D}|^{-1} \sum_i \bar{r}_i$, the operational form of $S(M)$, while the *Complete Case Rate* is $\mathrm{CCR} = |\mathcal{D}|^{-1} \sum_i \mathbb{1}[\min_j r_{ij} = 1]$, the fraction of entirely correct cases.

We evaluate evidence separately by setting $e_{ij} = 1$ when retrieved records contain a complete valid chain for question $j$ and defining the *Evidence Sufficiency Rate* as $\mathrm{ESR} = |\mathcal{D}|^{-1} \sum_i |\mathcal{Q}_i|^{-1} \sum_j e_{ij}$. Two memory-level diagnostics complement ESR: *Current-Record Precision* (CRP) measures the fraction of retrieved decision records current at the checkpoint, and *Dependency-Chain Recall* (DCR) measures recovery of required records and dependency links. ESR, CRP, and DCR require an explicit retrieved record set and are not reported for Long Context, which directly uses the interaction history. Finally, *Token Cost* and *Time Cost* report total tokens and elapsed wall-clock time per evaluation question. We report quality and resource measures separately.

## 4 Experiments

### 4.1 Experimental Setup

**Datasets.** The comparison contains 300 cases (100 each of Sequential, Parallel, and Entangled) and 12,165 timeline entries. Distractors account for 69.3% of non-probe records, including supersession decoys; gold annotations remain evaluator-only. We construct variants of the main evaluation dataset for three diagnostic experiments: Oracle A supplies golden evidence, serialization reorders selected Parallel cases, and depth extends selected Sequential cases from Hop2 to Hop5 while holding other content fixed.

**Methods.** We evaluate representative systems from several widely used memory categories. Long Context retains the unprocessed interaction history; BM25 (Robertson & Zaragoza, 2009) and Dense RAG (Lewis et al., 2020) provide lexical and dense retrieval; Rolling Summary compresses prior context; Mem0 (Chhikara et al., 2025) provides managed long-term memory; and GraphRAG (Edge et al., 2024) and G-Memory (Zhang et al., 2026a) represent graph-structured memory. We additionally include a Versioned Event Log, a deterministic append-only baseline that maintains provenance, workflow scope, dependencies, and explicit supersession relations before applying lexical retrieval. Oracle A is used only as an upper-bound reference for evidence access.

**Model and protocol.** The main and diagnostic experiments use `qwen3.8-max` with thinking disabled and run in the benchmark framework's SWE sandbox, with QA probes about repository history. Cross-model validation additionally uses `qwen3.8-flash` as a lightweight comparison under the same protocol. Across memory conditions, we hold fixed the agent team, workflow inputs and question checkpoints, tools, scheduling policy, and execution budgets.

**Metrics.** We report MQS, CCR, ESR, CRP, and DCR as percentages (Section 3.4), together with tokens and elapsed time per query; Appendix A contains full diagnostics. Overall is the arithmetic mean across the three workflow families. Oracle A uses the same 0/1/2 answer scoring but has no retrieval metrics. Higher scores and lower costs are better; percentage differences are in points (pp).

### 4.2 Main Results

Table 2: Main results of `qwen3.8-max` on CollabMemBench. Scores are percentages; boldface and underlining mark the best and second-best per setting. Higher scores and lower costs are better; – denotes not reported.

| Setting | Method | Effectiveness | | | Memory diagnostics | | Efficiency | |
|---|---|---|---|---|---|---|---|---|
| | | MQS (%) | CCR (%) | ESR (%) | CRP (%) | DCR (%) | Tokens/q | Time (s/q) |
| Sequential | Long Context | 21.6 | 0 | – | – | – | 7,353 | **45.4** |
| | BM25 | 35.8 | 3 | 45.9 | 26.9 | 92.9 | 6,930 | 52.6 |
| | Dense RAG | 38.6 | 3 | 31.5 | 27.7 | 63.7 | 7,078 | 57.4 |
| | Rolling Summary | 34.0 | 1 | 27.6 | 21.1 | 62.6 | 10,285 | 132.6 |
| | Mem0 | 35.9 | 5 | 37.6 | 13.0 | **95.4** | **6,782** | 411.4 |
| | GraphRAG | **56.5** | **29** | **60.9** | 23.3 | 77.9 | 10,776 | 190.3 |
| | G-Memory | 32.2 | 6 | 21.9 | **31.1** | 63.1 | 7,864 | 113.2 |
| Parallel | Long Context | 2.7 | 0 | – | – | – | 7,754 | **48.4** |
| | BM25 | **56.5** | 18 | **37.3** | 74.6 | 76.4 | 7,865 | 58.7 |
| | Dense RAG | 53.2 | 12 | 33.6 | 62.7 | 66.9 | 7,545 | 61.2 |
| | Rolling Summary | 43.7 | **24** | 26.0 | **83.1** | 66.8 | 11,138 | 168.2 |
| | Mem0 | 55.6 | 17 | 24.1 | 74.0 | **84.6** | **7,069** | 427.0 |
| | GraphRAG | 50.8 | 22 | 32.2 | 66.5 | 74.6 | 16,740 | 243.2 |
| | G-Memory | 41.0 | 17 | 24.2 | 61.7 | 67.5 | 8,296 | 147.3 |
| Entangled | Long Context | 16.3 | 0 | – | – | – | 7,682 | **49.5** |
| | BM25 | 33.2 | 10 | **22.6** | 83.8 | **87.4** | 7,666 | 55.7 |
| | Dense RAG | 30.7 | 6 | 20.2 | 78.3 | 58.2 | 7,354 | 59.0 |
| | Rolling Summary | 28.3 | 12 | 11.7 | 75.8 | 75.8 | 11,339 | 185.6 |
| | Mem0 | **47.9** | **19** | 14.6 | **89.5** | 77.9 | **7,015** | 395.2 |
| | GraphRAG | 35.4 | 10 | 19.2 | 86.6 | 50.0 | 17,062 | 248.7 |
| | G-Memory | 26.2 | 6 | 13.9 | 81.3 | 58.9 | 8,174 | 141.3 |
| **Overall** | Long Context | 13.5 | 0.0 | – | – | – | 7,596 | **47.8** |
| | BM25 | 41.8 | 10.3 | 35.3 | **61.8** | 85.6 | 7,487 | 55.7 |
| | Dense RAG | 40.8 | 7.0 | 28.4 | 56.2 | 62.9 | 7,326 | 59.2 |
| | Rolling Summary | 35.3 | 12.3 | 21.8 | 60.0 | 68.4 | 10,921 | 162.1 |
| | Mem0 | 46.5 | 13.7 | 25.4 | 58.8 | **86.0** | **6,955** | 411.2 |
| | GraphRAG | **47.6** | **20.3** | **37.4** | 58.8 | 67.5 | 14,859 | 227.4 |
| | G-Memory | 33.1 | 9.7 | 20.0 | 58.0 | 63.2 | 8,111 | 133.9 |

