Diagnostics and Infrastructure for Foundation Model-Based Multi-Agent Systems: A Review of the Runtime Stack

Yu Sun*[1], Zuodong Xiang[2], Yike Zhang[3], Xinyu Guan[4], Pengcheng Xu[5], Yizirui Fang[3], Zhenhua Huang[6], and Hailu Xu[7]

[1]School of Engineering and Applied Science, University of Pennsylvania, Philadelphia, PA, United States
[2]Department of Electrical and Computer Engineering, College of Engineering, University of California, Davis, Davis, CA, United States
[3]Department of Computer Science, Whiting School of Engineering, Johns Hopkins University, Baltimore, MD, United States
[4]University of Glasgow, Glasgow, Scotland, United Kingdom
[5]Department of Computer Science, University of California, Irvine, Irvine, CA, United States
[6]School of Computer Science, South China Normal University, Guangzhou, Guangdong, China
[7]Department of Computer Engineering and Computer Science, College of Engineering, California State University, Long Beach, Long Beach, CA, United States

Abstract

Foundation model-based multi-agent systems are moving into production, yet work on runtime reliability and serving efficiency is still fragmented across diagnostics and infrastructure. This review synthesizes 137 references, 45 systems, and 8 benchmarks through a unified runtime-stack perspective. We organize diagnostics into failure localization, observability, safety, and benchmarks, and infrastructure into memory management, prefix-aware scheduling, cache lifecycle, and power management. The central synthesis is that both layers act on overlapping runtime state but optimize different objectives. Diagnostics benefits from retaining traces, context, and replayable state, whereas serving infrastructure reclaims or compresses state to satisfy memory, latency, and throughput budgets. We term this interaction the *diagnostics–infrastructure retention tension*. We further identify three workload characteristics—sparse activation, invocation distance, and inter-agent memory sharing—that shape agent-aware serving, and use explicit evidence tiers to separate full-text-verified results from abstract-level or contextual evidence. Rather than ranking heterogeneous systems, we derive testable co-design directions: retention-aware eviction, failure-aware scheduling, budgeted observability, and infrastructure-triggered diagnostics. This framing positions runtime state retention as a shared systems resource and clarifies where current multi-agent diagnostics and serving research remain disconnected.

Keywords: multi-agent systems; foundation models; diagnostics; observability; serving infrastructure; memory management; scheduling; runtime reliability

1 Introduction

Foundation model-based multi-agent systems (MAS), in which multiple Large Language Model (LLM) agents collaborate to solve complex tasks [27, 39, 76], are increasingly moving from demonstrations toward deployed workflows. A vendor survey reports that 72% of organizations are actively using or testing multi-agent AI systems [114] (Tier-3 vendor data, reported as stated). As systems such as MetaGPT [27], Internet of Agents [14], and Graph-of-Agents [113]

*Corresponding author: yu.sun@alumni.upenn.edu

Table 1: Comparison with existing surveys. • = covered, ○ = not covered. [a]A 2026 survey of MAS collaboration and failure; it organises failure conceptually and does not catalogue system-level localization methods or cover infrastructure. [b]A 2024 survey covering workflow and architecture; its "infrastructure" is at that level, not multi-agent serving (memory, scheduling, cache lifecycle, power). "Quant." denotes that the survey reports quantitative results for the systems it reviews, at the verification depth described in Section 2; it does not imply that those numbers are compared across systems. **Pub.**: ✓ = peer-reviewed venue, ○ = preprint, — = not a publication. Pub. records the venue only and is not a quality judgement.

Survey	Year	Venue	Orch.	Diag.	Infra	Quant.	Pub.
Zhu et al.	2026	Future Internet	•	○	○	○	✓
AgentOps	2025	preprint	○	•	○	○	○
Mohammadi et al.	2025	ACM KDD	○	○	○	○	✓
Latency survey	2026	IEEE Access	○	○	•	○	✓
Qi et al.[a]	2026	preprint	○	•	○	○	○
Li et al.[b]	2024	Vicinagearth	○	○	○	○	✓
Ours	2026	—	○	•	•	•	—

become more architecturally sophisticated, reliability and efficiency depend on operational layers that are less mature than orchestration itself. We focus on two of them: the *diagnostics layer*, which supports failure localization, observability, and safety, and the *infrastructure layer*, which manages memory, scheduling, and resources for multi-agent workloads.

Existing surveys largely treat these concerns separately. Orchestration frameworks (LangGraph, CrewAI, AutoGen) have received substantial survey attention [137], as have offline agent evaluation [63, 67, 110], safety [19, 111], AgentOps [11, 21, 99], and single-model serving and latency optimization [74]. In our curated corpus, we count 45 systems across our own tables[1]. The resulting literature exposes complementary pieces of the runtime stack, but the relationship between the state required for diagnosis and the state managed by serving policies remains under-analyzed.

This review therefore brings the two operational layers into one frame. Table 1 contrasts our scope against six adjacent surveys. General agent surveys [3, 42, 91, 101] cover architectural patterns and collaboration mechanisms; [75] surveys collaboration and failure but not serving infrastructure; and [42] uses "infrastructure" for the workflow layer rather than runtime memory, scheduling, cache lifecycle, or power. In the surveyed literature, we did not find a review that jointly organizes runtime diagnostics and multi-agent serving infrastructure while also making the verification depth of quantitative claims explicit.

This is a curated critical review rather than a systematic review. The inclusion boundary, evidence-selection rules, and verification tiers are described in Section 2. Quantitative claims used for comparison come from full-text sources; abstract-level results are marked *[a]* and are not used to compute cross-system deltas or rankings. Figure 1 illustrates the scope.

Our contributions are:

1. **Diagnostics taxonomy.** We organize multi-agent diagnostics into four subcategories spanning failure localization (Section 3.1), observability (Section 3.2), safety (Section 3.3), and benchmarks (Section 3.4), covering three failure-localization paradigms (statistical, LLM-guided, RL-trained) and three safety paradigms (topology-based, policy verification, ML-layered).

2. **Infrastructure challenge analysis.** We identify three multi-agent workload characteristics—sparse activation, invocation distance, and inter-agent memory sharing—that are only partially exposed to general-purpose serving policies, and synthesize application-aware

[1]Counted across our own system tables: 12 failure-localization methods, 12 observability and safety systems, and 21 infrastructure systems. Benchmarks (8) and comparator surveys (6) are excluded.

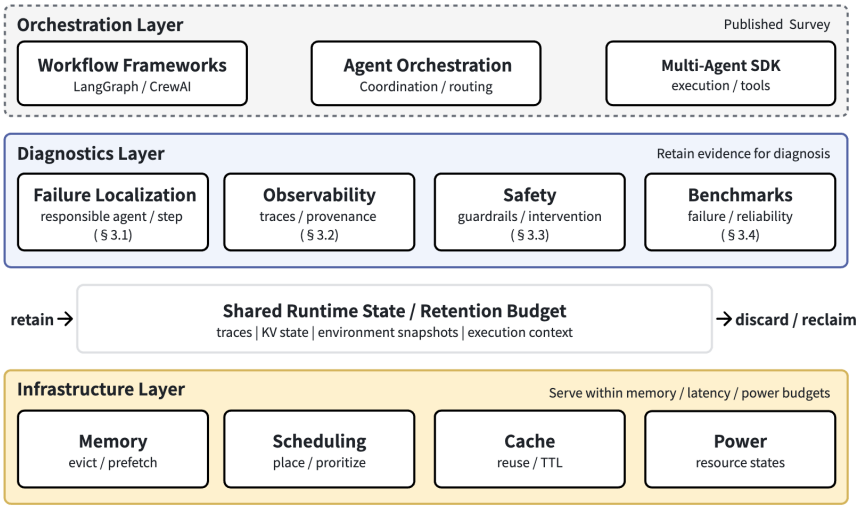


Figure 1: Survey scope: the multi-agent runtime stack. We review the diagnostics layer (Section 3) and the infrastructure layer (Section 4); the orchestration layer is shown for context and is well-covered by existing surveys. The shared runtime-state band marks the retention tension analyzed in Section 5.

systems spanning memory optimization (ScaleSim, up to 1.74× speedup [72]), prefix-aware scheduling (TokenCake [9]), cache lifecycle (Continuum), and power management (KAIROS).

3. **Cross-layer tension analysis.** We synthesize a cross-layer retention tension: diagnostics benefits from preserving runtime evidence, while serving infrastructure benefits from reclaiming state under memory and latency budgets. We connect this tension to observed mechanisms in the surveyed systems and derive four co-design opportunities that are made testable in Section 6.

2 Background and Review Methodology

A foundation model-based agent operates in a perception–reasoning–action loop, using an LLM to process observations and execute actions such as tool invocations, code execution, and inter-agent communication. The underlying model workloads can also be multimodal: recent systems combine visual, audio, and clinical or platform metadata for domain-specific prediction and governance [90, 102, 107]. A multi-agent system (MAS) comprises multiple such agents, as exemplified by MetaGPT [27], CAMEL [39], ChatDev [76], and IoA [14], operating within a shared environment or communicating to accomplish collective objectives [42, 101]. Recent multi-agent applications also span reinforcement-learning-enhanced collaborative reasoning and multimodal misinformation detection [43, 44]. MAST [12] identifies 14 recurring failure patterns in such systems, motivating runtime diagnostics rather than evaluation only at the final task output.

Representation-level adaptation is another adjacent concern: recent speech work studies adversarial attribute obfuscation, accent-controlled editing, and pronunciation representations [79–81]. These works are not multi-agent infrastructure methods, but they illustrate the heterogeneity of model state and representations that agentic applications may invoke.

Three broad *system properties* recur across the literature reviewed here [70]: (1) distributed execution across multiple agents; (2) inter-agent communication, which creates traces spanning agent boundaries and allows failures to propagate through communication channels [13, 24]; and (3) inter-agent memory sharing, where prompts, tools, and cached state may be reused across the

system. Section 4.1 derives a serving-side view from these properties and highlights three workload characteristics—sparse activation, invocation distance, and inter-agent memory sharing—that directly affect memory placement, scheduling, and cache policy. This distinction separates general MAS structure from the narrower workload characteristics used in our infrastructure analysis.

Review scope, selection, and evidence coding. This article is a curated critical review, not a systematic review. We focus on runtime operational mechanisms for foundation-model agents and MAS, distinct from three well-surveyed areas: orchestration [137], single-model inference [36], and offline evaluation [63]. For inclusion in the system tables, a work must directly address at least one runtime function in our scope: failure localization, observability/tracing, safety intervention, diagnostic benchmarking, or serving-time memory, scheduling, cache, or power management for agentic workloads. General LLM-serving systems are included when they provide a baseline or mechanism directly reused by agent workloads; orchestration-only frameworks, general agent architectures, and long-term memory representation without serving-time resource management are used only as context. To make the review more reproducible without overstating exhaustiveness, candidate papers were collected iteratively from primary systems papers, adjacent surveys, major peer-reviewed venues, and forward/backward citation tracing. We prioritize work from 2024–2026 because foundation-model multi-agent runtime research is unusually recent, while retaining older serving papers when they define mechanisms directly reused by agentic systems. A paper enters the core system tables only when its runtime role can be identified from the source and mapped to one of the functions above. Because the corpus was not generated by a preregistered database protocol with fixed search strings and PRISMA-style screening, the reported counts should be interpreted as a curated landscape rather than an exhaustive census.

