POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Beyond AI in the Wild: Optimizing Root Cause Analysis with Multi-Agent Systems

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Marcello Martini

Advisor: Prof. Marco Domenico Santambrogio

Co-advisors: Ing. Matteo Ferroni, Ing. Beatrice Branchini

Academic year: 2024-2025

---

1. Introduction

The transition from monolithic architectures to **distributed microservices** has drastically changed the operational landscape for **Site Reliability Engineering (SRE)** teams. While offering scalability, microservices obscure system visibility; a single user request often involves dozens of services, creating a complex dependency network that is difficult to map manually. Consequently, incident response has evolved into a bottleneck, where manual correlation of metrics, logs, and traces significantly increases the **Mean Time To Resolution (MTTR)**.

To address these limitations, the industry is exploring **Artificial Intelligence for IT Operations (AIOps)**. Specifically, Large Language Models (LLMs) are emerging as promising reasoning engines for **Root Cause Analysis (RCA)**. However, deploying probabilistic agents in deterministic infrastructure environments introduces the so-called **"AI in the Wild"** problem. This includes critical challenges such as prohibitive token usage, context window saturation due to high-volume telemetry, and the risk of hallucinations when models are fed unfiltered data.

To address this gap, the main contributions of this study lie in the introduction of a novel **Multi-Agent System (MAS)** for RCA in Kubernetes environments. The architecture employs a **divide and conquer strategy**, decomposing complex diagnostics into manageable sub-tasks executed by specialized agents. Key contributions include a **hybrid deterministic-generative** Triage agent, a **datagraph** to ground reasoning in the cluster topology, and a custom **Model Context Protocol (MCP) server** that distills telemetry data to optimize token efficiency and mitigate hallucinations. In our evaluation, the proposed system achieves a **20.46% gain** over the state of the art while using a **more compact model**, and a comparative analysis highlights how configuration choices impact **token usage** and **execution time**. The implementation is publicly available on GitHub [1].

2. Proposed Multi-Agent Architecture

The proposed system orchestrates **four specialized agents** to automate the diagnosis workflow. The process runs in a **feedback loop** until the evidence is sufficient.

① The workflow begins with the **Triage Agent**,

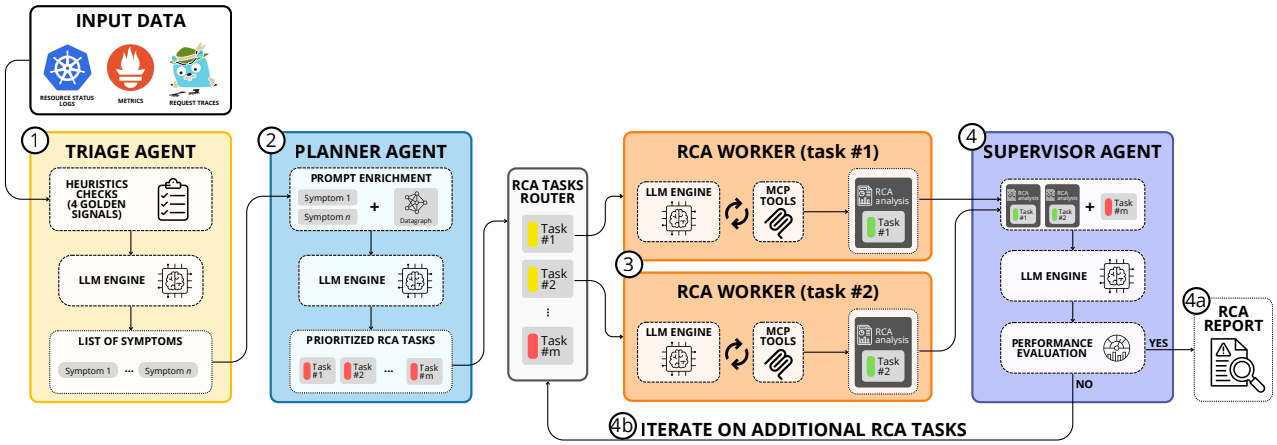

Figure 1: **Workflow of the proposed MAS.** The workflow moves from Triage to Planning, followed by parallel execution by RCA Workers, and final aggregation by the Supervisor.

which employs a hybrid deterministic-generative approach. It first executes deterministic heuristic checks on the **"Four Golden Signals"** [2] (Latency, Traffic, Errors, Saturation) to identify symptoms using cluster metrics, Pod statuses, and traces with error flags (as fallback). Then, the LLM aggregates the evidence and produces a list of symptoms, reducing the risk of hallucinating faults where none exist.