**Overall performance and information continuity.** GraphRAG achieves the strongest Overall downstream results, with 47.6% MQS, 20.3% CCR, and 37.4% ESR, yet no method completes even one quarter of cases correctly. Memory diagnostics yield a different ranking: BM25 has the highest Overall CRP (61.8%), while Mem0 has the highest DCR (86.0%). On Sequential workflows, GraphRAG leads MQS, CCR, and ESR. The large gap between MQS and CCR across methods further shows that answering individual probes does not ensure consistent success throughout a workflow.

**Evidence under concurrent workflows.** No method dominates: on Parallel workflows, BM25 leads MQS and ESR, Rolling Summary leads CCR and CRP, and Mem0 leads DCR. On Entangled workflows, Mem0 leads MQS, CCR, and CRP but obtains 14.6% ESR, whereas BM25 leads ESR (22.6%) and DCR (87.4%). This mismatch shows that higher answer scores need not coincide with recovery of a complete valid evidence chain. Likewise, GraphRAG reaches 86.6% CRP but only

50.0% DCR, showing that current records alone do not provide the complete dependency chain. All external-memory methods have lower MQS and ESR on Entangled than on Parallel workflows, suggesting that interacting branches are more difficult; however, this aggregate comparison alone cannot attribute the decline to supersession, branch contamination, or missing dependencies.

**Quality–cost trade-off.** Compared with BM25, GraphRAG improves Overall MQS by 5.8 pp and CCR by 10.0 pp, but uses about twice as many tokens and 4.1 times the latency. Mem0 approaches GraphRAG in MQS while requiring 411.2 seconds per query, whereas BM25 remains competitive in ESR at 55.7 seconds per query. Thus, higher memory overhead does not yield a proportional improvement in downstream quality.

### 4.3 Diagnostic Experiments

**Oracle evidence.** To isolate errors that can remain after evidence access, Oracle A bypasses memory retrieval and gives the agent the minimal sufficient golden evidence for each question while keeping downstream reasoning and judging unchanged. It achieves 86.8% MQS (Table 3), exceeding the strongest retrieved- evidence result by 39.2 pp. This gap indicates substantial headroom from evidence access, while Oracle A's score below 100% shows that downstream reasoning errors persist even with sufficient evidence. Retrieval-based ESR is not reported for Oracle A.

Table 3: Performance with oracle evidence.

| Condition | Evidence | MQS (%) | CCR(%) |
|---|---|---|---|
| Oracle A | Golden | 86.8% | 66.7% |

**Increasing causal depth.** We extend selected Sequential cases from Hop2 to Hop5 while keeping noise and superseding updates fixed. Figure 4 shows that longer chains reduce answer quality across memory systems. From Hop2 to Hop5, MQS falls for every method and CCR approaches zero for most; GraphRAG remains strongest (56.5% to 46.5% MQS; 29.0% to 16.0% CCR), but its token cost grows from 10,776 to 16,068, while Mem0 and G-Memory suffer the largest MQS drops at nearly constant cost. This contrast shows that explicit relational tracing preserves longer causal chains but expands GraphRAG's retrieved context, whereas the other systems keep token use comparatively stable but recover less of the required information as intermediate dependencies accumulate.

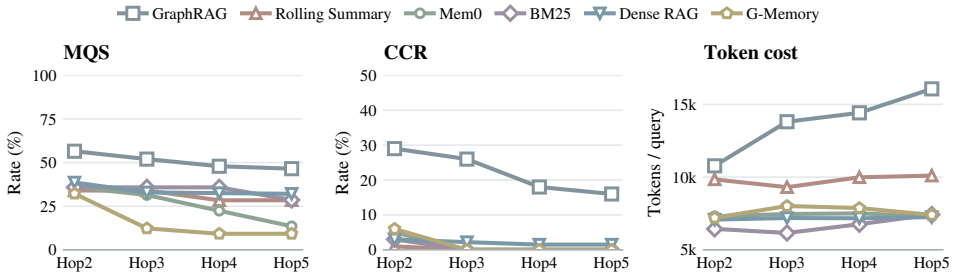


Figure 4: Effect of causal depth on MQS, CCR, and token cost across memory systems.

**Parallel versus serial delivery.** We serialize matched Parallel cases while preserving their facts, questions, and reference answers. Serialization benefits most methods on at least some metrics, although the effects are uneven: BM25 and Dense RAG gain 20.0 and 5.8 MQS points, whereas GraphRAG and Rolling Summary lose 11.0 and 5.3 points; Mem0 and G-Memory show comparatively smaller changes across answer and evidence metrics. Serialization can reduce interference among interleaved records, but it can also reweight evidence in summaries and graph states, producing method-dependent changes. This shows that collaboration order is part of the memory problem: asynchronous teams need shared memory that preserves branch and dependency structure independently of arrival order. The complete six-system comparison is shown in Figure 5 in Appendix A.