To make the evidential depth of individual claims explicit, we assign cited systems to the following tiers:

1. **Tier 1a (full-text verified).** The source was read in full and the specific quantitative claim was located in the body text or a table. These numbers may be compared and quoted when the underlying benchmark and protocol support the comparison.
2. **Tier 1b (read in full, not page-verified).** The source was read in full and has a confirmed peer-reviewed venue, but the quantitative claim was taken from the published text without being cross-checked against a specific page, table, or figure. These numbers may be quoted with the † marker but are not used to compute cross-system deltas.
3. **Tier 2 (abstract-only).** Only the abstract was read; the body was not examined. This applies equally to preprints and venue-published papers whose body we did not read. The claim is reported as stated and marked *[a]*. The distinction from Tier 1b is verification depth, not venue status.
4. **Tier 3.** Documentation, platform pages, and vendor reports, labelled by source type and used for context only.

Applying this procedure yields **8 Tier-1 papers across 9 table entries: 5 in Tier 1a** (FAMAS, TrajAudit, Agent Traces, TraceElephant, ScaleSim) and **3 in Tier 1b** (G-Safeguard at ACL 2025, ShieldAgent at ICML 2025, and AgenTracer at ICLR 2026). RootSE is the benchmark introduced by TrajAudit, so the two rows share one paper. Systems such as RAFFLES, AgentLocate, MultiAgentBench, and Who&When have confirmed venues but were read at abstract level and therefore remain Tier 2; LlamaFirewall is likewise Tier 2. Throughout the review, publication venue and verification tier are kept separate, and quantitative values

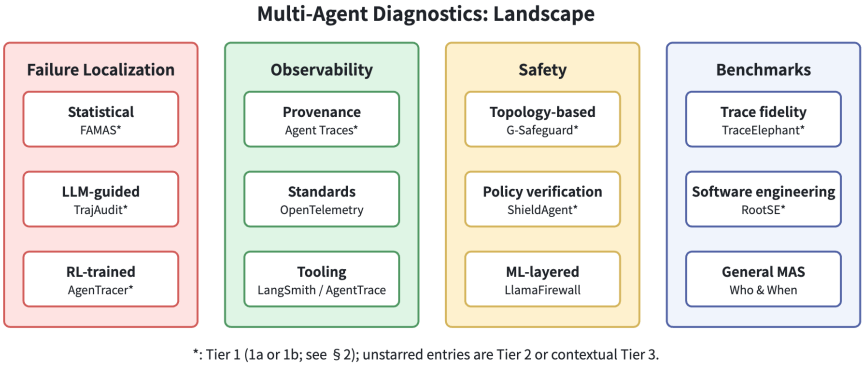


Figure 2: Landscape of multi-agent diagnostics across four subcategories. Stars ($\star$) mark Tier 1 systems (Tier 1a or 1b); unmarked leaves are Tier 2 (abstract-verified) or, for standards and tooling, Tier 3. The tier rule and the 1a/1b split are given in Section 2.

from heterogeneous benchmarks are presented as landscape evidence rather than as a global ranking.

3 Diagnostics

This section presents our taxonomy of multi-agent diagnostics, organized into four subcategories: failure localization (Section 3.1), observability (Section 3.2), safety (Section 3.3), and benchmarks (Section 3.4). Figure 2 maps the landscape of diagnostic functions and representative systems, whereas Figure 3 provides a complementary process view showing where tracing, localization, root-cause output, and safety act along execution.

3.1 Failure Localization

When a multi-agent system fails, identifying the responsible agent and faulty action remains a central challenge. Our surveyed set contains twelve failure-localization methods from 2024–2026. We group them into three recurring paradigms—statistical, LLM-guided, and RL-trained—that use distinct methodological assumptions and evaluation conventions. These paradigms differ along two axes: *input requirements* (replay-based vs. single-execution) and *attribution granularity* (agent-level, step-level, or action-level). Table 2 summarizes the landscape.

Statistical approaches These methods apply spectrum-based fault localization (SBFL) to multi-agent traces. FAMAS [24] replays trajectories $k = 20$ times and applies four suspiciousness metrics (action coverage, failure entropy, temporal decay, Kulczynski2), achieving 66% top-1 action-level accuracy on the 82-failure replayable subset of Who&When [24]. Complementary statistical methods extend SBFL without replays: CausalFlow [97] constructs hierarchical causal graphs (36.2% step accuracy, abstract-verified) and StepFinder [136] applies temporal semantic analysis (+10.35pp, abstract-verified). Zero-replay debugging [34] eliminates replay overhead entirely through knowledge-based reasoning (abstract-verified).

LLM-guided approaches These treat diagnosis as a reasoning task. Adjacent work on process supervision, structured reasoning supervision, and self-generated rewards similarly studies intermediate or internally generated supervision rather than only final outputs [30, 31, 48, 51]. TrajAudit [94] deploys an LLM investigator with Semantic Saliency Folding (SSF, 86–92% token reduction) and Prior Failure Reasoning (PFR), achieving +24.7pp step-level accuracy on RootSE

Table 2: Failure localization methods for multi-agent systems. Tier 1 results are from full-text verification, except those marked †, which are venue-published but reported from the published text without page-level cross-verification (Tier 1b); Tier 2 results marked [a] are as reported in source abstracts and are not independently verified. The tier rule and the 1a/1b split are given in Section 2. **Pub.**: ✓ = peer-reviewed main-track venue, ∘ = preprint or workshop paper. — = not stated in the source. *These numbers come from different benchmarks, protocols, and observability settings; this table is a landscape summary, not a head-to-head comparison (see Section 5).*

System	Approach	Gran.	Bench.	Overhead	Key Result	Tier	Pub.
FAMAS	Stat. (SBFL)	Action	Who&When	20 replays	66% top-1	1	∘
TrajAudit	LLM-guided	Step	RootSE	86–92% tok.	+24.7pp	1	∘
AgenTracer	RL (8B)	Agent	Who&When	8B train	69.10%	1†	✓
AgentRx	LLM-guided	Step	—	—	+23.6pp step [a]	2	∘
FALAT	Dependency	Step	Who&When	—	46.0% step [a]	2	∘
DoVer	Interv.+Replay	Step	—	multiple re-plays	18–28% [a]	2	✓
MASPrism	Prefill SLM	Step	—	—	— [a]	2	∘
RAFFLES	Reasoning	Step	Who&When	—	— [a]	2	✓
AgentLocate	LLM judge	Step	—	—	— [a]	2	✓
CausalFlow	Stat. (Causal)	Step	—	—	36.2% [a]	2	∘
StepFinder	Temporal sem.	Step	Who&When	—	+10.35pp [a]	2	∘
EAGER	Contrastive	System	—	—	prelim. [a]	2	∘

(93 failures); ablation confirms SSF as critical (−11.4% vs. −3.5% for PFR) [94]. The LLM-guided family has rapidly diversified: AgentRx [7] provides domain-agnostic diagnosis (+23.6pp step, +22.9pp category accuracy), TRAIL [20] combines trace reasoning with issue localization, RAFFLES [128] applies reasoning-based attribution (EACL 2026), and AgentLocate [104] adds multi-perspective verification. Hierarchical approaches [6] decompose errors across agent boundaries, while VerifyMAS [77] introduces hypothesis verification for attribution (all abstract-verified).

RL-trained approaches These replace hand-designed metrics with learned attribution. Related data-valuation work uses hierarchical contrastive Shapley values to prioritize samples, illustrating another structured attribution mechanism at the data level rather than the agent-execution level [106]. AgenTracer [116] trains an 8B-parameter failure tracer via reinforcement learning, achieving 69.10% agent-level accuracy on Who&When (184 failures), outperforming DeepSeek-R1 by 12.21pp (ICLR 2026) [116]. MASPrism [52] takes a lightweight alternative, using a prefill-stage SLM for attribution (abstract-verified).

Beyond attribution. Several methods extend failure localization toward repair and intervention. Related work on counterfactual rollout replay and trajectory-level guidance further illustrates how alternative execution paths or intermediate trajectory structure can provide supervision signals [29, 45]. diagnosis-driven automatic repair [58] closes the loop from attribution to fix, AgentTether [123] combines graph-guided diagnosis with runtime intervention, and interactive debugging [22] enables human-in-the-loop steering (CHI 2025). Additional methods include FALAT [82] (dependency-guided search), DoVer [57] (intervention + replay, recovering 18–28% of failed trials), EAGER [118] (unsupervised contrastive trace representation), and holistic evaluation frameworks [59] (all abstract-verified). Meta-evaluation work [50] has begun assessing the attribution methods themselves, while post-failure reporting benchmarks examine whether tool-using agents can accurately communicate what went wrong after execution [129], and the same shift toward online auditing appears in AgentForesight [115], which monitors a trajectory while it runs and emits an alarm before the failure surfaces rather than reconstructing it afterwards (abstract-verified).

Non-comparability. The "Key Result" column aggregates measurements taken on different benchmarks (Who&When, RootSE, or task-specific sets), under different observability settings (output-only vs. full traces),

Table 3: Observability, tracing, and safety systems. **Pub.**: ✓ = peer-reviewed main-track venue, ○ = preprint, workshop paper, or no confirmed venue, — = documentation or platform (not a publication). Pub. records the venue only and is not a quality judgement.

System	Category	Open	Tier	Pub.
Observability & Tracing				
Agent Traces	Taxonomy	—	1	○
OTel GenAI	Standard	✓	3	—
AgentTrace	Structured log	—	2	○
LangSmith	Platform	—	3	—
Insights Gen.	Diagnostic	—	2	○
Safety & Reliability				
G-Safeguard	Topology GNN	—	1†	✓
ShieldAgent	Policy verif.	—	1†	✓
LlamaFirewall	ML-layered	✓	2	○
SAIGuard	Proactive MAS	—	2	○
AGrail	Lifelong guard	—	2	○
AgentDoG	Alignment	—	2	○
ReGuard	Reactive	—	2	○

†Venue-published; not page-level verified.

and against different baselines. Methods also differ in what they require at inference time (replay budget, training, token cost). The rows therefore describe a landscape of approaches, not a ranked comparison; no cross-row ranking should be inferred. Pub. records the publication venue only and is not a quality judgement.

3.2 Observability and Tracing

Failure localization depends on rich execution traces, yet the multi-agent observability stack remains fragmented across three levels: conceptual models, protocols, and tooling [88]. More broadly, recent analyses of reasoning trajectories examine which tokens or intermediate states carry disproportionate learning signal and how reasoning can be evaluated under memorization constraints [28, 32].