② Detected symptoms are passed to the **Planner Agent**. Leveraging the **cluster topology** (i.e., mapping upstream services and infrastructure dependencies for each resource), the Planner decomposes the problem into **discrete, prioritized investigation tasks**. It filters out irrelevant services, ensuring that downstream analysis focuses only on the dependency chain of the affected component.

③ To reduce resolution time, a **divide and conquer** strategy is applied: tasks are dispatched to multiple **RCA Workers** that operate in **parallel**. Each Worker utilizes the **ReAct** [3] (Reason and Act) pattern to query MCP tools and gather evidence.

④ Finally, a **Supervisor Agent** aggregates the findings and either produces a ④a **RCA report** or triggers ④b **additional investigation rounds** when evidence is inconclusive.

3. Implementation Strategy

Building on the **"AI in the Wild"** challenges outlined in Section 1, we implemented targeted engineering choices in tooling and knowledge representation to support the MAS in performing RCA efficiently and reliably. We outline our MCP-based tool layer, datagraph grounding, and model selection.

3.1. Communication Layer & Optimized Tooling

A **custom MCP [4] server** was developed to facilitate interaction between the agents and the cluster. To avoid **context saturation** caused by raw, verbose telemetry data, the MCP tools performed "**data distillation**": rather than returning full telemetry dumps, each tool returns a filtered, task-oriented view and allows the agent to progressively request more detail.

Traces: the Jaeger [5] tool can return either a **high-level overview** (a compact list of candidate traces with key signals such as **error status** or high latency) or, on demand, the **full raw trace** for a specific trace ID.

Metrics: metrics are collected from Prometheus [6]. Instead of asking the agent to guess metric names, the tool takes a resource name as input and returns a **curated set of relevant metrics** for that resource, reducing both query errors and hallucinated metric names.

Logs: logs are filtered server-side by severity (e.g., `FATAL`, `WARN`) to **save context space**. Furthermore, **strict validation of the tool invocation schema** and resource naming was implemented to prevent tool execution errors and redundant ReAct iterations.

3.2. Topological Grounding

The system maintained a **datagraph**, a **graph-based representation** of the Kubernetes cluster derived from distributed traces (Jaeger) and infrastructure inspection. This graph allowed agents to understand **system relationships** by mapping **data dependencies** (upstream services required for a task) and **infrastructure dependencies** (critical backing services like databases). This topological awareness prevented the agents from performing unnecessary investigations of irrelevant resources. The datagraph can also be queried via MCP-exposed tools, allowing the RCA worker to determine which Pods are associated with a service (and vice versa), as well as the upstream and downstream dependencies of a given resource.

3.3. Model Selection

Conversely from literature approaches relying on flagship models, this system utilizes **GPT-5-mini** [7]. This choice rests on the principle that **task decomposition** and **architectural guardrails** can bridge the performance gap of **smaller models**. By partitioning the workflow, we aim to show that a **cost-effective model** can maintain **high reliability** within a well-structured environment.

4. Experimental Setup

To evaluate the performance of our MAS and identify the best configuration in terms of the accuracy–efficiency trade-off, we conducted the following experimental campaign.

The experimental validation leveraged **AIOpsLab** [8], a framework designed to evaluate AI agents in autonomous cloud environments. To ensure a standard and repeatable testbed, we utilized the framework exclusively for its environment management capabilities: deploying microservice architectures and injecting controlled faults. We replaced the framework's native interface with our custom MCP server to optimize data retrieval and implemented a tailored evaluation logic.

To ensure reproducibility and manage the combinatorial complexity of testing multiple agents across various fault scenarios, experiments are executed through an **automated pipeline**. This component orchestrates deployment, fault injection, traffic generation, and agent execution. After each run, we collect **execution traces** and run-level metadata via LangSmith [9] (e.g., **tool-call traces**, **token usage**, and **execution time**), which are aggregated to compute the performance metrics.

Prompt (S)	$P = 2$	$P = 3$	$P = 5$
Plain ReAct (★)	**A**	**B**	**C**
Conservative (♣)	**D**	**E**	–
Tool-free (■)	**F**	**G**	–

Table 1: **Agent configurations.** All configurations utilize a Budget $B = 7$

We evaluated **12 agent configurations**, formally identified by the Agent ID and the tuple $\langle P, B, S \rangle$. Here, **parallelism (P)** represents the number of concurrent RCA tasks (top-P prioritized tasks); **budget (B)** defines the maximum tool invocations allowed per RCA Worker; and **system prompt (S)** dictates the behavioral strategy.

