Agentic Bridge Framework: Closing the Gap Between Agentic Capability and Performance Benchmarks

Yun Du
Stanford University
yundu27@stanford.edu

Rubens Lacouture
Stanford University
rubensl@stanford.edu

Qizheng Zhang
Stanford University
qizhengz@stanford.edu

Genghan Zhang
Stanford University
zgh23@stanford.edu

Tian Zhao
Classie AI
tian@classie.ai

Kunle Olukotun
Stanford University
kunle@stanford.edu

Abstract

While agentic AI systems perform impressively on emerging capability benchmarks, existing performance evaluation suites focus on non-agentic workloads, leaving a critical gap in understanding system efficiency for multi-step, tool-using agents. We present the Agentic Bridge Framework for extracting actionable performance insights from capability evaluations through trace-level telemetry. Applying this framework to a multi-agent system on GAIA validation, we reveal that: (1) pass@N strategies provide diminishing accuracy returns; (2) search agents dominate token usage and latency, identifying web data gathering as the primary bottleneck; (3) reasoning models spend more tokens on context preservation than actual reasoning, highlighting costly inter-agent communication overhead. These findings inform critical design choices—context engineering, tool-use optimization, and phase-aware resource allocation—and illustrate how agent traces can inform reproducible performance workloads, bridging capability achievements with systems optimization for efficient agentic AI.

1 Introduction

The landscape of AI systems is rapidly evolving from single-turn chat completions to complex agentic workflows that autonomously plan, invoke tools, and execute multi-step tasks [1–7]. This shift has spawned capability benchmarks—GAIA [8], WebArena [9], AgentBench [10], and PaperBench [11]—that evaluate agents on multi-turn tasks requiring reasoning, decision-making, and sophisticated tool use. Yet a critical disconnect exists: performance benchmarks like MLPerf [12, 13] and Artificial Analysis [14] remain anchored to non-agentic workloads (text-to-image generation, MMLU-Pro [15], GPQA Diamond [16], LiveCodeBench [17]), measuring cost and latency for tasks that bear little resemblance to the iterative, tool-heavy patterns of modern agents.

This gap has practical consequences. Without performance benchmarks that capture agentic behaviors, critical questions remain unanswered: What fraction of latency stems from tool calls versus reasoning? How do inter-agent handoffs impact token efficiency? When do best-of-N sampling strategies justify their computational cost? Current understanding relies on anecdotal evidence rather than systematic measurement, hindering both research progress and production deployment of agentic systems.

We introduce the Agentic Bridge Framework (Figure 1) to transform capability evaluations into actionable performance insights. Our framework provides a structured approach to instrument agent systems, collect trace-level telemetry, and extract optimization opportunities. Through a concrete implementation on GAIA, we show how this framework reveals bottlenecks—search operations

Machine Learning for Systems Workshop at NeurIPS 2025.

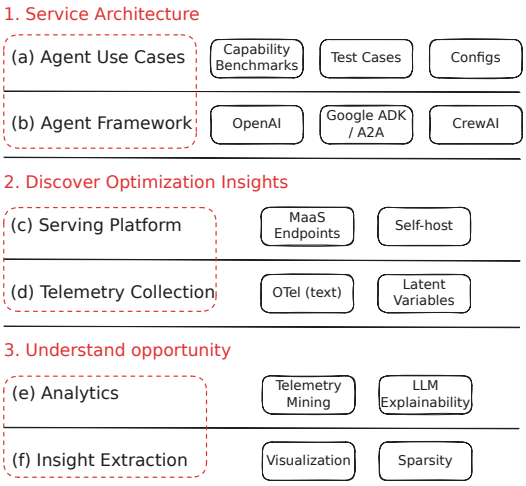


Figure 1: The Agentic Bridge Framework from Capability Tasks to Performance Insights.

dominating compute, context preservation exceeding reasoning costs—that inform both immediate optimizations and longer-term research directions.

2 The Agentic Bridge Framework

The Agentic Bridge Framework (Figure 1) provides a structured approach for extracting performance insights from capability-oriented agent evaluations. The framework consists of three layers that progressively transform high-level agent tasks into actionable system optimizations.

Service Architecture This layer captures the design decisions that define agent workloads and their implementation.

(a) Agent Use Cases: The choice of workload fundamentally shapes performance characteristics. Agents can execute established capability benchmarks (GAIA [8], PaperBench [11]) that stress multi-step reasoning and tool use, or simpler test patterns from frameworks like OpenAI Agents SDK [18]. Key configuration parameters significantly impact both accuracy and cost: (i) Pass@N strategies: Running workflows multiple times increases accuracy but multiplies computational cost; (ii) Model selection: Backend choice affects task completion patterns due to differences in model capabilities, tool-use pretraining, instruction following, and zero-shot prompt adaptation; (iii) Workload intensity: Single-user latency optimization differs fundamentally from multi-tenant throughput optimization.