At the *conceptual level*, [98] present a seven-dimension provenance taxonomy (trace sources, evidence units, execution units, nine relation types, run-to-token granularity, online/offline timing, interaction context). This taxonomy exposes a key architectural split between post-hoc debugging and runtime enforcement. Complementary taxonomies address observability from an operations perspective [11, 21, 100], and a multi-layer survey [88] proposes a five-layer framework spanning data, model, agent, system, and application observability (all abstract-verified).

At the *protocol level*, the OpenTelemetry GenAI conventions [69] provide vendor-neutral attributes (`gen_ai.*`) for LLM spans, but semantic interoperability remains limited. AgentTrace [4] proposes structured logging specifically for agent systems (AAAI 2026 Workshop, abstract-verified), and process mining approaches [23] apply workflow analysis to agent traces (abstract-verified).

At the *tooling level*, a vendor survey reports 89% observability adoption among enterprises using AI agents [114] (Tier-3 vendor data), with LangSmith [37], Langfuse, and Arize Phoenix as leading platforms. The Insights Generator [60] demonstrates that corpus-level trace mining improves downstream performance by 30.4pp (abstract-verified). Security-focused tooling has also emerged: auditable agent frameworks [38, 68] build unified graphs for security auditing, and view-oriented compilers [119] enable multi-perspective trace analysis (all abstract-verified).

Table 3 summarizes the observability and safety landscape.

3.3 Safety and Reliability

Multi-agent safety poses a distinct challenge: attacks propagate through inter-agent communication, so a single compromised agent can corrupt downstream outputs [19, 111]. Recent surveys identify this cross-agent propagation as the defining difference from single-agent safety [93]. Three paradigms have emerged, each at a different abstraction level. We follow the literature's convention of treating safety as a diagnostic subcategory; as Section 6 argues, this placement is contestable, because a guardrail must both inspect state and act on the serving path.

Topology-based approaches exploit the communication graph. G-Safeguard [96] uses a GNN to detect anomalies on the multi-agent utterance graph and applies topological intervention, recovering over 40% of the performance lost to prompt injection (ACL 2025; Tier 1b). SAIGuard [86] extends this to communication-state simulation for proactive defense, SentinelAgent [26] detects node-, edge-, and path-level anomalies on a typed execution graph, and BlindGuard [61] performs the same detection without attack labels by training on normal-only interaction data (all abstract-verified). This graph-level reasoning is particularly salient in multi-agent settings because failures can propagate across inter-agent communication edges.

Policy-reasoning guardrails verify another agent's action trajectory against explicit safety policies. ShieldAgent [16] is a single guardrail agent that extracts verifiable rules from policy documents and checks each proposed action against them, reporting 90.1% recall of violated rules while using 64.7% fewer API queries and 58.2% less inference time than prior approaches (ICML 2025; Tier 1b). [8] provide a theoretical foundation via a three-layer probabilistic architecture, and VeriGuard [62] applies verified code generation for safety-critical agents. The Swiss Cheese model [83] offers a complementary defense-in-depth framework (all abstract-verified).

ML-layered defenses stack classifiers. LlamaFirewall [18] layers PromptGuard2 (AUC 0.98) with an alignment verifier, reducing attack success from 17.6% to 1.75%. AGrail [56] introduces lifelong adaptive guardrails that evolve with agent deployment (abstract-verified), and GuardAgent [105] enables knowledge-based guardrail reasoning (abstract-verified).

Complementary systems span pre-execution alignment (AgentDoG [49]), control-theoretic recovery (ReGuard [73]), automated auditing (AgentAuditor [54], NeurIPS 2025), and admission control for privileged LLM agents [130]. Safety benchmarks include GuardianAgentBench [65] (580 scenarios), OSGuard [64] (2.7% FPR), Agent-SafetyBench [122], and AgentHarm [5] (all abstract-verified).

3.4 Benchmarks

Progress in diagnostics depends on benchmarks varying in failure coverage, trace fidelity, and evaluation scope. Table 4 summarizes the landscape.

The highest-fidelity benchmark is TraceElephant [13]: 220 failures from three production systems with full observability. At the *agent* level, complete traces raise attribution accuracy from 51% for output-only logs to 62.2% (+11.2pp, a 22% relative gain), both under the same All-at-Once protocol. At the *step* level, the same comparison raises accuracy from 16% to 28.1% (+12.1pp, a 76% relative gain), i.e. step-level attribution is substantially more sensitive to missing information than agent-level attribution [13]. Who&When [121] provides 184 failures from 127 multi-agent systems as the first dedicated failure attribution benchmark, now with a meta-evaluation extension [50] (abstract-verified). Domain-specific, RootSE [94] offers 93 SWE-bench failures with step-level annotations.

Beyond failure attribution, ReliabilityBench [25] reveals that pass@1 overestimates reliability by 20–40% (abstract-verified), BenchGuard [92] audits benchmark quality (abstract-verified), and MultiAgentBench [134] evaluates multi-agent collaboration (ACL 2025, abstract-verified). Safety-specific benchmarks include Agent-SafetyBench [122], AgentHarm [5], Agent Security

Table 4: Diagnostics and safety benchmarks. **Pub.**: ✓ = peer-reviewed venue, ○ = preprint / no confirmed venue. Pub. records the venue only and is not a quality judgement.

Benchmark	Scale	Traces	Domain	Tier	Pub.
TraceElephant	220	Full	Multi	1	○
Who&When	184	Output	Multi	2	✓
RootSE	93	Full	SWE	1	○
MultiAgentBench	—	—	Collab	2	✓
Agent-SafetyBench	—	—	Safety	2	○
AgentHarm	—	—	Safety	2	○
ReliabilityBench	—	—	Reliab.	2	○
BenchGuard	—	—	Meta	2	○

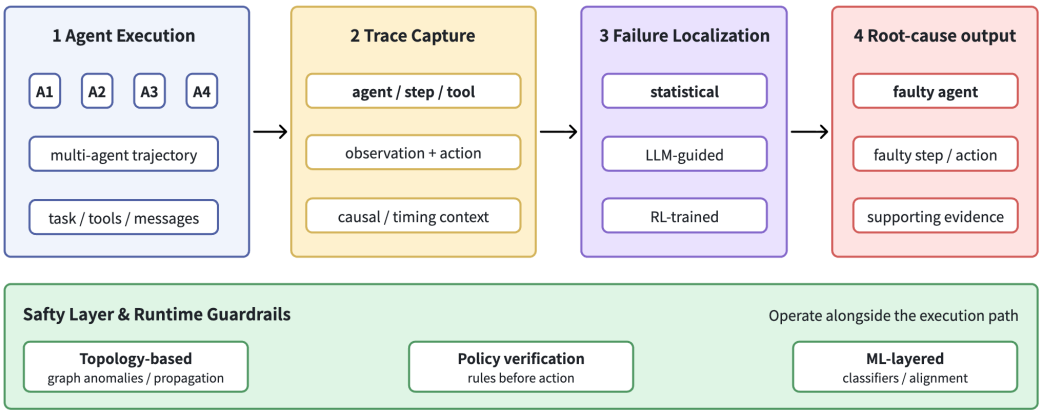


Figure 3: Process view of multi-agent diagnostics. Execution traces are collected through observability infrastructure, then processed through failure localization to produce root-cause attribution, while safety mechanisms may operate continuously alongside execution. This figure complements the landscape in Figure 2: it shows when diagnostic functions act rather than reclassifying the systems.

Bench [117] (ICLR 2025, which reports attack success above 84% with existing defenses barely effective), and TAMAS [35] (ACL 2026, covering Byzantine, colluding, and contradicting agents) (all abstract-verified). A unified capabilities framework [135] and a benchmarking survey [63] address the meta-question of what benchmarks should measure.

Among the benchmarks surveyed here, none evaluates diagnostics–infrastructure interaction: how diagnostic overhead affects serving performance, or how retention and serving policies affect diagnostic fidelity.

Diagnostics synthesis. Across the four diagnostic subcategories, the common dependency is retained execution evidence: localization methods require traces or replays, observability determines what evidence exists, and safety mechanisms must inspect enough context to intervene. The literature is broad but methodologically heterogeneous, so the taxonomy is most useful for comparing assumptions—trace fidelity, replay budget, granularity, and intervention timing—rather than headline accuracy across rows. This shared dependence on runtime evidence motivates the retention analysis in Section 5.

4 Infrastructure

Most LLM serving systems were designed around request-level inference and expose only part of the structure available in multi-agent workloads. We identify three multi-agent serving challenges

(Section 4.1), analyze application-aware systems that exploit them (Section 4.2), and map the broader infrastructure landscape (Section 4.3).

4.1 Multi-Agent Serving Challenges

Single-model serving systems (vLLM [36], SGLang [125], DistServe [126]) primarily optimize request execution. Energy-efficient retrieval-augmented generation on resource-constrained devices provides a complementary efficiency perspective outside the multi-agent serving setting [17]. Peer-reviewed serving work has also begun to expose application structure: Parrot [46] passes an application-level dependency graph (semantic variables) to the scheduler, Preble [89] optimizes distributed prompt sharing, Sarathi-Serve [2] introduces chunked prefill, InferCept [1] retains KV state across tool-call interruptions, and S-LoRA [84] and dLoRA [103] serve many concurrent adapters. These systems approach agentic serving from the general side; we return to how they differ from agent-native designs in Section 5. We highlight three workload properties that are particularly relevant to multi-agent execution (cf. the broader KV-cache management landscape surveyed by [41]).

Sparse activation. Only a fraction of agents invoke the LLM at any moment: a 1,000-agent simulation may have fewer than 10 actively generating tokens [72]. Under such workloads, keeping all agent state on the fastest memory tier can waste scarce accelerator memory.

Invocation distance. Many workloads have structured activation patterns such as sequential queues, event-driven messaging, and dependency graphs. The temporal gap before an agent's next call, termed *invocation distance* [72], can be estimated from application semantics in structured workloads but is not generally visible to request-level schedulers.

Inter-agent memory sharing. Agents may share context such as system prompts, environment descriptions, and tool schemas, and general serving systems already reuse such prefixes once they appear: SGLang's RadixAttention and vLLM's automatic prefix caching both do this. The additional opportunity in agentic systems is to use application-level future call structure (dependency graphs, message queues, or execution order) to guide prefetching and eviction. TokenDance [10] proposes collective KV-cache sharing across agents, DroidSpeak [53] enables cache reuse across different model variants, KVFlow [71] exploits an agent step graph to evict workflow-aware and prefetch ahead of the predicted next call, and KVCOMM [108] aligns cache offsets across divergent per-agent prefixes so that cross-agent reuse survives the prefix change (all abstract-verified). Persistent agent-memory systems (for example A-MEM and the Mesh Memory Protocol) address a related but distinct problem, long-term memory representation rather than serving-time cache reuse, and are outside our scope. Concept-graph memory for vision–language adaptation provides another example of structured long-term representation, distinct from serving-time KV-cache management [127].