Table 1 presents the **most relevant** configurations, covering three prompt strategies evaluated: **tool-guided** variants, where the Planner explicitly prescribes the tools the RCA Worker should use (*Plain ReAct*, ★; configurations **A**, **B**, **C**); **tool-free** variants, where the agent autonomously selects which tools to invoke (■; configurations **F**, **G**); and **conservative** variants, which enforce strict brevity in intermediate reasoning and outputs to mitigate hallucinations (♣; configurations **D**, **E**).

4.1. Evaluation criteria

The experimental campaign covered **two applications** ("Hotel Reservation" and "Social Network") across **17 fault types**. Performance is evaluated across three dimensions: **detection accuracy**, **localization accuracy**, and a semantic **RCA score**. While detection and localization follow standard AIOpsLab definitions, the **RCA score** is computed via an **LLM-as-a-Judge** procedure and captures the **quality of the final RCA report**, rather than only whether the run is correct/incorrect.

Unless otherwise stated, results reflect a **single run** per scenario. For the specific comparative analysis between **Agent A** and **Agent F**, we instead relied on repeated runs to assess statistical

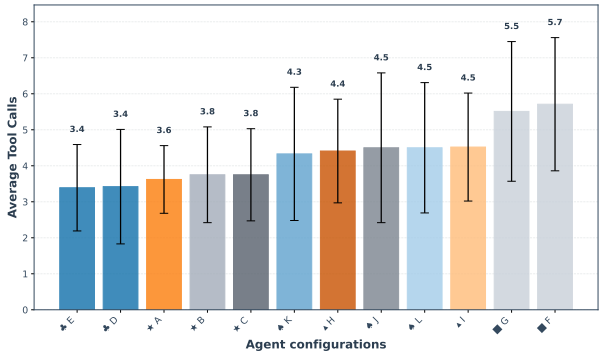

Figure 2: **Average number of tool calls performed by the RCA Worker per iteration.** Symbols denote the Prompt Strategy employed. The data shows that configurations guided by the Planner ($\star, \clubsuit$) require significantly fewer tool invocations (≈3.4-3.8) compared to autonomous Tool-free agents (■, ≈5.5-5.7).

significance (see Section 5.4).

5. Results and Discussion

The experimental results shows the efficacy of the proposed architecture in terms of both accuracy and efficiency. We first characterize agent reasoning and tool-usage patterns, then examine how data quality affects diagnosis, analyze the impact of different agent configurations on cost–accuracy trade-offs, and finally compare our system against state-of-the-art baselines.

5.1. Agent Reasoning Patterns and Tool Usage

Qualitative analysis of agent behavior revealed a **"broad-to-narrow" investigation strategy**. Agents consistently began by requesting **high-level system overviews** (e.g., trace summaries) before investigating **specific logs** (e.g., full raw traces). This workflow helps manage the **context window** by progressively narrowing which resources require deeper inspection, thereby reducing unnecessary **token usage** and mitigating **hallucinations**.
Regarding efficiency, task decomposition effectively limited the context required for each agent. Even with high autonomy, RCA Workers averaged only **3.4 to 5.7 tool calls per task** (Figure 2), indicating that well-scoped sub-tasks reduce loops and excessive token consumption.

5.2. Impact of Data Quality on MAS performance

The performance of the system is closely tied to the **telemetry signal-to-noise ratio** (Figure 3). Faults clearly visible in **high-signal logs** (e.g., *Port mismatch*) were significantly easier to diagnose (Loc.=1.00) than those obscured by **raw distributed traces** (e.g., *Network delay*, Loc.=0.58).

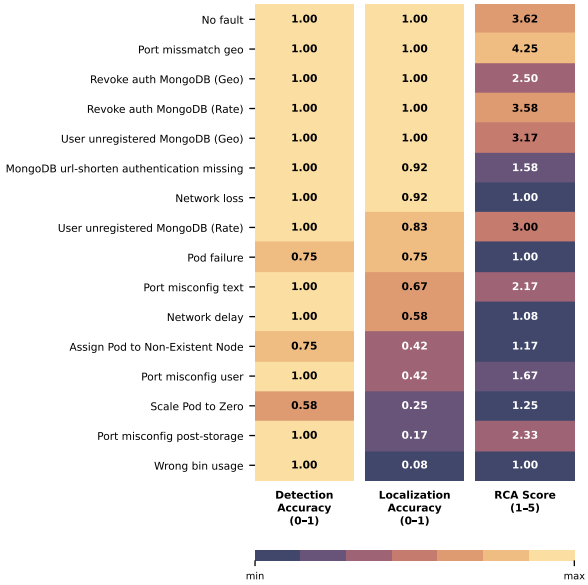

Figure 3: **Per-fault performance summary (Detection, Localization, and RCA Score).** Each row corresponds to a fault type; columns report Detection Accuracy and Localization Accuracy (both in $[0, 1]$) and semantic RCA Score (in $[1, 5]$). The heatmap highlights that log-driven faults (e.g., configuration mismatches) achieve consistently high Detection/Localization, while trace-heavy network faults exhibit lower Localization and RCA Score.