(b) Agent Framework: Framework choice determines critical trade-offs between flexibility, development speed, and observability. OpenAI Agents SDK [18] exemplifies the integrated approach: built-in tracing (OpenTelemetry [19] spans for LLM/tool/handoffs [20]), native tool support (web search, MCP), and tight API integration enable rapid deployment but lock users to OpenAI models. In contrast, frameworks like CrewAI [21] provide granular control over agent modules and support diverse model backends, but require manual implementation of tracing, routing, and guardrails. A middle path exists through OpenAI-compatible API endpoints that enable OSS model integration, though developers must still implement custom tools or wrestle with inter-framework compatibility issues. This tension between ease-of-use and flexibility directly impacts performance measurement: frameworks with rich telemetry simplify optimization but constrain architectural choices, while flexible frameworks enable novel optimizations but complicate systematic evaluation.

Discover Optimization Insights. This layer reveals performance bottlenecks and optimization opportunities through systematic telemetry collection and analysis.

(c) Serving Platform: Serving infrastructure choices create fundamental trade-offs between control and convenience. Model-as-a-Service (MaaS) endpoints provide immediate deployment with per-token pricing but constrain model selection and telemetry access. Self-served deployments (e.g. GPU rental + vLLM) enable arbitrary open-source model selection and fine-grained monitoring, with hourly pricing that favors continuous workloads and potentially offers cost advantages at high

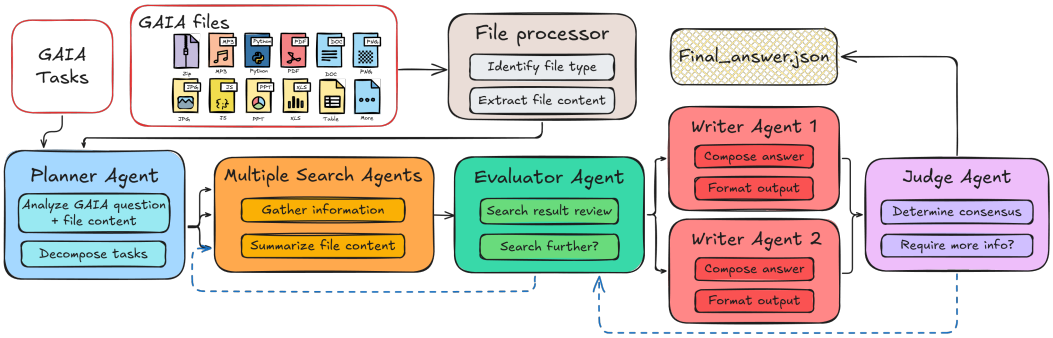


Figure 2: **Multi-agent system example for GAIA.** Solid arrows show the primary flow: task→planning→parallel search→evaluation→parallel writing→consensus. Dashed arrows indicate re-evaluation loops when consensus fails or more information is needed.

utilization. The serving choice cascades through the stack: MaaS limits telemetry to API-provided metrics, while self-hosting enables access to internal model states critical for optimization.

(d) Telemetry Collection: The depth of observable system behavior depends on serving architecture. API-based frameworks (OpenAI Agents SDK, LangChain [22]) provide OpenTelemetry (OTel) traces [19] capturing LLM inputs/outputs, tool calls, handoffs, exceptions, token usage, and span timing. Self-hosted deployments unlock richer telemetry: per-token logits/logprobs, layer activations, attention scores, KV cache states, MoE routing decisions, and RAG similarity scores. This granularity gap has practical implications—API traces suffice for identifying high-level bottlenecks, while diagnosing root causes requires low-level signals.

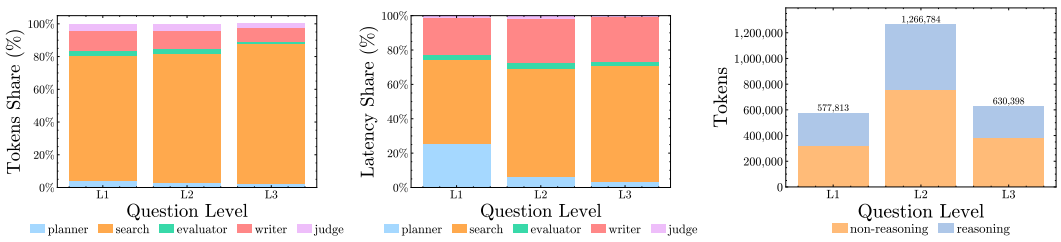
Understand Opportunity. This layer transforms telemetry into actionable optimizations.