4.2 Application-Aware Serving

Recent systems show how multi-agent or agentic workload structure can be exploited across four optimization axes: memory optimization, prefix-aware scheduling, cache lifecycle, and power management.

Memory optimization via invocation distance. ScaleSim [72] introduces invocation distance as a first-class scheduling primitive. Two mechanisms exploit predictions: *proactive prefetching* loads agent states before they are needed, and *distance-aware eviction* replaces LRU with future-need predictions, achieving up to 1.74× speedup over SGLang [72]. Its limitation is predictable simulation workloads; interactive agents are less structured.

Prefix-aware scheduling. TokenCake [9] and TOPAS [66] exploit shared prefixes across agents. TokenCake reports 47% latency reduction and 16.9% memory improvement over vLLM (abstract-verified). These figures are reported as stated by the source and are not comparable to ScaleSim's verified up to 1.74×, since the two measure different quantities on different workloads. ForkKV [95] introduces copy-on-write semantics for disaggregated KV caches, which allows efficient prefix sharing without redundant copies (abstract-verified). Prediction-based KV-cache management [124] applies learned policies to anticipate cache needs (abstract-verified).

Cache lifecycle and power management. Continuum [40] introduces KV cache TTL policies for interactive tool-calling agents with unpredictable pauses (abstract-verified). KAIROS [112] introduces context-aware power management adapting GPU power states to agentic workload phases (abstract-verified). Together with ScaleSim and TokenCake, these systems illustrate that multi-agent serving exposes optimization opportunities along multiple axes; in the surveyed set, coverage remains partial rather than end-to-end.

4.3 System Landscape

Table 5 summarizes the broader infrastructure landscape. Beyond the application-aware systems above, infrastructure-aware orchestration (INFRAMIND [33]) bridges scheduling and placement decisions (abstract-verified). Autellix [55] provides a dedicated serving engine for LLM agents (abstract-verified), and policy-driven runtime layers [120] and persistent agent memory [87] address the system-level integration gap. Baseline serving (vLLM [36], SGLang [125], FlexGen [85]) and disaggregated approaches (DistServe [126]) provide the foundation, and Mooncake [78] extends disaggregation to a cluster-wide KV-cache pool, reporting large capacity gains under SLO on production Kimi traffic (abstract-verified). An IEEE survey [74] proposes a four-layer latency optimization taxonomy, and the model-native computing architecture [47] reconsiders the full system stack for foundation model workloads (both abstract-verified).

Synthesis. The recurring pattern is that agent-specific infrastructure gains come from exposing structure that request-level serving would otherwise treat as opaque: future invocation order, shared prefixes, idle phases, and cache lifetime. Evidence remains uneven—ScaleSim is our full-text case study, while most other agent-specific systems are abstract-verified—so we treat the four axes as a map of design directions rather than a performance ranking. Within the surveyed set, we did not identify a system that jointly optimizes all four axes or uses diagnostic signals as a first-class serving input; this motivates the cross-layer analysis in Section 5.

5 Diagnostics–Infrastructure Retention Tension

The two layers surveyed so far are usually discussed as independent engineering problems, but both place requirements on the same runtime state. Diagnostics benefits from state being *retained* long enough to explain a failure; serving infrastructure benefits from reclaiming state when doing so improves memory, latency, or throughput. The organising axis is therefore *retention*, not simply timing: a serving policy determines what runtime evidence remains available, and diagnosis—online or after the fact—must work from what survives. In the surveyed systems, retention decisions are made locally inside the serving or diagnostic component rather than negotiated across layers.

TraceElephant provides the clearest measured example of why retained evidence matters. Under the same All-at-Once attribution protocol, complete traces raise agent-level accuracy from 51% to 62.2%, and step-level accuracy from 16% to 28.1% (a 76% relative gain) [13]. The comparison does not by itself establish why a production system might retain less information,

Table 5: Multi-agent infrastructure systems. [a] = abstract-verified. **Pub.**: ✓ = peer-reviewed venue, ∘ = preprint / no confirmed venue, — = no publication (platform/docs). Pub. records the venue only and is not a quality judgement.

System	Category	Tier	Pub.
ScaleSim	Memory mgmt.	1	∘
TokenCake	KV-cache sched.	2	∘
TOPAS	Prefix-aware	2	∘
KVFlow	Prefix KV-cache	2	∘
ForkKV	Copy-on-write KV	2	∘
Continuum	Cache lifecycle	2	∘
KAIROS	Power mgmt.	2	∘
TokenDance	KV sharing	2	∘
DroidSpeak	Cross-LLM KV	2	∘
Autellix	Agent serving	2	∘
INFRAMIND	Infra-aware	2	∘
KVCOMM	Cross-agent KV	2	✓
Mooncake	KV-cache pool	2	✓
Parrot	App-level DAG	2	✓
InferCept	KV intercept	2	✓
Sarathi-Serve	Chunked prefill	2	✓
Preble	Prompt sharing	2	✓
S-LoRA	Multi-adapter	2	✓
dLoRA	Adapter orch.	2	✓
vLLM	Baseline serving	2	✓
SGLang	Baseline serving	2	✓

but it quantifies the diagnostic consequence of reduced observability. This is the empirical anchor for treating trace retention as a cross-layer resource rather than assuming that diagnostic fidelity is independent of serving-state policy.

We call this the diagnostics–infrastructure retention tension. It is a trade-off, not a claim that every infrastructure optimization necessarily harms diagnosis: the two layers optimize different objectives over overlapping runtime objects, and the conflict appears when reclaiming or compressing those objects removes evidence needed later. Two cross-layer mechanisms make the issue concrete; a third instance shows the same shape within the diagnostic layer itself; and a fourth marks where the conflict would become acute for online diagnosis.

Swapping state off the fast path. ScaleSim's distance-aware eviction [72] prioritizes agent states with long invocation distance for movement off the fastest memory tier. From a diagnostic perspective, those inactive states can still contain history useful for later reconstruction. ScaleSim does not delete the state by definition; the relevant cross-layer question is whether moved state, associated KV cache, and related execution context remain available long enough for diagnosis or are later reclaimed or recomputed. The surveyed serving policy optimizes reuse and memory pressure, not diagnostic value, which exposes the difference in objectives without implying that eviction always destroys evidence.

Environment snapshots are not a first-class serving primitive. Replay-based attribution needs more than a text trace. FAMAS [24] re-executes a failed task $k = 20$ times and expects the environment to reach comparable states, and DoVer [57] intervenes mid-trajectory and reruns from the intervention point, which requires the environment to be restorable. The serving systems surveyed here do not expose environment-snapshot retention as a diagnostic primitive. Consequently, replay-based methods are typically evaluated on fixed corpora or reconstructable environments rather than under a live serving policy with an explicit snapshot budget.

The diagnostic layer has an isomorphic tradeoff internally. TrajAudit's Semantic Saliency Folding [94] compresses 86–92% of observations into placeholders to fit the model's context window. This is the diagnostician's own retention decision, with more context potentially

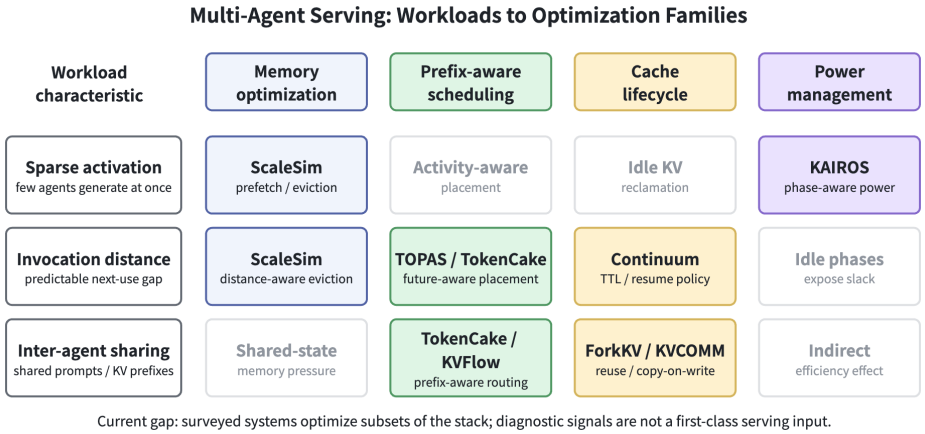


Figure 4: Multi-agent serving challenges and optimization axes. Three workload characteristics—sparse activation, invocation distance, and inter-agent memory sharing—motivate four families of application-aware systems spanning memory optimization, prefix-aware scheduling, cache lifecycle, and power management. The figure maps coverage in the surveyed set; it is not a claim that one system should optimize every axis simultaneously.

Table 6: A stylized contrast between diagnostic and infrastructure tendencies over shared runtime resources. These are design tendencies in the surveyed literature, not universal requirements.

Dimension	**Diagnostics tendency**	**Infrastructure tendency**
Runtime state	Benefits from richer per-step inputs, outputs, and tool logs	Retains state needed for serving or expected reuse
Retention horizon	Benefits from retaining evidence until diagnosis	Reclaims state when expected reuse or value becomes low
Eviction criterion	May value diagnostically informative low-reuse state	Often prioritizes reuse probability and memory pressure
Granularity	Often step-level and cross-agent (to localise cause)	Often request- and agent-level (to place work)
Overhead budget	May tolerate replay or richer capture, especially offline	Constrained by latency, throughput, and memory SLOs
Replay resources	May require restorable environment snapshots	Typically does not retain snapshots for diagnosis

supporting attribution and compression lowering cost. It is not a cross-layer conflict, but it shows that evidence retention already appears as a cost–fidelity trade-off inside diagnostics.

Replay is a serving cost: a boundary condition. The replay budget is distinct from the restorable environment required above: a method could retain a restorable environment and still replay only once. FAMAS's twenty replays conflict with serving only if diagnosis runs on the serving path. Because FAMAS is evaluated offline on a fixed corpus, the conflict is latent rather than actual; it marks where the tension would become acute for an online diagnostic. The existence of zero-replay debugging [34] is consistent with replay cost becoming important when diagnosis must run closer to execution time.

Table 6 contrasts the two layers along the dimensions that generate these conflicts. The rows are not independent: retention budget is the shared resource, and each mechanism above is a different way of spending it. Among the surveyed systems, we did not identify one that treats diagnostic value and serving value within a shared retention budget; the four co-design directions that follow from this view are developed in Section 6.