### 4.4 Cross-Model Validation

To examine whether our findings depend on the capability of the underlying agent model, we repeat the evaluation with the lightweight `qwen3.8-flash` model and compare it with `qwen3.8-max`, which is used in the main experiments. Table 4 shows that Flash reduces quality for every method, but not equally: GraphRAG's ESR and MQS drop by 23.0 and 19.1 pp, compared with 5.7 and 7.9 pp for BM25. BM25 retrieves records directly through lexical identifier overlap, whereas GraphRAG relies on the model to interpret graph relations and summaries before composing evidence, so errors

accumulate across more model-dependent stages (Robertson & Zaragoza, 2009; Edge et al., 2024). This makes lexical retrieval more stable under Flash while GraphRAG benefits more from Max; workflow-level results are reported in Appendix C.

Table 4: Overall cross-model results. Bold and underline mark the best and second-best values; lower costs are better.

| Method | Max | | | | | Flash | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MQS | CCR | ESR | Tok. | Time | MQS | CCR | ESR | Tok. | Time |
| BM25 | 41.8 | 10.3 | <u>35.3</u> | 7,487 | **55.7** | **33.9** | 4.7 | **29.6** | <u>4,659</u> | **136.9** |
| Dense RAG | 40.8 | 7.0 | 28.4 | <u>7,326</u> | <u>59.2</u> | 26.2 | 2.7 | 8.8 | **4,431** | <u>137.5</u> |
| Rolling Summary | 35.3 | 12.3 | 21.8 | 10,921 | 162.1 | 26.4 | **8.3** | <u>19.7</u> | 7,494 | 213.4 |
| Mem0 | <u>46.5</u> | <u>13.7</u> | 25.4 | **6,955** | 411.2 | **33.9** | <u>8.0</u> | 18.1 | 4,761 | 793.0 |
| GraphRAG | **47.6** | **20.3** | **37.4** | 14,859 | 227.4 | <u>28.5</u> | 3.7 | 14.4 | 10,388 | 322.9 |
| G-Memory | 33.1 | 9.7 | 20.0 | 8,111 | 133.9 | 18.0 | 2.0 | 8.0 | 4,922 | 204.7 |

### 4.5 Case Studies

We present two cases illustrating complementary memory failures. In `numpy/stubtest`, memory finds the decision but omits its passing validation, causing the agent to choose `refresh_validation` instead of `reuse_validation`. This demonstrates a failure of cross-agent information continuity. In `pydantic/urlschem`, contradictory CI records lose their links to revision status and merge selection, so an overturned build is treated as current. This demonstrates a failure of concurrent consistency. Oracle A answers both probes with complete provenance-linked evidence, localizing the errors to the retrieved memory view; Appendix B provides the full timelines and evidence chains.

## 5 Discussion and Limitations

Our results highlight the importance of evaluating shared memory through both downstream answers and their supporting evidence. Differences between answer quality and evidence sufficiency, together with the observed dependency and supersession failures, show that retrieving relevant records does not always provide a complete, currently valid basis for a decision. Evaluating these aspects jointly helps characterize how memory supports subsequent work as information passes between agents and related branches evolve.

GitHub histories provide an observable and auditable setting for studying these requirements, with timestamps, provenance, and linked artifacts that make dependencies and validity changes explicit. Controlled schedules preserve causal order and enable matched comparisons under reproducible event release. The current evaluation focuses on open-source development with reconstructed concurrency; extending it to less structured domains and naturally timed workflows would help assess how the observed patterns transfer.

Our consistency claims concern retrieved evidence and downstream decisions, rather than formal guarantees on the backend's internal state. The common adapter enables comparison under standardized inputs and budgets, while native integrations may expose additional system-specific capabilities. Dependency and conflict annotations are grounded in identifiable artifacts and checked for structural and temporal consistency; human agreement analysis would provide additional validation of judgments involved in these relations. Taken together, our findings motivate shared-memory designs that expose provenance, version and scope, causal dependencies, and validity state, alongside explicit support for identifying conflicts and tracking their reconciliation.

## 6 Conclusion

We introduced CollabMemBench, which isolates shared-memory backends in controlled asynchronous workflows derived from public open-source histories. Across representative systems, even the strongest completes only 20.3% of cases and often misses prerequisite evidence or confuses conflicting and superseded records. These results show that multi-agent memory must be evaluated by whether it maintains a coherent shared account of concurrent work, not only by what it retains. By making these failures measurable and reproducible, CollabMemBench lays the foundation for shared-memory systems that enable reliable asynchronous coordination.

## AI USE STATEMENT

Generative AI tools were used to assist with and polish the writing and to support literature retrieval and discovery, including identifying potentially relevant related work. The authors reviewed and verified the resulting text, references, and claims and take full responsibility for the paper's content.

## REFERENCES

Christian Bird, Peter C Rigby, Earl T Barr, David J Hamilton, Daniel M German, and Prem Devanbu. The promises and perils of mining git. In *2009 6th IEEE International Working Conference on Mining Software Repositories*, pp. 1–10. IEEE, 2009.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, et al. Gaia2: Benchmarking llm agents on dynamic and asynchronous environments. In *International Conference on Learning Representations*, volume 2026, pp. 119758–119789, 2026.

Longling Geng and Edward Y Chang. Realm-bench: A benchmark for evaluating multi-agent systems on real-world, dynamic planning and scheduling tasks. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 2652–2662, 2026.

