

AgenticMail Call Agent vs. OpenClaw Sessions Spawn: A Comparative Analysis of Inter-Agent Communication Paradigms

Ope Olatunji¹ Fola (AI Research Assistant)¹

¹AgenticMail / OpenClaw Research
ope.olatunji@outlook.com

March 27, 2026 — v4 (Side-by-Side Comparison)

Abstract

As multi-agent AI systems evolve from single-model architectures to orchestrated ensembles, the communication layer between agents becomes a critical design decision. This paper presents a **side-by-side empirical comparison** of two production inter-agent communication paradigms: (1) **AgenticMail Call Agent** (`call_agent`), a structured synchronous/async delegation pattern with three execution modes (`light`, `standard`, `full`), and (2) **OpenClaw Sessions Spawn** (`sessions_spawn`), a free-text sub-agent spawning pattern with full session isolation. Through **9 Call Agent tests** and **16 Sessions Spawn tests** across 10 industries, we evaluate both paradigms on latency, output quality, domain versatility, output structure, and collaboration capability. Our key finding overturns an earlier assumption: **Call Agent in standard/full mode produces deep, structured analysis comparable to Sessions Spawn**—not just shallow data retrieval. We additionally demonstrate **inter-agent email coordination** in Call Agent’s full mode, where a finance agent autonomously emailed a writer agent. We conclude with a decision framework: use Call Agent for structured, composable outputs; use Sessions Spawn for autonomous specialist work requiring tool use and free-form reasoning. Full test results for both paradigms are provided in companion directories: [results-callagent/](#) and [results/](#).

Keywords: multi-agent systems, AgenticMail, OpenClaw, inter-agent communication, orchestration, AI agents, structured delegation, `call_agent`, `sessions_spawn`

1 Introduction

The rapid advancement of large language model (LLM) capabilities has given rise to agentic AI systems—autonomous agents equipped with tools, memory, and the ability to take actions in the world [5, 2]. Two production platforms for multi-agent orchestration are **AgenticMail**, a structured agent communication framework with typed task delegation, and **OpenClaw**, an open-source personal AI assistant platform supporting isolated sub-agent sessions.

Each platform offers a distinct paradigm for inter-agent communication. This paper tests both paradigms head-to-head across overlapping task domains to answer: *when should you use AgenticMail Call Agent, and when should you use OpenClaw Sessions Spawn?*

Prior versions of this paper (v1–v3) tested Call Agent only in `light` mode, leading to a biased conclusion that it could only handle shallow data retrieval (~1 s, structured JSON but no deep reasoning). **This version (v4)** tests Call Agent across all three modes—`light`, `standard`, and `full`—revealing that it produces **deep, structured analysis** on complex tasks, fundamentally changing the comparison.

We identify five critical axes of comparison:

1. **Latency**—How quickly does the orchestrator receive results?
2. **Output structure**—Can results be programmatically consumed?
3. **Reasoning depth**—How well does each handle complex analysis?
4. **Domain versatility**—Performance across diverse industries?

5. **Multi-agent collaboration**—Can agents coordinate on larger problems?

2 The Two Paradigms

2.1 AgenticMail Call Agent

The AgenticMail `call_agent` function implements a **structured delegation** pattern. The orchestrating agent submits a task to a named subordinate agent and receives a structured JSON result.

Three execution modes control the subordinate’s environment:

- `mode="light"` — No email, no memory, no workspace. Lowest latency. Ideal for data retrieval and lookups.
- `mode="standard"` — Web search, tools, trimmed context. Balanced for analysis tasks. Supports sync and async execution.
- `mode="full"` — All coordination features including inter-agent email. For multi-agent collaboration workflows.

```
// Call Agent standard mode (sync)
result = call_agent(
  target: "finance-agent",
  task: "Analyze AAPL stock",
  mode: "standard"
)
// Returns structured JSON:
// { status: "completed",
//   result: { price: "$248.80",
//     pe_ratio: 31.41,
//     recommendation: "Hold" ... } }
```

Key characteristic: Returns structured JSON regardless of task complexity. The orchestrator receives machine-parseable data and decides presentation.