Observed instances. Cross-layer: ScaleSim's eviction moves inactive state off the fast path [72]; surveyed serving systems do not expose the environment snapshots needed by replay-based attribution [24, 57]. Intra-layer

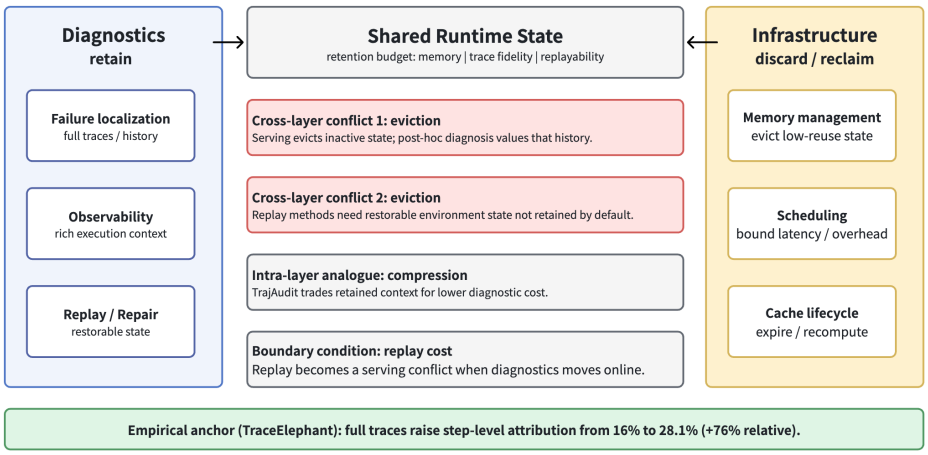


Figure 5: The diagnostics–infrastructure retention tension. The diagnostics layer (left) benefits from retaining runtime evidence for failure explanation, while the infrastructure layer (right) reclaims state under memory, latency, and throughput budgets. The centre summarizes two cross-layer conflict mechanisms, one intra-layer analogue, and one boundary condition. Among the surveyed systems, none manages retention as an explicit cross-layer budget.

analogue: TrajAudit compresses 86–92% of observations [94]. Boundary: replay (k=20) becomes a serving concern when diagnosis runs online [24]. Continuum's TTL expiry [40] raises resumption cost within the cache layer, not across the diagnostics–infrastructure boundary. Reduced observability in TraceElephant lowers step-level attribution from 28.1% to 16% under the same protocol [13].

6 Discussion and Research Directions

This survey has treated diagnostics and infrastructure as two bodies of work and, in Section 5, as two sets of requirements acting on overlapping runtime objects. This section discusses why the two literatures remain weakly connected, where their assumptions differ, what the current evidence cannot resolve, and what a research agenda grounded in the retention tension would look like.

Why the two layers remain weakly connected. Part of the separation reflects community boundaries. Diagnostics draws on software-engineering fault localization and testing, where the unit of analysis is an execution trace, and on machine-learning work at ICLR, ICML, NeurIPS, and ACL. Infrastructure draws more heavily on systems and architecture, where the unit is a request or resource allocation decision and the goal is throughput and latency under a budget. In our surveyed set, these communities show little direct overlap: the serving systems in Section 4 do not evaluate on diagnostic benchmarks, while the failure-localization paradigms in Section 3.1 largely compare within their own evaluation conventions. Their cost models differ as well: an offline diagnostic can afford replay or rich trace processing that would be difficult to place on a latency-sensitive serving path. The resulting disconnect is therefore both methodological and architectural, not evidence that the communities are universally disjoint.

Contested questions. Several unresolved choices recur across the literature.

Should attribution be statistical, reasoning-based, or learned? The three paradigms in Section 3.1 embody different assumptions. Statistical approaches expect reproducible failure signatures across replays; reasoning-based approaches ask a model to infer cause from a trace; learned approaches train the mapping directly. The paradigms increasingly touch overlapping

benchmarks such as Who&When, but their input costs, granularity, and observability assumptions still differ. Their reported headline numbers are therefore not commensurable, and the evidence does not support a universal ranking.

How much execution evidence should be retained? TraceElephant shows that observability materially changes attribution quality, with step-level accuracy especially sensitive to trace completeness [13]. Serving systems, by contrast, operate under memory, bandwidth, and latency budgets that make unlimited trace retention unattractive. The literature does not yet define a common minimum set of runtime evidence that should survive for diagnosis, nor how that minimum should vary with cost.

Is retention an engineering detail or an architectural resource? Current surveyed systems treat retention locally: as an eviction rule, cache TTL, trace-compression choice, or logging policy. Our synthesis proposes a different abstraction: retention as a shared budget whose value differs across layers. This is a research hypothesis motivated by the surveyed mechanisms, not an experimentally established design principle.

Should safety intervene proactively or reactively? The safety paradigms in Section 3.3 act at different points in execution. Proactive approaches need enough context to judge an action before it executes; reactive approaches accept some delay in exchange for lower continuous overhead. The surveyed papers do not evaluate these choices under a common workload and serving-cost budget, so the trade-off remains empirical rather than settled. More generally, robustness evaluation under distribution shift can be sensitive to structural calibration, partition matching, rejection behavior, and non-IID data harmonization, as illustrated in adjacent graph-based and federated-learning studies [15, 131–133].

The safety layer as a third constraint. Safety is often grouped with diagnostics, and we follow that convention in the taxonomy, but it behaves differently from purely post-hoc attribution. A guardrail must inspect runtime state and may also intervene on the serving path, so it is simultaneously sensitive to diagnostic fidelity and latency overhead. Multi-agent settings sharpen this issue because the relevant context can span a communication graph rather than a single request. This suggests that safety may be best viewed as an interface between diagnostics and infrastructure, even though the present literature does not yet provide enough comparative evidence to reorganize the taxonomy around that view.

Open problems from the retention view. The retention framing turns several observations into measurable questions. A future system could treat retained state as a budgeted quantity, allocate it per agent or trajectory, assign different value to different evidence, and release it according to both reuse probability and diagnostic importance. We did not identify such a cross-layer policy in the surveyed set. The nearest mechanisms are local cache-retention and trace-compression policies, which optimize one side of the trade-off at a time. Testing whether a shared budget improves the joint reliability–efficiency frontier is therefore an open systems problem rather than a conclusion of this review.

Three gaps are especially actionable. First, *interaction benchmarks are missing from the surveyed corpus*: current diagnostics benchmarks measure attribution quality on fixed traces or replayable environments, while serving evaluations measure latency, throughput, memory, or power. A benchmark that jointly varies retention policy and diagnostic fidelity would connect the two. Second, *capture standards do not imply retention standards.* OpenTelemetry conventions [69] and structured-logging proposals [4] help specify what may be recorded, but not how long evidence should remain available or under what budget. Third, *evaluation scale differs sharply across the two literatures.* Diagnostics studies often use tens to hundreds of failures, whereas serving systems target sustained production-scale workloads. Whether diagnostic gains persist at that scale, and what retention cost is required to preserve them, remains under-measured.

A concrete validation protocol. The retention hypothesis can be tested without first building a complete production serving stack. Starting from a benchmark with full traces, such as TraceElephant, one can define controlled retention policies that progressively remove, expire, or compress evidence at fixed budgets (for example, by agent, step age, or evidence type), then measure agent- and step-level attribution accuracy under the same diagnostic method. A replay-capable or instrumented serving harness can record the corresponding storage footprint, memory pressure, I/O, and latency overhead. Plotting diagnostic fidelity against these serving costs would produce the joint reliability–efficiency frontier that is currently missing. The same protocol can then compare naive retention policies with policies that use reuse probability, diagnostic importance, or both.

To make such experiments comparable across future work, we recommend reporting at least four quantities together: (1) diagnostic fidelity at agent and step granularity; (2) retained-state volume or retention ratio; (3) latency/throughput overhead under the same workload; and (4) replay or recomputation cost. Reporting these jointly would prevent systems from appearing superior merely because they optimize one layer while externalizing cost to the other.

Evidence and its limits. The evidence coding in Section 2 intentionally separates claim verification from publication venue. The result is uneven depth across the taxonomy: failure localization contains several full-text-verified quantitative cases, whereas much of the observability, safety, and agent-specific infrastructure landscape is abstract-verified. We therefore use quantitative values to illustrate source-specific results and matched comparisons, not to construct a global leaderboard. Readers should treat the taxonomy as broader than the set of claims that support quantitative synthesis.

The field is also unusually recent. Many surveyed systems are preprints or 2026 papers, so details can change between revisions and venue versions. This does not invalidate structural observations such as the coexistence of replay-heavy diagnostics and state-reclaiming infrastructure, but it makes claims about which individual method performs best especially fragile. For that reason, we avoid ranking language where evaluation conditions differ.

Broader impact. Retention has a privacy cost as well as a systems cost. Execution state can contain user inputs, retrieved knowledge, tool outputs, credentials, intermediate reasoning, and other sensitive material; retrieval pipelines can additionally encounter conflicts among knowledge sources that require explicit handling [109]; richer retention extends the lifetime and attack surface of that information. Predictive infrastructure mechanisms may also expose behavioral patterns about agent activity. The same policies proposed to improve diagnosis should therefore include bounded retention windows, access control, encryption where appropriate, and auditability [68]. A useful retention budget must price privacy and governance constraints alongside memory and latency, not treat them as an afterthought.

Limitations. Four limitations bound this review. First, it is a curated critical review rather than a systematic review: the inclusion rules are explicit, but the corpus was not produced by a preregistered database search with fixed queries, date windows, and PRISMA-style screening, so it should not be interpreted as exhaustive. Second, the reviewed studies use heterogeneous benchmarks, observability settings, baselines, and outcome metrics; this supports structured mapping and within-source comparison, but not a global performance ranking. Third, evidence depth is uneven. ScaleSim is the principal full-text infrastructure case study, while many observability, safety, and agent-specific serving systems were examined at abstract level; conclusions that depend on those branches are therefore qualitative. Fourth, the review deliberately focuses on the runtime stack. Orchestration design, general agent architecture, offline task-quality evaluation, and long-term memory representation are included only when they directly inform runtime

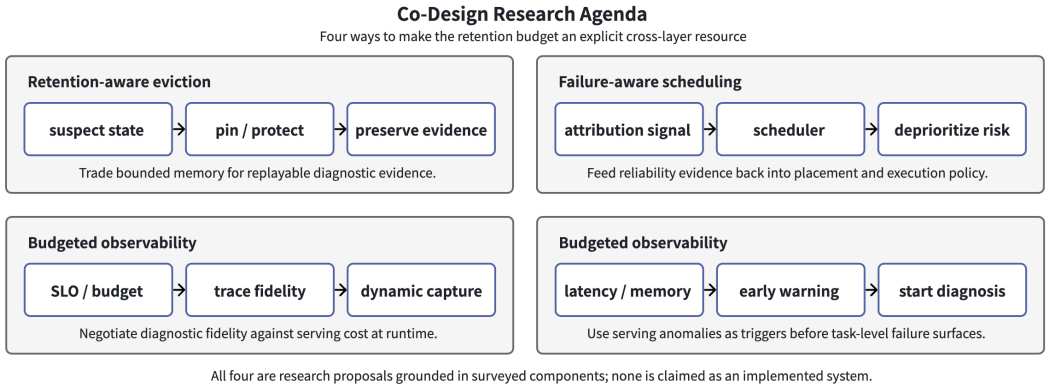


Figure 6: Co-design research agenda derived from the diagnostics–infrastructure tension. The four directions translate the tension dimensions summarized in Table 6 into testable research questions at the boundary between the two layers.

diagnostics or infrastructure. These limits narrow the strength and scope of the conclusions, but they do not alter the central purpose of the review: to identify cross-layer questions that become visible only when the two operational literatures are read together.