Gonzalo Gonzalez-Pumariega, Leong Yean, Neha Sunkara, and Sanjiban Choudhury. Robotouille: An asynchronous planning benchmark for llm agents. In *International Conference on Learning Representations*, volume 2025, pp. 83757–83792, 2025.

Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, et al. Benchmarking agent memory in interdependent multi-session agentic tasks. In *Forty-third International Conference on Machine Learning*.

Chuanrui Hu, Tong Li, Xingze Gao, Hongda Chen, Yi Bai, Dannong Xu, Tianwei Lin, Xiaohong Li, Yunyun Han, Jian Pei, et al. Evaluating long-horizon memory for multi-party collaborative dialogues. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 9082–9090, 2026.

Xinyi Jiang, Tianyi Hu, Yuheng Qin, Guoming Wang, Zhou Huan, Kehan Chen, Gang Huang, Rongxing Lu, and Siliang Tang. Chatmap: Mining human thought processes for customer service chatbots via multi-agent collaboration. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11927–11947, 2025.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Gowtham Kaki. Version control is for your data too. *3rd Summit on Advances in Programming Languages (SNAPL 2019)*, 2019.

Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. In *Concurrency: the Works of Leslie Lamport*, pp. 179–196. 2019.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

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

Fanqing Meng, Lingxiao Du, Zijian Wu, Guanzheng Chen, Xiangyan Liu, Jiaqi Liao, Chonghe Jiang, Zhenglin Wan, Jiawei Gu, Pengfei Zhou, et al. Clawmark: A living-world benchmark for multi-turn, multi-day, multimodal coworker agents. *arXiv preprint arXiv:2604.23781*, 2026.

H Penny Nii. Part one: The blackboard model of problem solving and the evolution of blackboard architectures. *AI magazine*, 7(2):38–53, 1986.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. 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, 2024.

Zhe Ren, Yibo Yang, Yimeng Chen, Zijun Zhao, Benshuo Fu, Zhihao Shu, Bingjie Zhang, Yangyang Xu, Dandan Guo, and Shuicheng Yan. Gatemem: Benchmarking memory governance in multi-principal shared-memory agents. *arXiv preprint arXiv:2606.18829*, 2026.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and trends® in information retrieval*, 4(1-2):1–174, 2009.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using llm agents as research assistants. *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043, 2025.

Issa Sugiura, Daichi Hattori, Kazuo Araragi, Keita Ogawa, Shota Onose, Taro Makino, Teppei Usuki, and Takashi Ishida. Coffeebench: Benchmarking long-horizon llm agents in heterogeneous multi-agent economies. *arXiv preprint arXiv:2606.16613*, 2026.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. Membench: Towards more comprehensive evaluation on the memory of llm-based agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19336–19352, 2025.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Tao Yu, Hao Wang, Changyu Li, Shenghua Chai, Minghui Zhang, Zhongtian Luo, Yuxuan Zhou, Haopeng Jin, Zhaolu Kang, Jiabing Yang, et al. Beyond the all-in-one agent: Benchmarking role-specialized multi-agent collaboration in enterprise workflows. *arXiv preprint arXiv:2605.08761*, 2026.

Zhaoyang Yu, Aoyang Fang, Minghua Ma, Jaskaran Singh Walia, Chaoyun Zhang, Shu Chi, Ze Li, Murali Chintalapati, Xuchao Zhang, Rujia Wang, et al. Triangle: Empowering incident triage with multi-agent. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 674–686. IEEE, 2025.

Guibin Zhang, Muxin Fu, Kun Wang, Frank Wan, Miao Yu, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems. *Advances in Neural Information Processing Systems*, 38:12988–13018, 2026a.

Shiqi Zhang, Xinbei Ma, Yunqing Xu, Zouying Cao, Pengrui Lu, Haobo Yuan, Tiancheng Shen, Zhuosheng Zhang, Hai Zhao, and Ming-Hsuan Yang. Paracook: On time-efficient planning for multi-agent systems. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 27683–27701, 2026b.

Yujie Zhao, Boqin Yuan, Junbo Huang, Haocheng Yuan, Zhongming Yu, Haozhou Xu, Lanxiang Hu, Abhilash Shankarampeta, Zimeng Huang, Wentao Ni, et al. Ama-bench: Evaluating long-horizon memory foragentic applications.

Shanshan Zhong, Kate Shen, and Chenyan Xiong. Agentwebbench: Benchmarking multi-agent coordination in agentic web. *arXiv preprint arXiv:2604.10938*, 2026.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Daisy Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, et al. Multiagentbench: Evaluating the collaboration and competition of llm agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8580–8622, 2025.

## A Detailed Diagnostic Results

This section provides the complete diagnostic results summarized in Section 4. We separate sensitivity to delivery order from memory-level behavior to clarify where downstream failures arise.

