# DECOUPLING REASONING FROM ACTION: ARCHITECTURAL IMPACTS ON AGENTIC PLANNING CONSISTENCY

**Himaneesh Sompalle***
Stonehill International School
himaneeshsompalle7@gmail.com

## ABSTRACT

Large Language Models (LLMs) serving as autonomous agents often conflate reasoning (planning) with action (tool execution) within a single context window. We investigate how context architecture impacts logical consistency and planning effectiveness through six ablation studies using MCPBench with GPT-4o, evaluating monolithic, partially partitioned, and fully role-separated architectures under 6–25 concurrent distractors. We find that monolithic architectures maximize raw efficiency and fine-grained accuracy (Dependency Awareness 5.95 vs. 5.67; Parameter Accuracy 8.08 vs. 7.12), while role-separated architectures offer superior scaling stability (1.38× vs. 3.24× latency degradation). These findings reveal a "context-reasoning trade-off" that challenges the assumption that modular agents are universally superior.

## 1 INTRODUCTION

A fundamental challenge in long-horizon agentic planning is the degradation of logical consistency over time. As agents execute tools, their context windows fill with verbose observations, error logs, and retrieval artifacts. In monolithic architectures—where a single LLM instance handles planning, execution, and observation parsing in a unified context—this "context pollution" forces the model to perform high-level reasoning alongside low-level parsing within the same noisy latent space.

While ReAct (Yao et al., 2023) and chain-of-thought prompting (Wei et al., 2022) have improved LLM reasoning, they operate within monolithic context windows by default. The Model Context Protocol (MCP) (Anthropic, 2024) enables multi-server architectures where agent components can be isolated into separate context spaces. Systems such as AutoGPT (Richards, 2023) and LangChain (LangChain, 2023) have explored modular designs, and Personal Knowledge Graphs (PKGs) (Balog & Kenter, 2019) provide formal frameworks for structured state management relevant to the memory component of such architectures. However, systematic empirical comparisons of how architectural separation affects reasoning quality remain scarce.

We hypothesize that **architectural decoupling**—separating the Planner from the Executor—preserves logical consistency by isolating planning state from execution noise. We test this through controlled ablation studies using MCPBench (Zhang et al., 2025), comparing three architectures across six noise levels. Our key finding reveals a nuanced trade-off: monolithic architectures outperform on fine-grained reasoning metrics, while role-separated architectures provide superior scaling stability.

## 2 ARCHITECTURES FOR REASONING ISOLATION

We evaluate three patterns representing increasing degrees of reasoning-action separation:

**Monolithic (1-Server).** A single LLM handles Plan → Act → Observe loops. Reasoning steps, tool calls, and raw outputs coexist in one context window.

*Work done during an internship at Emergence AI.

Table 1: Reasoning fidelity metrics at 6 distractors. Monolithic agents show higher raw accuracy on all fine-grained metrics.

| Metric | Mono | 2-Srv | 3-Srv |
|---|---|---|---|
| Planning Effectiveness | 4.77 | 4.48 | 4.63 |
| Dependency Awareness | **5.95** | 5.65 | 5.67 |
| Parameter Accuracy | **8.08** | 7.30 | 7.12 |
| Tool Appropriateness | **7.80** | 7.36 | 6.94 |
| Grounding | **7.49** | 7.39 | 7.37 |

**Partially Partitioned (2-Server).** Planner and Executor are separate processes sharing state through a common message bus, with partial mutual visibility.

**Role-Separated (3-Server).** Complete decomposition: a *Planner* (pure reasoning, receives sanitized observations), *Executor* (pure action, handles API calls and raw outputs), and *Memory* node (structured intermediary maintaining ground-truth state, analogous to a PKG (Balog & Kenter, 2019)).

## 3 Experimental Setup

**Benchmark.** All experiments use MCPBench (Zhang et al., 2025), a framework built on MCP (Anthropic, 2024) providing multi-step API orchestration tasks that require respecting logical dependencies between tool calls (e.g., creating a resource before updating it, querying data before analyzing it). MCPBench provides standardized task definitions, tool interfaces with schema validation, and evaluation metrics.

**Model.** GPT-4o (`gpt-4o-2024-05-13`) with temperature $T{=}0$ and identical system prompts across all architectures, ensuring observed differences are attributable to architecture alone.