2.2 OpenClaw Sessions Spawn

OpenClaw `sessions_spawn` creates an **isolated sub-agent session** with a natural language prompt. The sub-agent has its own context, tools, and environment. Results arrive via push notification.

```
// Sessions Spawn
sessions_spawn(
  task: "Analyze AAPL stock...",
  mode: "run", label: "finance"
)
sessions_yield() // yield turn
// Push: sub-agent completed
// Returns free-form markdown:
```

```
// "## AAPL Investment Brief
// Current Price: $249.23 ...
// Bull Case: AI monetization ..."
```

Key characteristic: Returns free-form natural language with expert reasoning, tool use traces, and autonomous decision-making.

3 Methodology

3.1 Experimental Setup

All tests ran on consumer hardware (Apple Mac mini, ARM64, macOS 25.2.0, Node.js v25.5.0) with Claude Opus (Anthropic) as the underlying model for both paradigms.

AgenticMail Call Agent: 9 tests across light (1), standard (7, including 2 async), and full (1) modes.

OpenClaw Sessions Spawn: 16 tests executed in three parallel waves (5–6 concurrent sub-agents per wave).

3.2 Overlapping Test Domains

To enable direct comparison, we tested both paradigms on **7 overlapping domains**. Two additional domains were tested only with Call Agent (weather, which is trivial for Spawn) and 9 additional domains were tested only with Sessions Spawn (legal, creative, research, clinical trials, M&A legal/ops).

4 Results: Side-by-Side Comparison

4.1 Weather / Data Retrieval

Table 1: Weather: Charlotte, NC.

Metric	Call Agent	Spawn
Mode	light	(not tested)
Runtime	18 s	—
Output	Structured JSON	—
Ref	CA-01	—

Verdict: Call Agent is the obvious choice for data retrieval. Spawning a full sub-agent for weather data would be wasteful.

4.2 Software Engineering: Code Review

Table 2: Python security code review.

Metric	Call Agent	Spawn
Mode	standard	run
Runtime	~100 s	24 s
Issues found	9	9
Corrected code	Full JSON	Inline mark-down
Machine-parse	Yes	No
Ref	CA-02	SS-01

Verdict: Spawn was 4× faster here. Both found 9 issues with similar severity breakdown. Call Agent’s advantage is structured output (CWE codes, line numbers, severity as fields) that can feed into CI/CD pipelines. Spawn’s output reads better as a human-facing report.

4.3 Finance: AAPL Stock Analysis

Table 3: AAPL investment analysis.

Metric	Call Agent	Spawn
Mode	standard	run
Runtime	~130 s	36 s
Price data	\$248.80	\$249.23
PE ratio	31.41	31.2
Bull/bear cases	Yes	Yes
Analyst consensus	42 analysts	General
Technical levels	Support/resist.	Not included
Machine-parse	Yes	No
Ref	CA-03	SS-02

Verdict: Call Agent was slower (web search overhead) but produced more data points including analyst counts and technical levels. Spawn was faster and wrote a more readable narrative. For a dashboard: Call Agent. For a client brief: Spawn.

4.4 Healthcare: Drug Interactions

Table 4: Drug interaction analysis (Metformin / Lisinopril / Atorvastatin).

Metric	Call Agent	Spawn
Mode	standard	run
Runtime	~96 s	72 s
Pairs analyzed	3	3
Severity ratings	Per-pair	Narrative
Mechanisms	Detailed	Detailed
Monitoring plans	Yes	Yes
FDA alerts	4 with dates	General
Machine-parse	Yes	No
Ref	CA-04	SS-06

Verdict: Both produced comprehensive drug interaction reports. Call Agent structured each pair with explicit severity fields and included dated FDA alerts—better for clinical decision support systems. Spawn wrote a more readable clinical narrative. Quality is comparable; the difference is format.

4.5 Marketing: Gen Z Campaign

Table 5: Gen Z budgeting app launch campaign.