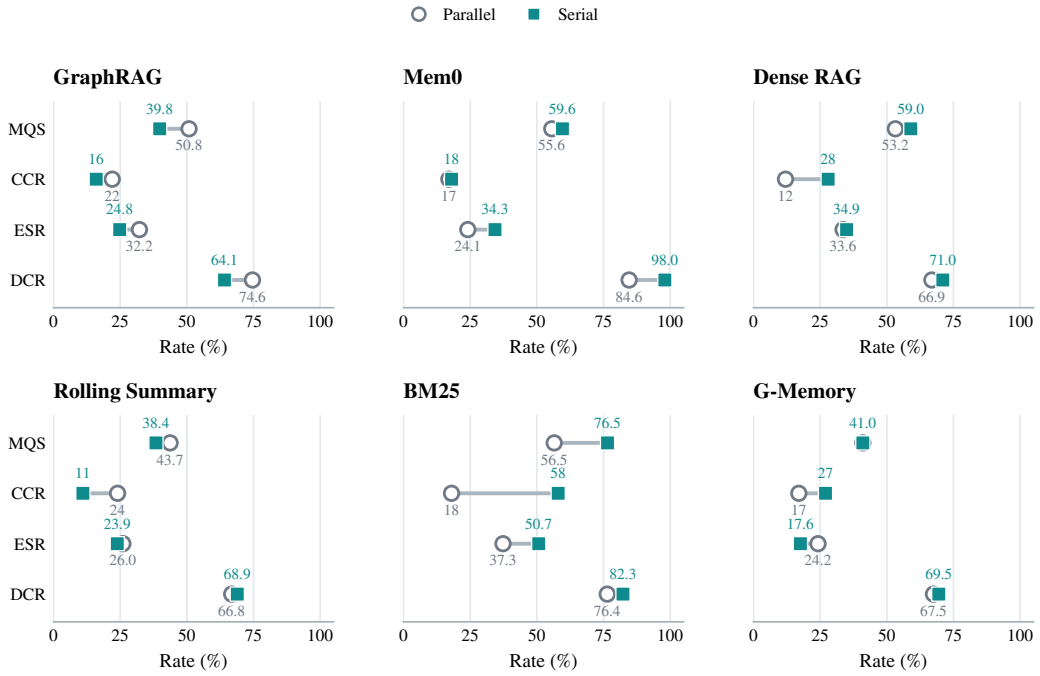


Figure 5: Effect of serializing Parallel delivery on MQS, CCR, ESR, and DCR across six memory systems.

Figure 5 shows method-dependent shifts in both answer quality and evidence recovery. BM25 improves broadly, gaining 20.0 MQS, 40 CCR, 13.4 ESR, and 5.9 DCR points, consistent with serial delivery reducing lexical competition among interleaved records. Dense RAG gains 5.8 MQS and 16 CCR points but only 1.3 ESR and 4.1 DCR points, indicating that nearly unchanged evidence is used more effectively downstream. GraphRAG moves in the opposite direction, losing 11.0 MQS, 6 CCR, 7.4 ESR, and 10.5 DCR points, so delivery order affects both its constructed graph state and the evidence exposed to the agent.

Rolling Summary loses 5.3 MQS and 13 CCR points even as DCR rises by 2.1 points, showing that retaining the dependency chain does not prevent an order-sensitive summary from changing the decision. Mem0 instead gains 10.2 ESR and 13.4 DCR points but only 4.0 MQS and 1 CCR point, so improved evidence recovery does not fully translate into complete-case success. G-Memory is similarly mixed: CCR rises by 10 points while MQS is unchanged and ESR falls by 6.6 points. Together, these results show that serialization can reduce interleaving but cannot replace order-robust branch and dependency tracking; the four five-case pilot comparisons remain descriptive.

Table 5 further decomposes evidence failures. Overall, Mem0 leads supersession accuracy, dependency-chain recall, and conflict-detection F1, while BM25 leads current-record precision,

Table 5: Memory-level diagnostics by workflow structure. All values are percentages and Overall is the arithmetic mean of the three settings. CRP: Current-Record Precision; SA: Supersession Accuracy; BCR: Branch Contamination Rate; DCR: Dependency-Chain Recall; CD-F1: Conflict-Detection F1; RA: Resolution Accuracy. Bold and underline mark the best and second-best results within each setting; BCR is lower-is-better, and dashes denote metrics that do not apply.

| Setting | Method | CRP | SA | BCR | DCR | CD-F1 | RA |
|---|---|---|---|---|---|---|---|
| Sequential | Long Context | – | – | – | – | – | – |
| | BM25 | 26.9 | 46.7 | <u>55.0</u> | <u>92.9</u> | <u>73.0</u> | 23.0 |
| | Dense RAG | <u>27.7</u> | 24.7 | 59.7 | 63.7 | 69.0 | 17.0 |
| | Rolling Summary | 21.1 | 45.0 | 75.7 | 62.6 | 70.4 | 5.0 |
| | Mem0 | 13.0 | **60.7** | 58.4 | **95.4** | **76.3** | <u>28.0</u> |
| | GraphRAG | 23.3 | <u>47.7</u> | **45.2** | 77.9 | <u>73.0</u> | **35.0** |
| | G-Memory | **31.1** | 24.3 | 58.3 | 63.1 | 65.2 | 18.0 |
| Parallel | Long Context | – | – | – | – | – | – |
| | BM25 | <u>74.6</u> | <u>73.7</u> | 47.9 | <u>76.4</u> | 61.1 | 31.0 |
| | Dense RAG | 62.7 | 53.0 | 49.7 | 66.9 | 64.7 | 30.0 |
| | Rolling Summary | **83.1** | 51.3 | 72.2 | 66.8 | 65.6 | 33.0 |
| | Mem0 | 74.0 | **74.3** | <u>45.0</u> | **84.6** | **72.0** | **40.0** |
| | GraphRAG | 66.5 | 52.0 | **40.3** | 74.6 | <u>68.1</u> | <u>37.0</u> |
| | G-Memory | 61.7 | 44.3 | 48.9 | 67.5 | 63.1 | 26.0 |
| Entangled | Long Context | – | – | – | – | – | – |
| | BM25 | 83.8 | <u>56.3</u> | **35.0** | **87.4** | **79.3** | **47.0** |
| | Dense RAG | 78.3 | 32.7 | 54.5 | 58.2 | 67.1 | 14.0 |
| | Rolling Summary | 75.8 | 35.3 | 62.1 | 75.8 | <u>71.4</u> | <u>35.0</u> |
| | Mem0 | **89.5** | **71.3** | <u>49.0</u> | <u>77.9</u> | 69.4 | 26.0 |
| | GraphRAG | <u>86.6</u> | 41.0 | 56.3 | 50.0 | 69.4 | 27.0 |
| | G-Memory | 81.3 | 28.3 | 52.6 | 58.9 | 66.7 | 13.0 |
| Overall | Long Context | – | – | – | – | – | – |
| | BM25 | **61.8** | <u>58.9</u> | **46.0** | <u>85.6</u> | <u>71.1</u> | **33.7** |
| | Dense RAG | 56.2 | 36.8 | 54.6 | 62.9 | 66.9 | 20.3 |
| | Rolling Summary | <u>60.0</u> | 43.9 | 70.0 | 68.4 | 69.1 | 24.3 |
| | Mem0 | 58.8 | **68.8** | 50.8 | **86.0** | **72.6** | 31.3 |
| | GraphRAG | 58.8 | 46.9 | <u>47.3</u> | 67.5 | 70.2 | <u>33.0</u> |
| | G-Memory | 58.0 | 32.3 | 53.3 | 63.2 | 65.0 | 19.0 |