(e) Analytics: Telemetry analysis operates at two levels. High-level OpenTelemetry traces identify macro-patterns: which agents dominate costs, tool-selection accuracy, planning effectiveness. These patterns can reveal which components dominate resource consumption and where context accumulates in multi-agent workflows. Low-level telemetry enables micro-optimizations: declining KV cache hit rates during tool calls indicate memory pressure; logit entropy spikes reveal model uncertainty requiring specialized prompting. Beyond debugging, telemetry enables workload characterization for system design. Sparse autoencoders [23] can classify execution phases (planning/tool-use/reasoning) from hidden states, informing phase-aware resource allocation strategies such as dynamic batch sizing, adaptive KV cache allocation, and precision tuning.

(f) Insight Extraction: Analytics crystallize into workload characterizations that bridge capability and performance evaluation. From telemetry patterns, we extract reproducible performance profiles: token distribution across agent roles, latency breakdowns by phase, accuracy-cost Pareto frontiers for pass@N strategies. These profiles enable systematic optimization—for instance, batching tool calls could reduce latency despite token overhead, while aggressive caching might cut redundant API calls. More importantly, these insights define performance benchmarks for agentic systems: QoS metrics (P90/95/99 latencies under multi-tenant load), scaling characteristics (throughput degradation with concurrent agents), and system-level trade-offs (time-to-first-token versus end-to-end latency). This transformation—from agent traces to performance workloads—provides the foundation for MLPerf-style evaluation of agentic systems, finally closing the gap between what agents can do and how efficiently they do it.

3 Case Study: Agent System For GAIA

We instantiate our framework on GAIA validation set as a proof-of-concept, chosen for its diverse task types (reasoning, tool-use, and file handling). We use a multi-agent system (Figure 2) implemented with OpenAI Agents SDK and Pydantic Logfire [24] telemetry. The system employs model specialization: o4-mini for orchestration (Planner, Evaluator, Judge), gpt-4.1 for information gathering (Search Agents), and o3 for synthesis (dual Writer Agents). The workflow processes GAIA's diverse file formats, decomposes questions into parallel searches, and enforces answer consensus through



(a) Tokens share by agent per level (b) Latency share by agent per level (c) Reasoning tokens usage

Figure 3: Token and latency distribution across GAIA validation set (165 tasks)

independent writers. We evaluate on the GAIA validation set containing 165 tasks (53 Level-1, 86 Level-2, 26 Level-3), running each task once for pass@1 and twice for pass@2 measurements.

4 Evaluation

Our GAIA validation results reveal critical insights about agentic system performance. The system achieves 52.12% accuracy (pass@1, 86/165 tasks) and 55.67% (pass@2, 92/165 tasks)—a modest 3.55% gain at doubled computational cost. Interestingly, pass@2 provides uniform absolute improvement across difficulty levels (+2 tasks each): L1: 66.04%→69.81% (+3.77%, 35→37 tasks), L2: 51.16%→53.49% (+2.33%, 44→46 tasks), L3: 26.92%→34.62% (+7.69%, 7→9 tasks). These results highlight that simply increasing N offers diminishing returns, underscoring the need for architectural and system-level optimizations beyond repeated sampling.

Figure 3 exposes two fundamental bottlenecks. First, search agents dominate resource consumption (60-80% of tokens and latency across all levels), identifying web data gathering as the primary optimization target—explaining the emergence of specialized tools like Tavily [25] and browser-use [26]. Second, reasoning models spend more tokens on context than reasoning itself: non-reasoning tokens exceed reasoning tokens by about 2x for L1-L2 and 1.5x for L3 (Figure 3c). This overhead stems from inter-agent handoffs where aggregated search results must be passed wholesale to downstream agents, presenting a stark trade-off: preserve full context at high token cost or risk information loss through summarization.

The economic implications are striking: our validation costs $67.06 via OpenAI APIs ($0.60/task), with total runtime of 2,931 minutes yielding $1.37/hour effective rate—comparable to Lambda's $1.49/hour on-demand GH200 (96GB) pricing [27]. However, this cost parity masks performance disparities: a self-hosted GH200 + Llama-3.1-70B could potentially reduce latency through dedicated compute and optimized batching, eliminating the 2-15 second compounding queuing delays we observed in API calls due to GPU multiplexing across users. These findings suggest a hybrid strategy: leverage APIs for o3-level reasoning (Writers) while self-hosting search agents where speed matters more than sophistication—especially for continuous evaluation, which benefits from per-hour pricing.