Metric	Call Agent	Spawn
Mode	standard	run
Runtime	~48 s	88 s
Persona	Yes	Yes
Msg. pillars	4	4
Channels	6	6
CAC projections	Per-channel	General
Budget alloc.	Detailed	Detailed
Week-1 content	Per-channel	Summary
Machine-parse	Yes	No
Ref	CA-05	SS-16

Verdict: Call Agent was faster *and* more detailed here—per-channel CAC projections, specific week-1 content for each channel. This is a case where structured JSON contains more actionable data than the free-form output. Call Agent wins on both speed and substance.

4.6 Supply Chain: Shenzhen to Charlotte

Table 6: Shipping route optimization.

Metric	Call Agent	Spawn
Mode	standard	run
Runtime	~141 s	62 s
Routes compared	3	3
Cost per TEU	Ranges	Estimates
Carrier names	Specific	General
Alt. conditions	Decision tree	Brief
Machine-parse	Yes	No
Ref	CA-06	SS-08

Verdict: Call Agent was slower (141 s vs 62 s) but produced more granular data—specific carrier options, cost ranges per TEU, and explicit decision criteria for route selection. Spawn noted that web search was unavailable and relied on domain knowledge. When web access matters, Call Agent’s standard mode (which includes search) has an advantage.

4.7 Data Science: Forecasting Models

Table 7: Sales forecasting model comparison.

Metric	Call Agent	Spawn
Mode	standard	run
Runtime	~53 s	62 s
Models compared	4	4
MAPE ranges	Yes	Yes
Impl. plan	8 steps	6 steps
Code examples	Described	Inline
Machine-parse	Yes	No
Ref	CA-07	SS-04

Verdict: Nearly identical quality and speed. Call Agent’s 8-step implementation plan is more granular; Spawn included inline code snippets. Both recommended XGBoost. A tie on substance; Call Agent wins on structure, Spawn wins on readability.

4.8 Education: ML Curriculum

Table 8: 4-week ML curriculum design.

Metric	Call Agent	Spawn
Mode	standard	run
Runtime	~93 s	70 s
Total hours	60	60
Prerequisites	Detailed	Detailed
Weekly projects	4	4
Resources	Per-week	Per-week
Assessment	Rubric + %	General
Post-curriculum	4 paths	3 paths
Machine-parse	Yes	No
Ref	CA-08	SS-05

Verdict: Both produced excellent curricula. Call Agent included explicit grading rubrics and an extra career path. Spawn’s narrative format is more immediately usable as course documentation. For LMS integration: Call Agent. For a syllabus document: Spawn.

4.9 M&A Finance: Due Diligence

Table 9: M&A financial due diligence (TechCorp \$400M acquisition).

Metric	Call Agent	Spawn
Mode	full	run
Runtime	~58 s	78 s
Red flags	8	8
Valuation	Detailed	Detailed
Revenue quality	ARR/NRR/churn	ARR/NRR/churn
Recommendation	Proceed w/ caut.	Upper value fair
Email writer	Yes	N/A
Machine-parse	Yes	No
Ref	CA-09	SS-10

Verdict: This is the most significant comparison. Both produced 8 red flags, detailed valuation analysis, and cautious recommendations. Call Agent was faster (58 s vs 78 s) and **autonomously emailed the writer agent** to begin formatting the deliverable—demonstrating inter-agent coordination within the AgenticMail ecosystem. Spawn produced a richer narrative but had no inter-agent communication capability.

5 Inter-Agent Email Coordination

The M&A finance test (Call Agent, mode="full") demonstrated a capability unique to AgenticMail: **inter-agent email coordination**. The finance agent, upon completing its analysis, autonomously identified that the results should be shared with a writer agent for formatting into a board-ready presentation. It sent an email within the AgenticMail system:

```
email_coordination: {
  email_sent_to_writer: true,
  recipient_agent: "writer",
  email_subject: "M&A Due Diligence Summary
  -- TechCorp Acquisition",
  email_purpose: "Sent executive summary
  to writer agent for formatting
  into board-ready presentation"
}
```

This demonstrates a **workflow pattern** unique to Call Agent’s full mode: agents can trigger downstream work

in other agents without orchestrator intervention. Open-Claw Sessions Spawn requires the orchestrator to manually coordinate between agents (spawn A, wait for result, spawn B with A’s output).