Future development. *Retention-aware eviction* targets the retention-horizon and eviction-criterion dimensions in Table 6. A serving policy could preserve state that an online detector marks as diagnostically valuable even when predicted reuse is low, then evaluate whether the additional memory cost improves later attribution or repair.

Failure-aware scheduling extends the runtime-state and granularity dimensions by feeding reliability evidence back into placement or execution priority. A useful evaluation would compare latency and throughput against failure containment or recovery under the same workload, rather than reporting either systems efficiency or diagnostic quality alone.

Budgeted observability directly addresses the granularity and overhead-budget dimensions. Instead of fixing trace fidelity at design time, the system would vary capture detail under an explicit serving SLO and test whether adaptive capture preserves diagnostic accuracy more efficiently than uniform logging.

Infrastructure-triggered diagnostics uses serving-side anomalies—for example latency spikes, memory pressure, or repeated cache churn—as early-warning signals for diagnosis. Its value would be measured by whether earlier intervention improves reliability enough to justify continuous monitoring cost. Across all four directions, the common requirement is to report the joint reliability–efficiency frontier under a shared workload and budget rather than optimize one side of the trade-off in isolation.

7 Conclusion

Foundation model-based multi-agent systems increasingly depend on two operational layers that have largely developed apart: diagnostics, which explains and constrains failures, and infrastructure, which keeps execution efficient. This review brings them into one runtime-stack view, organizing diagnostics into failure localization, observability, safety, and benchmarks, and infrastructure into memory optimization, prefix-aware scheduling, cache lifecycle, and power management. The purpose is not to construct a leaderboard across heterogeneous systems, but to expose which assumptions each layer makes about runtime state.

The main synthesis is that diagnostics and infrastructure act on overlapping runtime objects

under different objectives. Diagnostics benefits from preserving traces, context, and replayable state; serving infrastructure benefits from reclaiming or compressing state under resource budgets. The surveyed evidence makes this retention tension concrete through eviction, snapshot availability, trace compression, and replay cost, with TraceElephant providing a measured example of the sensitivity of attribution to observability. No surveyed system currently manages diagnostic value and serving value through a shared retention budget. Treating retention as an explicit cross-layer resource therefore defines a concrete research program rather than a settled result.

For future work, the most important methodological step is to evaluate reliability and efficiency jointly. Retention policies should be compared under a common workload using diagnostic fidelity, retained-state volume, serving overhead, and replay/recomputation cost. Such evaluation would make it possible to test whether retention-aware eviction, failure-aware scheduling, budgeted observability, and infrastructure-triggered diagnostics improve the joint reliability–efficiency frontier. In this sense, the contribution of the review is not only a taxonomy of recent systems, but a systems-level abstraction and validation agenda for the next generation of foundation-model multi-agent runtimes.

Acknowledgements

The authors acknowledge the use of AI-assisted tools for language editing and manuscript preparation. All technical claims, interpretations, citations, and conclusions were reviewed and remain the responsibility of the authors.

Competing interests

The authors declare that they have no competing interests or financial conflicts to disclose.

References

[1] Reyna Abhyankar, Zijian He, Vikranth Srivatsa, Hao Zhang, and Yiying Zhang. InferCept: Efficient intercept support for augmented large language model inference. In *ICML*, 2024.

[2] Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, et al. Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve. In *OSDI*, pages 117–134, 2024.

[3] Mohamad Abou Ali and Fadi Dornaika. Agentic AI: A comprehensive survey of architectures, applications, and future directions. *Artificial Intelligence Review*, 2025.

[4] Adam AlSayyad, Kelvin Yuxiang Huang, and Richik Pal. AgentTrace: Structured logging for LLM agent systems. In *AAAI Workshop*, 2026.

[5] Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Eric Winsor, Jerome Wynne, Yarin Gal, and Xander Davies. AgentHarm: A benchmark for measuring harmful agent behaviors. *arXiv preprint arXiv:2410.09024*, 2024.

[6] Adi Banerjee, Anirudh Nair, and Tarik Borogovac. Where did it all go wrong? hierarchical error attribution in multi-agent systems. *arXiv preprint arXiv:2510.04886*, 2025.

[7] Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. AgentRx: Diagnosing AI agent failures from execution trajectories. *arXiv preprint arXiv:2602.02475*, 2026.

[8] Saddek Bensalem, Yi Dong, Martin Fränzle, Xiaowei Huang, Janis Kröger, Dejan Nickovic, Ayoub Nouri, Rajarshi Roy, and Changshun Wu. Position: A three-layer probabilistic assume-guarantee architecture is structurally required for safe LLM agent deployment. *arXiv preprint arXiv:2605.18672*, 2026.

[9] Zhuohang Bian, Feiyang Wu, Teng Ma, and Youwei Zhuo. TokenCake: A KV-cache-centric serving framework for LLM-based multi-agent applications. *arXiv preprint arXiv:2510.18586*, 2025.

[10] Zhuohang Bian, Feiyang Wu, Chengrui Zhang, et al. TokenDance: Scaling multi-agent LLM serving via collective KV cache sharing. *arXiv preprint arXiv:2604.03143*, 2026.

[11] Shaunak Biswas, Hiya Bhatt, and Karthik Vaidhyanathan. Architecting AgentOps: Design patterns for LLM agent operations. *arXiv preprint arXiv:2601.06456*, 2026.

[12] Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, et al. Why do multi-agent LLM systems fail? In *NeurIPS Datasets and Benchmarks Track (Spotlight)*, 2025.

[13] Mengzhuo Chen, Junjie Wang, Fangwen Mu, et al. Seeing the whole elephant: A benchmark for failure attribution in LLM-based multi-agent systems. *arXiv preprint arXiv:2604.22708*, 2026.

[14] Weize Chen, Ziming You, Ran Li, Yitong Guan, Chen Qian, Chenyang Zhao, Cheng Yang, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Internet of agents: Weaving a web of heterogeneous agents for collaborative intelligence. In *ICLR*, 2025.

[15] Xiaohong Chen, Canran Xiao, and Yongmei Liu. Confusion-resistant federated learning via diffusion-based data harmonization on non-iid data. *Advances in Neural Information Processing Systems*, 37:137495–137520, 2024.

[16] Zhaorun Chen, Mintong Kang, and Bo Li. ShieldAgent: Shielding agents via verifiable safety policy reasoning. In *ICML*, PMLR 267, pages 8313–8344, 2025.

[17] Zhiyuan Cheng, Longying Lai, Yue Liu, and Yu Sun. Toward sustainable on-device intelligence: A survey on energy-efficient rag systems with small language models. *Available at SSRN 6698538*, 2026.

[18] Sahana Chennabasappa, Cyrus Nikolaidis, et al. LlamaFirewall: An open source guardrail system for building secure AI agents. *arXiv preprint arXiv:2505.03574*, 2025.

[19] Pierre Dantas, Lucas Cordeiro, Ehsan Nowroozi, and Tihanyi Norbert. Toward safe LLM agents: A survey of specification, verification, and enforcement. *arXiv preprint arXiv:2608.14590*, 2026.

[20] Darshan Deshpande, Varun Gangal, et al. TRAIL: Trace reasoning and agentic issue localization. *arXiv preprint arXiv:2505.08638*, 2025.

[21] Liming Dong, Qinghua Lu, and Liming Zhu. AgentOps: Enabling observability of LLM agents. *arXiv preprint arXiv:2411.05285*, 2024.

[22] Will Epperson, Gagan Bansal, et al. Interactive debugging and steering for multi-agent systems. In *CHI*, 2025.

[23] Fabiana Fournier, Lior Limonad, and Yuval David. Agentic AI process observability via process mining. *arXiv preprint arXiv:2505.20127*, 2025.

[24] Yu Ge, Linna Xie, Zhong Li, Yu Pei, and Tian Zhang. Who is introducing the failure? automatically attributing failures of multi-agent systems via spectrum analysis. *arXiv preprint arXiv:2509.13782*, 2025.

[25] Aayush Gupta. ReliabilityBench: Evaluating LLM agent reliability under production-like stress conditions. *arXiv preprint arXiv:2601.06112*, 2026.

[26] Xu He, Di Wu, Yan Zhai, and Kun Sun. SentinelAgent: Graph-based anomaly detection in multi-agent systems. *arXiv preprint arXiv:2505.24201*, 2025.

[27] Sirui Hong, Mingchen Zhuge, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In *ICLR*, 2024.

[28] Yuxuan Jiang and Francis Ferraro. Beyond math: Stories as a testbed for memorization-constrained reasoning in LLMs. In Vera Demberg, Kentaro Inui, and Lluís Marquez, editors, *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5590–5607, Rabat, Morocco, March 2026. Association for Computational Linguistics.

[29] Yuxuan Jiang and Francis Ferraro. Bridging reasoning trajectories in on-policy distillation via near-future guidance, 2026.

[30] Yuxuan Jiang and Francis Ferraro. SCRIBE: Structured mid-level supervision for tool-using language models, 2026.

[31] Yuxuan Jiang, Dawei Li, and Francis Ferraro. DRP: Distilled reasoning pruning with mathematical skill-aware step decomposition for efficient large reasoning models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors, *Findings of the Association for Computational Linguistics: ACL 2026*, pages 4020–4039, San Diego, California, United States, July 2026. Association for Computational Linguistics.

[32] Yuxuan Jiang, Runchao Li, Shubhashis Roy Dipta, Dawei Li, and Zhao Yang. Cornerstones or stumbling blocks? deciphering the rock tokens in on-policy distillation, 2026.

[33] Ahasan Kabir, Jiaqi Xue, Mengxin Zheng, and Qian Lou. INFRAMIND: Infrastructure-aware multi-agent orchestration. *arXiv preprint arXiv:2606.11440*, 2026.

[34] Dong Ho Kang, Hyeonjeong Cha, and Daein Weon. Knowledge-based zero-replay debugging for multi-agent systems. *arXiv preprint arXiv:2606.14805*, 2026.

[35] Ishan Kavathekar, Hemang Jain, Ameya Rathod, Ponnurangam Kumaraguru, and Tanuja Ganu. TAMAS: Benchmarking adversarial risks in multi-agent LLM systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 31238–31268, 2026.

[36] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

[37] LangChain. LangSmith: Production observability platform, 2026.

[38] Chaofan Li, Lyuye Zhang, Jintao Zhai, Siyue Feng, Xichun Yang, Huahao Wang, Shihan Dou, Yu Ji, Yutao Hu, Yueming Wu, Yang Liu, and Deqing Zou. Towards security-auditable LLM agents: A unified graph representation. *arXiv preprint arXiv:2605.06812*, 2026.