5 Limitations and Future Work

Limitations. Our results face three constraints: (1) temporal instability—web content and API latencies drift across runs, limiting reproducibility; (2) observability gaps—MaaS endpoints provide only OpenTelemetry traces, hiding low-level signals (KV-cache states, attention patterns) critical for micro-optimizations; (3) limited ablations—infinite multi-agent system design space and compute quotas restrict exploration of agent topologies and pass@N scaling beyond N=2.

Future work. The framework points to concrete optimizations: *Serving*—implement phase-aware resource allocation (larger KV cache during reasoning, reduced precision during handoffs) and heterogeneous model routing. *Telemetry*—standardize minimal agentic schemas combining OTel spans with critical latents (logit entropy, cache hit rates). *Analytics*—build phase detectors to enable real-time budgeting. *Insights*—develop context-engineering policies (intelligent summarization before handoffs) and search result caching. These optimizations, suggested by our telemetry patterns, warrant investigation for their potential to reduce token usage while maintaining accuracy.

References

[1] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv:2210.03629*, 2022. URL https://arxiv.org/abs/2210.03629.

[2] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv:2302.04761*, 2023. URL https://arxiv.org/abs/2302.04761.

[3] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *arXiv:2303.17580*, 2023. URL https://arxiv.org/abs/2303.17580.

[4] OpenAI. Introducing chatgpt agent: bridging research and action. https://openai.com/index/introducing-chatgpt-agent/, 2025. Accessed Aug 14, 2025.

[5] OpenAI. Introducing operator. https://openai.com/index/introducing-operator/, 2025. Accessed Aug 14, 2025.

[6] OpenAI. Introducing deep research. https://openai.com/index/introducing-deep-research/, 2025. Accessed Aug 14, 2025.

[7] OpenAI. Introducing codex. https://openai.com/index/introducing-codex/, 2025. Accessed Aug 14, 2025.

[8] Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: A benchmark for general ai assistants. *arXiv preprint arXiv:2311.12983*, 2023. URL https://arxiv.org/abs/2311.12983.

[9] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. URL https://arxiv.org/abs/2307.13854.

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

[11] Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. Paperbench: Evaluating ai's ability to replicate ai research, 2025. URL https://arxiv.org/abs/2504.01848.

[12] Peter Mattson, Christine Cheng, Cody Coleman, Greg Diamos, Paulius Micikevicius, David Patterson, Hanlin Tang, Gu-Yeon Wei, Peter Bailis, et al. Mlperf training benchmark. In *Proceedings of Machine Learning and Systems (MLSys)*, 2020. URL https://proceedings.mlsys.org/paper_files/paper/2020/hash/411e39b117e885341f25efb8912945f7-Abstract.html.

[13] Vijay Janapa Reddi, Christine Cheng, David Kanter, et al. Mlperf inference benchmark. In *Proceedings of the 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020. doi: 10.1109/ISCA45697.2020.00045. URL https://dl.acm.org/doi/10.1109/ISCA45697.2020.00045.

[14] Artificial Analysis. Artificial analysis: Ai model & api providers analysis. https://artificialanalysis.ai/, 2025. Accessed Aug 14, 2025.

[15] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

[16] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

[17] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

[18] OpenAI. Openai agents sdk (python) — documentation. https://openai.github.io/openai-agents-python/, 2025. Accessed Aug 2025.

[19] OpenTelemetry. Opentelemetry concepts: Traces. https://opentelemetry.io/docs/concepts/signals/traces/, 2025. Accessed Aug 12, 2025.

[20] OpenAI. Openai agents sdk: Tracing. https://openai.github.io/openai-agents-python/tracing/, 2025. Accessed Aug 12, 2025.

[21] crewAI Inc. crewai. https://github.com/crewAIInc/crewAI, 2025. GitHub repository; accessed Aug 15, 2025.

[22] Harrison Chase. Langchain, 2022. URL https://github.com/langchain-ai/langchain.

[23] Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders, 2024. URL https://arxiv.org/abs/2406.04093.

[24] Pydantic Team. Pydantic logfire. https://pydantic.dev/logfire, 2024. Accessed: 2025-08-16.

[25] Tavily. Tavily search api. https://docs.tavily.com, 2025. Accessed: 2025-08-17.

[26] browser-use contributors. browser-use: Autonomous web browsing for llm agents. https://github.com/browser-use/browser-use, 2025. Accessed: 2025-08-17.

[27] Lambda. Ai cloud pricing, 2025. URL https://lambda.ai/pricing. Accessed 2025-08-18.