branch isolation, and resolution accuracy; no method dominates every dimension. The workflow breakdown also separates record validity from relational completeness: on Entangled cases, several systems retrieve mostly current records but recover far fewer dependency chains, while Sequential BM25 and Mem0 recover over 92% of the chain despite low current-record precision. Thus, retrieving valid individual records does not ensure a complete or correctly reconciled memory view.

# B Detailed Case Studies

The following cases illustrate missing prerequisites and unresolved branch relations behind the aggregate errors; Oracle A serves as a positive control with complete evidence.

## B.1 Missing root evidence (numpy/stubtest).

Reuse of the owner decision `decision-59a00aee` is licensed only by the passing validation `validation-ac60fdb4` emitted just after it ($t$=5 vs. $t$=4). Amid concurrent validations, non-owner decisions and a name-only handoff index, the memory returns this residue plus an unbound later validation (`validation-b2181b`), but not `validation-ac60fdb4` nor the decision↔validation binding. The agent names the right decision yet, lacking the licensing report, follows the failed branch and answers `refresh_validation` instead of `reuse_validation`: recognizing an artifact by name does not recover the evidence that justifies its reuse. The bias is systematic—gold `reuse_validation` probes hit only 0–38% across seven systems (modal error always `refresh_validation`), while gold `refresh_validation` hits 84–100%—and costs a redundant re-validation of an already-validated revision.

**Case 1: Missing Root Evidence — `numpy/stubtest` (`entangled_13`)**

An owner decision may be **reused** only while the validation that justified it is still on record; otherwise it must be **refreshed**. The probe asks which holds for `decision-59a00aee`.

**1. Concurrent timeline**

- t=1 submission-475cedec (rev 15); t=2 review-c58e9b9f;
- t=3 validation-3704ac *passed* || validation-c61f67 *passed* || review-49487aa2;
- t=4 **decision-59a00aee** by *owner*-cf2754 (rev 15);
- t=5 **validation-ac60fdb4** *passed* — the validation that justifies the owner decision;
- t=13–30 a stream of *non-owner* decisions interleaved with other-ticket notes;
- t=30 handoff_scope-9e503ca0 (index of artifact names only).

**2. Correct evidence chain**

submission-475cedec → review-c58e9b9f → **validation-ac60fdb4 (passed)** → **decision-59a00aee (owner)** → handoff_scope-9e503ca0. The decisive link is the binding validation-ac60fdb4 ↔ decision-59a00aee.

**3. Records actually retrieved (long_context)**

- handoff_scope-9e503ca0 — the index, artifact *names* only;
- validation-b2181b — a passed validation from t=28, *not* bound to decision-59a00aee;
- **absent:** validation-ac60fdb4 and the decision↔validation binding; the owner decision surfaces only by name.

**4. Failure type**

**Failure type:** missing root evidence / broken decision–evidence provenance (cross-agent information continuity): the decision is recognized by name but the validation that justifies its reuse is not recovered. Scale: gold `reuse_validation` accuracy 0–38% across 7 systems, modal error `refresh_validation` (vs. 84–100% correct when gold is `refresh_validation`).

**5. Downstream consequence**

**Downstream:** the agent answers `refresh_validation` — it discards a reusable owner decision and re-validates an already-validated revision, spending an extra CI cycle and delaying the handoff.

**6. Memory mechanism that would have avoided it**

**Avoidable by:** provenance / dependency linking — store each decision together with (or edge-linked to) the validation report that supports it, or verbatim archival that never drops the passing report. When the binding is retained the same probe is answered `reuse_validation` with complete evidence (see Success Case, bm25).

Figure 6: Missing-root-evidence failure in the `numpy/stubtest` case. The memory view retains the owner decision and artifact names but omits the passing validation needed to justify reuse.

### B.2 Conflicting evidence across parallel branches (pydantic/urlschem).

On lane_A rev 2, `build-dddad1` fails at the same step `build-9cc0c2` passes ($t$=3), and the merge consumes only the passing pair—an open conflict calling for `resolve_reports`. The memory returns these records in isolation, with no link stating which supersedes which, so the agent picks the overturned failed build and answers `rerun_integration`/`rerun_lane`. Since lane/revision identifiers and the post-merge event are present, the loss is specifically the unmodeled supersession relation; accordingly gold `reuse_integration` collapses to 0–58%—records exist, but their relations do not reveal which evidence survives branch interaction.

**Case 2: Conflicting Evidence Across Parallel Branches — `pydantic/urlschem` (`parallel_4`)**

Parallel lanes report **opposing** results for one revision, then a merge consumes only **selected** builds. The agent must judge which evidence stays **valid** after the branches interact.