[39] Guohao Li, Hasan Abed Al Kader Hammoud, et al. CAMEL: Communicative agents for "mind" exploration of large language model society. In *NeurIPS*, 2023.

[40] Hanchen Li, Runyuan He, Qiuyang Mang, Qizheng Zhang, Huanzhi Mao, Xiaokun Chen, Hangrui Zhou, Alvin Cheung, Joseph Gonzalez, and Ion Stoica. Continuum: Efficient and robust multi-turn LLM agent scheduling with KV cache time-to-live. *arXiv preprint arXiv:2511.02230*, 2025.

[41] Jie Li, Tongyang Wang, and Yong Chen. From tensor buffer to distributed memory hierarchy: A survey of KV cache management for LLM serving. *arXiv preprint arXiv:2607.02574*, 2026.

[42] Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. A survey on LLM-based multi-agent systems: Workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9, 2024.

[43] Yuanhao Li, Mingshan Liu, Hongbo Wang, Yiding Zhang, Yifei Ma, and Wei Tan. DRAFT-RL: Multi-Agent Chain-of-Draft Reasoning for Reinforcement Learning-Enhanced LLMs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(35):29530–29537, 2026.

[44] Yuanhao Li, Hongbo Wang, Xuhong Chen, and Yiming Cao. Retrieval-Augmented Multi-Agent Multimodal Framework for Fake News Detection. In *ICASSP 2026*, 2026.

[45] Yuanhao Li, Hongbo Wang, Xuhong Chen, Yiming Cao, and Xunzhu Tang. Counterfactual Rollout Replay: Forkable Environments as Free Process Rewards for Software Engineering Agents, 2026. Accepted at NeurIPS 2026.

[46] Chaofan Lin, Chen Chen, Zhenhua Han, Chengruidong Zhang, Yuqing Yang, et al. Parrot: Efficient serving of LLM-based applications with semantic variable. In *OSDI*, pages 929–945, 2024.

[47] Haishan Lin, Hoilam Pao, Shaoxiong Zhan, and Hai-Tao Zheng. Model-native computing architecture: Envisioning future system architecture through the lens of computer architecture. *arXiv preprint arXiv:2606.00288*, 2026.

[48] Zhiming Lin, Kai Zhao, Sophie Zhang, Peilai Yu, and Canran Xiao. CEC-Zero: Zero-supervision character error correction with self-generated rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 23612–23620, 2026.

[49] Dongrui Liu, Yu Li, Zhonghao Yang, et al. AgentDoG 1.5: A lightweight and scalable alignment framework for AI agent safety and security. *arXiv preprint arXiv:2605.29801*, 2026.

[50] Jiale Liu, Huajun Xi, et al. Who&When Pro: Meta-evaluation of failure attribution in multi-agent systems. *arXiv preprint arXiv:2607.09996*, 2026.

[51] Mingchao Liu, Yu Sun, Ruixiao Sun, Xin Dong, Xiang Shen, Hongwei Wang, Hongyu Xiong, and Yang Song. IPS: In-prompt process supervision for short video content moderation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 1277–1288, 2026.

[52] Yang Liu et al. MASPrism: Lightweight failure attribution for multi-agent systems using prefill-stage signals. *arXiv preprint arXiv:2605.07509*, 2026.

[53] Yuhan Liu, Yuyang Huang, Jiayi Yao, Shaoting Feng, Zhuohan Gu, Kuntai Du, Hanchen Li, Yihua Cheng, Junchen Jiang, Shan Lu, Madan Musuvathi, and Esha Choukse. DroidSpeak: KV cache sharing across LLMs. *arXiv preprint arXiv:2411.02820*, 2024.

[54] Hanjun Luo, Shenyu Dai, Chiming Ni, Xinfeng Li, Gui-Min Zhang, Kun Wang, Tongliang Liu, and Hanan Salam. AgentAuditor: Automated auditing of LLM agent safety. In *NeurIPS*, 2025.

[55] Michael Luo, Xiaoxiang Shi, Colin Cai, Tianjun Zhang, Justin Wong, Yichuan Wang, Chi Wang, Yanping Huang, Zhifeng Chen, Joseph E. Gonzalez, and Ion Stoica. Autellix: An efficient serving engine for LLM agents. *arXiv preprint arXiv:2502.13965*, 2025.

[56] Weidi Luo, Shenghong Dai, Xiaogeng Liu, Suman Banerjee, Huan Sun, Muhao Chen, and Chaowei Xiao. AGrail: A lifelong agent guardrail with effective and adaptive safety detection. *arXiv preprint arXiv:2502.11448*, 2025.

[57] Ming Ma, Jue Zhang, Fangkai Yang, Yu Kang, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. DoVer: Intervention-driven auto debugging for LLM multi-agent systems. In *ICLR*, 2026.

[58] Xuyan Ma, Yawen Wang, et al. Diagnosis-driven automatic repair for multi-agent systems. *arXiv preprint arXiv:2607.02882*, 2026.

[59] Netta Madvil, Gilad Dym, et al. Holistic evaluation and failure diagnosis for LLM agents. *arXiv preprint arXiv:2605.14865*, 2026.

[60] Aayush Manglik et al. Insights generator: Systematic corpus-level trace diagnostics for LLM agents. *arXiv preprint arXiv:2605.21347*, 2026.

[61] Rui Miao, Yixin Liu, Yili Wang, Xu Shen, Yue Tan, Yiwei Dai, Shirui Pan, and Xin Wang. BlindGuard: Safeguarding LLM-based multi-agent systems under unknown attacks. *arXiv preprint arXiv:2508.08127*, 2025.

[62] Lesly Miculicich, Mihir Parmar, Hamid Palangi, Krishnamurthy Dj Dvijotham, Mirko Montanari, Tomas Pfister, and Long T. Le. VeriGuard: Enhancing LLM agent safety via verified code generation. *arXiv preprint arXiv:2510.05156*, 2025.

[63] Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. Evaluation and benchmarking of LLM agents: A survey. In *Proceedings of the 31st ACM SIGKDD Conference*, 2025.

[64] Mina Mohammadmirzaei and Jeffrey Flanigan. OSGuard: A benchmark for safety in computer-use agents. *arXiv preprint arXiv:2606.15034*, 2026.

[65] Vishal Ishwar Naik, Chenyu Xu, Donna Dong, et al. GuardianAgentBench: Where agents fail and how to guard them. *arXiv preprint arXiv:2607.20982*, 2026.

[66] Hongqiu Ni, Han Tian, Chi Zhang, Guopeng Li, and Haisheng Tan. TOPAS: Task-oriented prefix-aware scheduling for multi-agent serving. *arXiv preprint arXiv:2608.25523*, 2026.

[67] Shiwen Ni, Guhong Chen, Shuaimin Li, Xuanang Chen, Siyi Li, Bingli Wang, Qiyao Wang, Xingjian Wang, Yifan Zhang, Liyang Fan, Chengming Li, Ruifeng Xu, Le Sun, and Min Yang. A survey on large language model benchmarks. *arXiv preprint arXiv:2508.15361*, 2025.

[68] Yi Nian, Aojie Yuan, et al. Auditable agents: Unified graph for security auditing. *arXiv preprint arXiv:2604.05485*, 2026.

[69] OpenTelemetry Authors. OpenTelemetry generative AI semantic conventions, 2026.

[70] Abdelghny Orogat, Ana Rostam, and Essam Mansour. Understanding multi-agent LLM frameworks: A unified benchmark and experimental analysis. *arXiv preprint arXiv:2602.03128*, 2026.

[71] Zaifeng Pan, Ajjkumar Patel, Zhengding Hu, Yipeng Shen, Yue Guan, Wan-Lu Li, Lianhui Qin, Yida Wang, and Yufei Ding. KVFlow: Prefix KV-caching for multi-agent systems. *arXiv preprint arXiv:2507.07400*, 2025.

[72] Zaifeng Pan, Yipeng Shen, Zhengding Hu, et al. ScaleSim: Scaling large multi-agent simulations via distance-guided proactive state management. *arXiv preprint arXiv:2601.21473*, 2026.

[73] Ravi Pandya, Madison Bland, Duy P. Nguyen, et al. From refusal to recovery: A control-theoretic approach to generative AI guardrails (ReGuard). *arXiv preprint arXiv:2510.13727*, 2025.

[74] Gyeongmuk Park, Seonghyeon Lee, and Yeonsu Park. Minimizing response latency in LLM-based agent systems: A comprehensive survey. *IEEE Access*, 14:26140–26168, 2026.

[75] Shihao Qi, Jie Ma, et al. Beyond individual intelligence: A survey on collaboration and failure in multi-agent systems. *arXiv preprint arXiv:2605.14892*, 2026.

[76] Chen Qian, Wei Liu, et al. ChatDev: Communicative agents for software development. In *ACL*, 2024.

[77] Hezhe Qiao, Hanghang Tong, et al. VerifyMAS: Hypothesis verification for failure attribution in multi-agent systems. *arXiv preprint arXiv:2605.17467*, 2026.

[78] Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Feng Ren, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: Trading more storage for less computation — a KVCache-centric architecture for serving LLM chatbot. In *Proceedings of the 23rd USENIX Conference on File and Storage Technologies (FAST)*, pages 155–170, 2025.

[79] Yangyang Qu, Michele Panariello, Massimiliano Todisco, and Nicholas Evans. Reference-free adversarial sex obfuscation in speech. In *2025 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 2128–2133, 2025.

[80] Yangyang Qu, Michele Panariello, Massimiliano Todisco, and Nicholas Evans. Partial accent-control editing in frozen speech representations for accent conversion, 2026.

[81] Yangyang Qu, Massimiliano Todisco, and Nicholas Evans. Phoneme-aware pronunciation representations for l2-english l1-background accent identification, 2026.

[82] Md Nakhla Rafi, Md Ahasanuzzaman, Dong Jae Kim, Zhijie Wang, and Tse-Hsun Chen. FALAT: Tracing failures in LLM agent trajectories via dependency-guided search. *arXiv preprint arXiv:2606.00765*, 2026.

[83] Md Shamsujjoha, Qinghua Lu, Dehai Zhao, and Liming Zhu. Swiss cheese model for AI safety: A taxonomy and reference architecture for multi-layered guardrails of foundation model based agents. In *IEEE ICSA*, pages 37–48, 2025.

[84] Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, et al. S-LoRA: Serving thousands of concurrent LoRA adapters. In *MLSys*, 2024.

[85] Ying Sheng, Lianmin Zheng, Binhang Yuan, et al. FlexGen: High-throughput generative inference of large language models with a single GPU. *arXiv preprint arXiv:2303.06865*, 2023.

[86] Ruxue Shi, Yili Wang, et al. SAIGuard: Communication-state simulation for multi-agent safety. *arXiv preprint arXiv:2606.12474*, 2026.