This behavior reflects a known limitation of current LLMs when processing raw tracing data. Raw traces are often verbose and consume a large amount of tokens while containing few relevant details. This floods the model's context window with noise, triggering a **"garbage in, garbage out" dynamic** where the LLM struggles to isolate the root cause compared to the clear evidence found in logs.
In contrast, the **deterministic** component of the **Triage Agent** is effective at suppressing **false positives**, favoring precision over speculative diagnoses, and achieves **100% detection**

accuracy in **"no fault"** scenarios (Figure 3). However, this design can become a limitation when telemetry is **incomplete** (for instance, if the **workload generator** fails to trigger sufficient traffic), as the system may correctly (but uninformatively) conclude **"no fault"** due to lack of evidence.

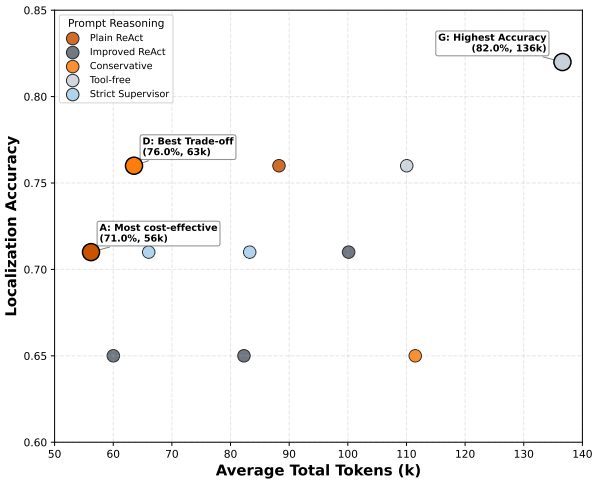

Figure 4: **Efficiency vs. Accuracy Trade-off across Agent Configurations.** The x-axis represents average token consumption (lower is better), while the y-axis shows Localization Accuracy (higher is better). The plot highlights the system's design choices: Agent A offers the most cost-effective solution, Agent G achieves the performance ceiling at a higher cost, and Agent D provides the optimal balance.

5.3. Agent Configuration Comparison

To compare agent configurations, we analyze the **accuracy–efficiency trade-off** by jointly considering **localization accuracy** and **token consumption** (Figure 4). Among all tested configurations, three emerged as noteworthy:

- **Agent G (Best Performance):** adopting a tool-free strategy with parallelism $P = 3$ and budget $B = 7$, this configuration achieved the **highest accuracy** (**100%** detection, **82%** localization). However, this performance comes at the expense of the **highest computational cost**, making it suitable for critical scenarios where resource consumption is secondary to precision.
- **Agent D (Best Trade-off):** utilizing a "conservative" strategy that enforces concise reasoning in the RCA Worker ($P = 3, B = 7$), it provided the **best balance** between accuracy and efficiency, achieving **76%** localization accuracy while consuming $\approx 53\%$ **fewer tokens** than Agent G. The slight drop in detection accuracy (**88%**) is primarily attributed to experimental artifacts (insufficient workload traffic; see Section 5.2).
- **Agent A (Most Cost-Effective):** using a standard tool-guided ReAct strategy (Plain ReAct) with reduced parallelism $P = 2$ and budget $B = 7$, it achieved the **lowest resource consumption** ($\approx 56k$ **tokens**, **188s**) while maintaining **100%** detection and **71%** localization. This result suggests that high parallelism is not strictly necessary for effective diagnosis, offering a viable lightweight alternative for resource-constrained environments.

5.4. Efficiency of Tool-Guided Planning

To quantify the efficiency benefits of tool-guided planning, we compared **Agent A** (tool-guided) against **Agent F** (tool-free) using repeated runs and performed a **one-sided** hypothesis test on **token usage** and **execution time** ($H_0 : \mu_A \geq \mu_F$ vs. $H_1 : \mu_A < \mu_F$, $\alpha = 0.05$). We verified normality via **Shapiro–Wilk** and applied either a **Student's** t**-test** or a **Mann–Whitney U** test accordingly.