**1. Concurrent timeline**

- t=1 dispatch-d8d420 (lane_A←rev2, lane_B←rev2);
- t=1–2 lane_C/E/F rev-1 builds fail/pass (other branches, noise);
- t=3 lane_A rev2: **build-dddad1** *failed* (independent_ci) ‖ **build-9cc0c2** *passed* (workflow) — opposing current results, same lane+revision;
- t=4 lane_B rev2: build-bd0041 *passed*;
- t=5 integration-213e96 *passed* over {9cc0c2, bd0041} — merge consumed only the passing pair;
- t=19–27 further rev-1 integrations fail/pass; t=30 handoff_scope-41f683.

**2. Correct evidence chain**

dispatch-d8d420 + build-9cc0c2 + build-bd0041 + integration-213e96 + **build-dddad1** + handoff_scope-41f683. The decisive relation: dddad1 (failed) *opposes* 9cc0c2 (passed) on (lane_A, rev2), and merge 213e96 selected only the passing pair ⇒ the conflict is unresolved ⇒ `resolve_reports`.

**3. Records actually retrieved (dense_rag)**

- dispatch-d8d420, build-dddad1 (failed), integration-213e96 (passed merge);
- **no link** tying dddad1 to the (lane_A, rev2) result it opposes, nor that 213e96 consumed only a subset;
- (long_context retrieved only the handoff index — a pure retrieval loss.)

**4. Failure type**

**Failure type:** unmodeled supersession / opposition (concurrent consistency): records expose the entities but not which evidence remains valid after the branches interact. Scale: gold `reuse_integration` accuracy 0–58%; gold `resolve_reports` modal confusions `rerun_integration`/`rerun_lane`; parallel case accuracy 18–33%.

**5. Downstream consequence**

**Downstream:** the agent answers `rerun_integration` / `rerun_lane` — it re-runs builds or the integration instead of reconciling the lane_A rev2 conflict, so the overturned failed build keeps poisoning later merges and CI capacity is wasted.

**6. Memory mechanism that would have avoided it**

**Avoidable by:** revision-scoped current status with explicit supersession / opposition edges — per (lane, revision) keep the current result and record "conflicts-with / supersedes" links plus which builds each merge consumed; the lane_A rev2 opposition then becomes visible and `resolve_reports` follows.

**Mechanism disambiguation (which candidate is the root?)**

- *branch mixing*: present as rev-1 noise, but not the root — the decisive conflict lies within lane_A rev2;
- *missing revision identifiers*: **not** the cause — every build record carries lane+revision;
- *post-merge resolution event omitted*: **not** the cause — integration-213e96 is on the timeline and was retrieved, yet it only *selects* builds;
- ⇒ root = **unmodeled supersession / opposition relation**, not the other three.

Figure 7: Conflicting-evidence failure in the `pydantic/urlschem` case. Records expose both CI outcomes but lose the relations that identify the selected build after the branches interact.

### B.3 PAIRED COMPARISON WITH PERFECT RECALL (ORACLE A).

Re-running the identical probes under Oracle A, which injects the complete provenance-linked gold chain loop-free, the agent answers `reuse_validation` and `resolve_reports` correctly (score 2). The two errors therefore stem from incomplete or mis-scoped memory views—a dropped decision↔validation binding and a dropped supersession relation—not from unanswerable questions; a real memory retaining the binding (bm25, `entangled_9`) is likewise correct. Suite-level, Oracle A's 86.8% ceiling versus 49.0% for the best real memory (mem0) attributes the gap to what each memory view drops.

**Success Case: Oracle A (Perfect Recall) — `numpy/stubtest` (`entangled_13`)**

The *same* probe as Case 1, but memory is replaced by **perfect recall** (Oracle A): the complete gold evidence chain is injected loop-free. The agent then answers correctly — proving Case 1's error is a **memory / retrieval** failure, not a reasoning failure.

**1. Concurrent timeline**

- t=1 submission-475cedec (rev 15); t=2 review-c58e9b9f;
- t=4 **decision-59a00aee** by *owner*-cf2754 (rev 15);
- t=5 **validation-ac60fdb4** *passed* — the validation that justifies the owner decision;
- t=30 handoff_scope-9e503ca0.

**2. Correct evidence chain**

submission-475cedec → review-c58e9b9f → **validation-ac60fdb4 (passed)** → **decision-59a00aee (owner)** → handoff_scope-9e503ca0.

**3. Records actually retrieved (Oracle A, perfect recall)**

- the *exact* gold chain is injected, complete and noise-free: submission-475cedec, review-c58e9b9f, decision-59a00aee, **validation-ac60fdb4**, handoff_scope-9e503ca0;
- the decision↔validation binding is therefore present.

**4. Failure type**

**Failure type:** none — with complete provenance the binding that Case 1 lost is available, so no failure occurs.

**5. Downstream consequence**

**Downstream:** the agent answers `reuse_validation` (target decision-59a00aee), judge score 2 with complete evidence. The same holds for Case 2's probe: `parallel_4` `resolve_reports` also scores 2 under Oracle A.

**6. Memory mechanism responsible**

**Why it worked:** perfect recall / complete provenance co-storage. Because Oracle A is loop-free and reasoning-only, its success on the *identical* probe isolates Case 1's `refresh_validation` error to the memory layer (the lost binding), not to reasoning. A real memory that retains the binding reproduces this success (bm25 on sibling case `entangled_9`, joint-correct), so the effect is not oracle-specific.

Figure 8: Oracle-evidence success on the same probes. Complete provenance-linked evidence enables the agent to recover the valid decision.