6 Aggregate Comparison

6.1 Latency

Table 10: Latency comparison across all tests.

Metric	Call Agent	Sessions Spawn
Tests run	9	16
Minimum	18 s	24 s
Maximum	141 s	107 s
Median	93 s	65 s
Mean	82 s	64 s

Note: In standard/full modes, Call Agent is *not* consistently faster than Spawn. Light mode (~18 s) is the fastest option. Standard mode runtimes (48–141 s) overlap significantly with Spawn (24–107 s). The latency advantage from v1–v3 (which only tested light mode) does not hold across all modes.

6.2 Output Quality

Table 11: Quality comparison on 7 overlapping domains (1–5 scale).

Domain	Call Agent	Spawn
Code Review	5.0	5.0
Finance (AAPL)	4.75	4.5
Healthcare	4.75	4.75
Marketing	5.0	4.75
Supply Chain	4.5	4.5
Data Science	5.0	5.0
Education	5.0	5.0
M&A Finance	5.0	4.75
Average	4.88	4.78

Key finding: Quality is comparable across both paradigms when Call Agent uses standard/full mode. The v2 assumption that Call Agent produces only shallow output is **incorrect**. Both produce expert-level

analysis; they differ in *format* (structured JSON vs. narrative prose), not *depth*.

6.3 Capability Matrix

Table 12: Head-to-head capability ranking.

Criterion	Call Agent	Spawn
Structured output	Always JSON	Free-form
Machine-parseability	Native	Requires parsing
Composability	Programmatic	Manual
Human readability	Raw JSON	Formatted prose
Reasoning depth	High (std/full)	High
Creative output	Structured	Narrative
Tool autonomy	Yes (std/full)	Yes
Inter-agent email	Full mode	Not available
Async execution	Native	Via yield
Parallel agents	Sequential	Native parallel
Session persistence	Stateless	Persistent
Cost efficiency	Lower overhead	Full session

8 OpenClaw Spawn: Unique Capabilities

Sessions Spawn was tested on 9 additional domains not covered by Call Agent, demonstrating capabilities that benefit from session isolation and free-form output:

Table 13: Sessions Spawn: unique test domains.

Domain	Time	Quality	Ref
Legal (SaaS ToS)	53 s	5.0	SS-03
Creative (Story A)	29 s	5.0	SS-07
Creative (Story B)	29 s	5.0	SS-11
Research (RAG)	69 s	4.5	SS-09
M&A Legal	90 s	5.0	SS-13
M&A Operations	107 s	5.0	SS-15
Clinical Stats	88 s	5.0	SS-12
Clinical Regulatory	97 s	5.0	SS-14
Marketing (expanded)	88 s	4.75	SS-16

7 What Changed from v2 to v4

Version 2 of this paper concluded that Call Agent was limited to shallow data retrieval because it was tested only in `light` mode. The v4 tests reveal three corrections:

1. **Call Agent does deep analysis.** In `standard` mode, it produced 9-issue code reviews, 8-red-flag M&A reports, and comprehensive 4-week curricula—all as structured JSON.
2. **Call Agent is not always faster.** Standard/full modes (48–141 s) overlap with Spawn (24–107 s). Only `light` mode provides consistent speed advantage.
3. **Call Agent coordinates between agents.** In `full` mode, the finance agent autonomously emailed the writer agent. This workflow pattern is not possible with Sessions Spawn.

The fundamental difference is not *depth* but *format*: Call Agent returns structured JSON, Spawn returns narrative prose. Both achieve comparable analytical quality.

Multi-agent collaboration highlights:

- **Collaborative story:** Agent A wrote Part 1 ([SS-07](#)); Agent B continued with Part 2 ([SS-11](#)) maintaining perfect tonal consistency. Quality: 5/5.
- **M&A due diligence (3 agents):** Financial ([SS-10](#)), Legal ([SS-13](#)), Operational ([SS-15](#)) agents worked in parallel with zero contradictions. Reports directly combinable.
- **Clinical trial (2 agents):** Statistical ([SS-12](#)) and Regulatory ([SS-14](#)) agents independently identified CV events as underpowered and cited ICH E9(R1). Perfect alignment.