Overall, we rejected H_0 in **9/10** comparisons, showing that tool guidance (Agent A) yields consistent efficiency gains (up to **42%** fewer tokens and **29%** lower execution time). The only non-significant result was execution time for *Social Network – Auth Missing* ($p = 0.103$).

5.5. Comparison with State of the Art

Finally, we benchmarked our approach against the state-of-the-art baselines provided by AIOpsLab. Due to hardware constraints, we evaluated **2/3** of the available testbed applications and a **subset** of the benchmark fault types (**17** in total).

Moreover, **token usage** and **execution time** are not directly comparable to AIOpsLab due to differences in the execution model (our **MAS** performs the full analysis in a single run, whereas AIOpsLab uses a **single agent** and executes separate tasks) and in the interaction layer

Task	Approach	Agent/Model	Accuracy	Gain
Detection	SoTA (AIOpsLab)	FLASH	100.00%	-
	Ours (Best)	**Agent G (GPT-5-mini)**	**100.00%**	=
	Ours (Trade-off)	Agent D (GPT-5-mini)	88.00%	-12.00%
Localization	SoTA (AIOpsLab)	GPT-4-W-SHELL (GPT-4)	61.54%	-
	Ours (Best)	**Agent G (GPT-5-mini)**	**82.00%**	**+20.46%**
	Ours (Trade-off)	Agent D (GPT-5-mini)	76.00%	+14.46%

Table 2: **Comparison with State-of-the-Art (AIOpsLab).** Accuracy comparison between the best AIOpsLab single-agent baselines (using GPT-4) and our Multi-Agent System configurations (using GPT-5-mini).

(our **MCP** interface is more verbose than their custom ACI). Therefore, we restrict the comparison to **detection** and **localization** accuracy.
As reported in Table 2, our **best** configuration (Agent G, GPT-5-mini) matches SoTA **detection** (100%) and substantially improves **localization** (82% vs. 61.54%, i.e., **+20.46%**). The **trade-off** configuration (Agent D) achieves **76% localization (+14.46%)** with lower detection accuracy (88%), consistent with the workload-related artifacts discussed in Section 5.2.

6. Conclusions

This thesis validates the efficacy of a **MAS** for RCA, showing that architectural guardrails such as **heuristic triage**, **explicit planning**, and **topological grounding** are stronger determinants of success than raw model scale, achieving a **localization accuracy of 82%**. Moreover, we show that the agent configuration can be tuned to trade accuracy for efficiency: the **best trade-off** setting (Agent D) reduces **token usage** and **execution time** while maintaining competitive localization performance.
While handling high-volume **distributed traces** remains a challenge due to context saturation, the distinct issue of interpreting **numerical metrics** will be addressed in future iterations by integrating specialized **Deep Learning models** exclusively for time-series anomaly detection. Furthermore, the research paves the way for a transition from **proprietary APIs** to on-premise **Small Language Models (SLMs)**, ensuring data sovereignty and enabling fully local, privacy-compliant deployment.

References

[1] Marcello Martini. *Github repository SRE-Agent*. URL: https://github.com/martinimarcello00/SRE-agent.

[2] Chris Jones, Betsy Beyer, Niall Richard Murphy, and Jennifer Petoff. *Site Reliability Engineering: How Google Runs Production Systems*. en. 2016. ISBN: 1-4919-2912-X. URL: https://sre.google/sre-book/.

[3] Shunyu Yao et al. *ReAct: Synergizing Reasoning and Acting in Language Models*. arXiv:2210.03629 [cs]. Mar. 2023. DOI: 10.48550/arXiv.2210.03629. URL: http://arxiv.org/abs/2210.03629.

[4] *Model Context Protocol*. en. URL: https://github.com/modelcontextprotocol.

[5] *Jaeger: open source, distributed tracing platform*. URL: https://www.jaegertracing.io.

[6] *Prometheus*. URL: https://prometheus.io.

[7] Aaditya Singh et al. *OpenAI GPT-5 System Card*. arXiv:2601.03267 [cs]. Dec. 2025. DOI: 10.48550/arXiv.2601.03267. URL: http://arxiv.org/abs/2601.03267.

[8] Yinfang Chen et al. *AIOpsLab: A Holistic Framework to Evaluate AI Agents for Enabling Autonomous Clouds*. arXiv:2501.06706 [cs]. Jan. 2025. DOI: 10.48550/arXiv.2501.06706. URL: http://arxiv.org/abs/2501.06706.

[9] *LangSmith*. URL: https://www.langchain.com/langsmith.