[87] Yakov Pyotr Shkolnikov. Agent memory below the prompt: Persistent Q4 KV cache for multi-agent LLM inference on edge devices. *arXiv preprint arXiv:2603.04428*, 2026.

[88] Twinkll Sisodia. AI observability: A multi-layer survey. *arXiv preprint arXiv:2604.26152*, 2026.

[89] Vikranth Srivatsa, Zijian He, Reyna Abhyankar, Dongmin Li, and Yiying Zhang. Preble: Efficient distributed prompt scheduling for LLM serving. In *ICLR*, 2025.

[90] Yu Sun, Yin Li, Ruixiao Sun, Chunhui Liu, Fangming Zhou, Ze Jin, Linjie Wang, Xiang Shen, Zhuolin Hao, and Hongyu Xiong. Audio-enhanced vision-language modeling with latent space broadening for high quality data expansion. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pages 4872–4881. Association for Computing Machinery, 2025.

[91] Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, et al. Multi-agent collaboration mechanisms: A survey of LLMs. *arXiv preprint arXiv:2501.06322*, 2025.

[92] Xinming Tu, Tianze Wang, Yingzhou Lu, Kexin Huang, Yuanhao Qu, and Sara Mostafavi. BenchGuard: Who guards the benchmarks? automated auditing of LLM agent benchmarks. *arXiv preprint arXiv:2604.24955*, 2026.

[93] Kun Wang, Guibin Zhang, Zhenhong Zhou, Jiahao Wu, Miao Yu, Shiqian Zhao, et al. A comprehensive survey in LLM(-agent) full stack safety: Data, training and deployment. *arXiv preprint arXiv:2504.15585*, 2025.

[94] Minxing Wang, Xiaofei Xie, and Yintong Huo. TrajAudit: Automated failure diagnosis for agentic coding systems. *arXiv preprint arXiv:2605.26563*, 2026.

[95] Shaoqi Wang, Rui Ren, and Lin Gui. ForkKV: Copy-on-write disaggregated KV cache. *arXiv preprint arXiv:2604.06370*, 2026.

[96] Shilong Wang, Guibin Zhang, Miao Yu, Guancheng Wan, Fanci Meng, Chongye Guo, Kun Wang, and Yang Wang. G-Safeguard: A topology-guided security lens and treatment on LLM-based multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7261–7276, 2025.

[97] Yawen Wang, Wenjie Wu, Junjie Wang, and Qing Wang. From flat logs to causal graphs: Hierarchical failure attribution for LLM-based multi-agent systems. *arXiv preprint arXiv:2602.23701*, 2026.

[98] Yiqi Wang, Jiaqi Zhang, Taotao Cai, et al. From Agent traces to trust: A survey of evidence tracing and execution provenance in LLM agents. *arXiv preprint arXiv:2606.04990*, 2026.

[99] Zexin Wang, Jingjing Li, Quan Zhou, et al. A survey on AgentOps: Categorization, challenges, and future directions. *arXiv preprint arXiv:2508.02121*, 2025.

[100] Zexin Wang, Changhua Pei, et al. Agent system operations: Lifecycle management for LLM agents. *arXiv preprint arXiv:2606.01581*, 2026.

[101] Zixiang Wang, Mengjia Gong, Qiyu Sun, et al. Multi-agent systems: From classical paradigms to large foundation model-enabled futures. *arXiv preprint arXiv:2604.18133*, 2026.

[102] Zixuan Wang, Yu Sun, Hongwei Wang, Baoyu Jing, Xiang Shen, Xin Dong, Zhuolin Hao, Hongyu Xiong, and Yang Song. Reasoning-enhanced domain-adaptive pretraining of multimodal large language models for short video content governance. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1104–1112. Association for Computational Linguistics, 2025.

[103] Bingyang Wu, Ruidong Zhu, Zili Zhang, Peng Sun, Xuanzhe Liu, and Xin Jin. dLoRA: Dynamically orchestrating requests and adapters for LoRA LLM serving. In *OSDI*, pages 911–927, 2024.

[104] Yufei Xia et al. Who broke the system? failure localization in LLM-based multi-agent systems. In *COLM*, 2026.

[105] Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. GuardAgent: Safeguard LLM agents by a guard agent via knowledge-enabled reasoning. *arXiv preprint arXiv:2406.09187*, 2024.

[106] Canran Xiao, Jiabao Dou, Zhiming Lin, Zong Ke, and Liwei Hou. From points to coalitions: hierarchical contrastive shapley values for prioritizing data samples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 15995–16003, 2026.

[107] Yingfan Xu, Tieming Liu, Ye Liang, and Taiping Liu. MobiFusion-DRM: Composite multimodal fused multi-task prediction of diabetic retinopathy severity, macular edema and referable status on real-world handheld low-quality fundus images and clinical metadata. In *IISE Annual Conference. Proceedings*, pages 1–6. Institute of Industrial and Systems Engineers, 2026.

[108] Hancheng Ye, Zhengqi Gao, Mingyuan Ma, Qinsi Wang, Yuzhe Fu, Ming-Yu Chung, Yueqian Lin, Zhijian Liu, Jianyi Zhang, Danyang Zhuo, and Yiran Chen. KVCOMM: Online cross-context KV-cache communication for efficient LLM-based multi-agent systems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[109] Hua Ye, Siyuan Chen, Ziqi Zhong, Canran Xiao, Haoliang Zhang, Yuhan Wu, and Fei Shen. Seeing through the conflict: Transparent knowledge conflict handling in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 34423–34431, 2026.

[110] Asaf Yehudai, Lilach Eden, Alan Li, Guy Uzel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. A survey on evaluation of LLM-based agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 26690–26714, 2026.

[111] Miao Yu, Fanci Meng, Xinyun Zhou, Shilong Wang, Junyuan Mao, Linsey Pang, Tianlong Chen, Kun Wang, Xinfeng Li, Yongfeng Zhang, Bo An, and Qingsong Wen. A survey on trustworthy LLM agents: Threats and countermeasures. In *KDD*, 2025.

[112] Yichao Yuan, Mosharaf Chowdhury, and Nishil Talati. KAIROS: Stateful, context-aware power-efficient agentic inference serving. *arXiv preprint arXiv:2604.16682*, 2026.

[113] Sukwon Yun, Jie Peng, Pingzhi Li, Wendong Fan, Jie Chen, James Y. Zou, Guohao Li, and Tianlong Chen. Graph-of-agents: A graph-based framework for multi-agent LLM collaboration. In *ICLR*, 2026.

[114] Zapier. State of agentic AI adoption survey, 2025.

[115] Boxuan Zhang, Jianing Zhu, Zeru Shi, Dongfang Liu, and Ruixiang Tang. AgentForesight: Online auditing for early failure prediction in multi-agent systems. *arXiv preprint arXiv:2605.08715*, 2026.

[116] Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. AgenTracer: An agent failure tracer via reinforcement learning. In *ICLR*, 2026. arXiv:2509.03312.

[117] Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. In *International Conference on Learning Representations (ICLR)*, 2025.

[118] Lingzhe Zhang, Tong Jia, Mingyu Wang, et al. Efficient failure management for multi-agent systems with reasoning trace representation (EAGER). *arXiv preprint arXiv:2603.21522*, 2026.

[119] Lvmin Zhang and Maneesh Agrawala. View-oriented conversation compiler for LLM agent traces. *arXiv preprint arXiv:2603.29678*, 2026.

[120] Rui Zhang, Chaeeun Kim, and Liting Hu. A policy-driven runtime layer for agentic LLM serving. *arXiv preprint arXiv:2605.27744*, 2026.

[121] Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, et al. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In *ICML (Spotlight)*, 2025.

[122] Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-SafetyBench: Evaluating safety of LLM agents. *arXiv preprint arXiv:2412.14470*, 2024.

[123] Chenyu Zhao, Shenglin Zhang, et al. AgentTether: Graph-guided diagnosis and intervention for agents. *arXiv preprint arXiv:2607.06273*, 2026.

[124] Haoyu Zheng, Fangcheng Fu, Jia Wu, Binhang Yuan, Yongqiang Zhang, Hao Wang, Yuanyuan Zhu, Xiao Yan, and Jiawei Jiang. Efficient serving for dynamic agent workflows with prediction-based KV-cache management. *arXiv preprint arXiv:2605.06472*, 2026.

[125] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[126] Yinmin Zhong, Shengyu Liu, Junda Chen, et al. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. *arXiv preprint arXiv:2401.09670*, 2024.

[127] Heng Zhou, Jing Tang, Jusheng Zhang, Yanshu Li, Canran Xiao, Liwei Hou, Zong Ke, and Jiawei Yao. CoMem: Compositional concept-graph memory for vision–language adaptation. In *The Fourteenth International Conference on Learning Representations*, 2026.

[128] Chenyang Zhu, Spencer Hong, et al. RAFFLES: Reasoning-based attribution of faults for LLM agent systems. In *EACL*, 2026.

[129] Junru Zhu, Shiming Xie, Aime Lu Fan Chen, Xiaoqing Ding, Chunxin Tang, Ruoyu Qi, and Yulang Fei. Failure-transparent agents: Benchmarking post-failure reporting in tool-using language models. *arXiv preprint arXiv:2609.35732*, 2026.

[130] Junru Zhu, Yixin Yang, Xiaoqing Ding, and Ruoyu Qi. Leaseguard: Incumbent-preserving admission control for privileged llm agents. *arXiv preprint arXiv:2609.24077*, 2026.

[131] Junru Zhu, Yixin Yang, Xiaoqing Ding, and Ruoyu Qi. Partition-matched evaluation of community features under distribution shift in android malware function-call graphs. *arXiv preprint arXiv:2609.25256*, 2026.

[132] Junru Zhu, Yixin Yang, Xiaoqing Ding, and Ruoyu Qi. Residual community prototypes under-reject held-out malware families in fcg-mfd. *arXiv preprint arXiv:2609.24980*, 2026.

[133] Junru Zhu, Yixin Yang, Xiaoqing Ding, and Ruoyu Qi. Stress-testing structure-aware calibration of malware graph neural networks under type shift. *arXiv preprint arXiv:2609.28517*, 2026.

[134] Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. 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)*, pages 8580–8622, 2025.

[135] Pengyu Zhu, Lijun Li, Yaxing Lyu, Qianxin Luo, Jingyi Yang, Yi Liu, Tingfeng Hui, Xinyu Yuan, Li Sun, Sen Su, and Jing Shao. A unified framework for the evaluation of LLM agentic capabilities. *arXiv preprint arXiv:2605.27898*, 2026.

[136] Taiyu Zhu, Yifan Wu, Weilin Jin, Ying Li, and Gang Huang. StepFinder: A temporal semantic framework for failure attribution in multi-agent systems. *arXiv preprint arXiv:2606.03467*, 2026.

[137] Yiwen Zhu, Lihe Liu, Jiaqian Yu, and Di Zhang. LLM-based multi-agent orchestration: A survey of frameworks, communication protocols, and emerging patterns. *Future Internet*, 18(6):326, 2026.