## C Detailed Cross-Model Results

Table 6: Workflow-level comparison of memory systems under `qwen3.8-max` and `qwen3.8-flash`. MQS, CCR, and ESR are percentages; Tok. and Time denote tokens and seconds per query. Max values are from the main experiment. Bold and underline mark the best and second-best results within each workflow and model; lower costs are better. – denotes not reported.

| Setting | Method | Max | | | | | Flash | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MQS | CCR | ESR | Tok. | Time | MQS | CCR | ESR | Tok. | Time |
| Sequential | Long Context | 21.6 | 0 | – | 7,353 | **45.4** | 1.7 | 0 | – | 4,471 | **112.6** |
| | BM25 | 35.8 | 3 | 45.9 | 6,930 | 52.6 | 30.1 | 2 | **36.1** | 4,468 | 128.3 |
| | Dense RAG | 38.6 | 3 | 31.5 | 7,078 | 57.4 | 21.1 | 2 | 8.3 | **4,365** | 145.2 |
| | Rolling Summary | 34.0 | 1 | 27.6 | 10,285 | 132.6 | 25.0 | 1 | 12.5 | 7,208 | 218.4 |
| | Mem0 | 35.9 | 5 | 37.6 | **6,782** | 411.4 | 22.5 | 5 | 18.9 | 4,915 | 799.1 |
| | GraphRAG | **56.5** | **29** | **60.9** | 10,776 | 190.3 | **30.7** | **7** | 10.3 | 9,518 | 296.3 |
| | G-Memory | 32.2 | 6 | 21.9 | 7,864 | 113.2 | 11.5 | 3 | 5.7 | 4,624 | 190.9 |
| Parallel | Long Context | 2.7 | 0 | – | 7,754 | **48.4** | 1.3 | 0 | – | 4,815 | **120.4** |
| | BM25 | **56.5** | 18 | **37.3** | 7,865 | 58.7 | **40.9** | 4 | **31.3** | 4,798 | 151.9 |
| | Dense RAG | 53.2 | 12 | 33.6 | 7,545 | 61.2 | 31.1 | 2 | 12.9 | **4,557** | 148.4 |
| | Rolling Summary | 43.7 | **24** | 26.0 | 11,138 | 168.2 | 30.0 | **20** | 13.3 | 7,381 | 247.9 |
| | Mem0 | 55.6 | 17 | 24.1 | **7,069** | 427.0 | 33.9 | 7 | 14.2 | 4,798 | 936.5 |
| | GraphRAG | 50.8 | 22 | 32.2 | 16,740 | 243.2 | 32.2 | 0 | 21.1 | 10,829 | 349.0 |
| | G-Memory | 41.0 | 17 | 24.2 | 8,296 | 147.3 | 21.8 | 3 | 11.5 | 5,210 | 232.6 |
| Entangled | Long Context | 16.3 | 0 | – | 7,682 | **49.5** | 9.7 | 0 | – | 4,541 | 143.2 |
| | BM25 | 33.2 | 10 | **22.6** | 7,666 | 55.7 | 30.7 | 8 | 21.3 | 4,711 | 130.5 |
| | Dense RAG | 30.7 | 6 | 20.2 | 7,354 | 59.0 | 26.5 | 4 | 5.2 | **4,370** | **119.0** |
| | Rolling Summary | 28.3 | 12 | 11.7 | 11,339 | 185.6 | 24.3 | 4 | **33.3** | 7,894 | 174.0 |
| | Mem0 | **47.9** | **19** | 14.6 | **7,015** | 395.2 | **45.2** | **12** | 21.2 | 4,570 | 643.4 |
| | GraphRAG | 35.4 | 10 | 19.2 | 17,062 | 248.7 | 22.6 | 4 | 11.9 | 10,816 | 323.4 |
| | G-Memory | 26.2 | 6 | 13.9 | 8,174 | 141.3 | 20.7 | 0 | 6.9 | 4,931 | 190.5 |
| Overall | Long Context | 13.5 | 0.0 | – | 7,596 | **47.8** | 4.2 | 0.0 | – | 4,609 | **125.4** |
| | BM25 | 41.8 | 10.3 | 35.3 | 7,487 | 55.7 | **33.9** | 4.7 | **29.6** | 4,659 | 136.9 |
| | Dense RAG | 40.8 | 7.0 | 28.4 | 7,326 | 59.2 | 26.2 | 2.7 | 8.8 | **4,431** | 137.5 |
| | Rolling Summary | 35.3 | 12.3 | 21.8 | 10,921 | 162.1 | 26.4 | **8.3** | 19.7 | 7,494 | 213.4 |
| | Mem0 | 46.5 | 13.7 | 25.4 | **6,955** | 411.2 | **33.9** | 8.0 | 18.1 | 4,761 | 793.0 |
| | GraphRAG | **47.6** | **20.3** | **37.4** | 14,859 | 227.4 | 28.5 | 3.7 | 14.4 | 10,388 | 322.9 |
| | G-Memory | 33.1 | 9.7 | 20.0 | 8,111 | 133.9 | 18.0 | 2.0 | 8.0 | 4,922 | 204.7 |

**Analysis.** Flash generally lowers MQS and CCR across all three workflows while preserving most method-specific strengths: with both models, GraphRAG leads Sequential MQS and CCR, BM25 leads Parallel MQS and ESR, and Mem0 leads Entangled MQS and CCR. The largest shift occurs for GraphRAG on Sequential cases, where MQS and ESR fall from 56.5% and 60.9% with Max to 30.7% and 10.3% with Flash. On Entangled cases, Mem0 leads answer quality under both models without leading ESR; Flash also uses fewer tokens for every method but generally takes longer.

These differences reflect how each memory system supplies evidence. GraphRAG traces and composes relations along causal chains, so errors accumulate across multiple model-dependent steps under Flash; BM25 directly matches repeated identifiers and remains more stable when Parallel branches are lexically distinct. Mem0 can recover an actionable decision without retaining its complete prerequisite chain, explaining the gap between its MQS/CCR and ESR on Entangled cases. Because elapsed time includes the full memory pipeline rather than token generation alone, lower token use does not guarantee lower latency.