Sessions Spawn’s native parallelism (5–6 agents simultaneously) achieved $4.2\times$ speedup over sequential execution across 16 tasks.

9 When to Use Which

9.1 Decision Framework

Table 14: Decision framework: AgenticMail Call Agent vs. OpenClaw Sessions Spawn.

Scenario	Call Agent	Spawn
Data retrieval (weather, prices)	✓✓	—
Structured API response	✓✓	—
Dashboard/pipeline input	✓✓	—
Code review for CI/CD	✓	✓
Financial analysis	✓	✓
Client-facing report	—	✓✓
Creative writing	—	✓✓
Legal document drafting	—	✓✓
Multi-specialist parallel	—	✓✓
Agent-to-agent workflow	✓✓	—
Long-running research	—	✓✓
Quick async subtask	✓✓	—

9.2 Use AgenticMail Call Agent When:

- Output must be **machine-parseable** (APIs, dashboards, CI/CD)
- You need **structured data** for downstream programs
- The task benefits from **inter-agent email coordination** (use `full` mode)
- **Async fire-and-forget** is desired (standard/full with `async=true`)
- The orchestrator needs to **synthesize data from multiple sources**
- Use `light` for lookups, `standard` for analysis, `full` for workflows

9.3 Use OpenClaw Sessions Spawn When:

- The **output is the deliverable** (reports, briefs, documents, stories)

- You need **multiple specialists in parallel** on complementary aspects
- The task requires **creative or narrative output**
- The sub-agent needs **autonomous tool use** with session persistence
- **Sequential handoff** between agents is needed (story continuation, iterative refinement)
- Human readability matters more than machine parseability

10 Conclusion

This v4 study corrects a significant bias in earlier versions by testing AgenticMail Call Agent across all three execution modes. Our findings:

1. **Both paradigms produce expert-level analysis.** Quality scores are comparable (Call Agent: 4.88/5, Spawn: 4.78/5 on overlapping domains). The difference is format, not depth.
2. **Call Agent always returns structured JSON.** Even for complex M&A due diligence with 8 red flags, the output is machine-parseable. This is its defining advantage.
3. **Sessions Spawn always returns readable prose.** For deliverables that will be read by humans, Spawn’s narrative format requires no post-processing.
4. **Call Agent enables inter-agent email.** In full mode, agents can trigger downstream work in other agents autonomously—a workflow pattern impossible with Sessions Spawn.
5. **Sessions Spawn enables native parallelism.** Running 5–6 specialists simultaneously with $4.2\times$ speedup is a natural pattern for Sessions Spawn but not for Call Agent.

The bottom line: Use Call Agent when you need structured, composable data. Use Sessions Spawn when you need autonomous specialist work with human-readable output. For architects building multi-agent systems: implement both.

All test results are available in the companion directories: [results-callagent/](#) (AgenticMail Call Agent, 9 tests) and [results/](#) (OpenClaw Sessions Spawn, 16 tests).

References

- [1] A. D. Birrell and B. J. Nelson, “Implementing remote procedure calls,” *ACM Trans. Comput. Syst.*, vol. 2, no. 1, pp. 39–59, 1984.
- [2] L. Wang, C. Ma, X. Feng, *et al.*, “A survey on large language model based autonomous agents,” *Frontiers of Computer Science*, vol. 18, no. 6, p. 186345, 2024.
- [3] M. Wooldridge, *An Introduction to MultiAgent Systems*, 2nd ed. John Wiley & Sons, 2009.
- [4] Q. Wu, G. Bansal, J. Zhang, *et al.*, “AutoGen: Enabling next-gen LLM applications via multi-agent conversation,” *arXiv preprint arXiv:2308.08155*, 2023.
- [5] S. Yao, J. Zhao, D. Yu, *et al.*, “ReAct: Synergizing reasoning and acting in language models,” in *Proc. ICLR*, 2023.