**Distractor protocol.** Six ablation studies inject 6, 10, 14, 18, 22, and 25 concurrent distractor contexts—syntactically valid but irrelevant tool schemas and API responses that compete for model attention. Distractor-to-signal ratios range from $\sim$1:1 to $\sim$4:1. Each study runs 15 tasks per configuration (18 benchmark runs total, December 2025–January 2026).

**Metrics.** We organize measurements into four categories. *Quality*: planning effectiveness, dependency awareness, parameter accuracy, tool appropriateness, and grounding (0–10 scale via programmatic checks and GPT-4o-as-judge). *Robustness*: tool call success rate and task completion rate. *Efficiency*: execution time (wall-clock seconds) and total tokens (input + output, thousands).

## 4 Results

All architectures achieved **100% task success** across all distractor levels, allowing us to focus on reasoning quality and scaling behavior.

### 4.1 Reasoning Quality Under Low Noise

Table 1 presents reasoning fidelity at 6 distractors, where architectural differences are most visible.

The monolithic architecture outperforms on *all* fine-grained quality metrics at low noise: dependency awareness (5.95 vs. 5.67), parameter accuracy (8.08 vs. 7.12), tool appropriateness (7.80 vs. 6.94), and grounding (7.49 vs. 7.37). This counter-intuitive result suggests that GPT-4o benefits from seeing raw execution traces when making logical deductions—complete isolation deprives the Planner of subtle context cues in tool outputs.

Table 2: Task completion scores across distractor levels (higher indicates better alignment).

| Distractors | Monolithic | 2-Server | 3-Server |
|---|---|---|---|
| 6 | 6.418 | 6.339 | 6.450 |
| 10 | 6.012 | 6.099 | 6.216 |
| 14 | 6.351 | 6.536 | 6.398 |
| 18 | 6.427 | 6.303 | 6.440 |
| 22 | 6.320 | 6.353 | 6.461 |
| 25 | 6.397 | 6.390 | 6.499 |
| **Mean** | 6.321 | 6.337 | **6.411** |

Table 3: Efficiency metrics and scaling behavior across distractor levels (6–25).

| | Exec. Time (s) | | | Tokens (K) | | |
|---|---|---|---|---|---|---|
| **Distractors** | **Mono** | **2-Srv** | **3-Srv** | **Mono** | **2-Srv** | **3-Srv** |
| 6 | 71 | 108 | 189 | 83 | 132 | 132 |
| 10 | 123 | 164 | 192 | 112 | 151 | 160 |
| 14 | 161 | 191 | 251 | 129 | 167 | 206 |
| 18 | 95 | 192 | 196 | 216 | 277 | 275 |
| 22 | 99 | 186 | 191 | 246 | 320 | 337 |
| 25 | 231 | 244 | 260 | 212 | 238 | 264 |
| **Scale (6→25)** | 3.24× | 2.27× | **1.38**× | 2.56× | 1.81× | 2.00× |

### 4.2 Task Completion Across Noise Levels

Table 2 shows task completion scores across all six distractor levels. Despite lower fine-grained scores, the 3-Server architecture achieves marginally higher mean task completion (6.411 vs. 6.321), with the advantage becoming more consistent at higher distractor counts.

### 4.3 Tool Call Reliability

Tool call success rate directly impacts practical reliability. Monolithic and 3-Server achieve comparable rates (96.99% vs. 96.83%), while the 2-Server consistently underperforms (94.29%). Notably, the 3-Server achieves its highest tool success (98.05%) at 18 distractors, suggesting role separation becomes increasingly beneficial as noise grows (full per-level data in Appendix B).

### 4.4 Efficiency and Scaling Stability

Table 3 presents efficiency across all distractor levels.

Monolithic architectures are 1.2–2.7× faster and consume 1.2–1.6× fewer tokens in absolute terms. However, the critical insight is in *degradation rate*: monolithic latency scales 3.24× from 6 to 25 distractors, while the 3-Server architecture scales only 1.38×. Token consumption tells a complementary story: monolithic tokens increase 2.56× while 3-Server increases 2.00×, reflecting the constant overhead of context synchronization. The 3-Server is also the only architecture where task completion and tool success *improve* slightly under higher noise (scaling ratios of 1.008×; see Table 5 in Appendix B).

## 5 Discussion

**The context-reasoning trade-off.** Our results reveal a fundamental tension between two forms of reasoning robustness. Monolithic architectures excel at *micro-reasoning*—fine-grained logical deductions that benefit from dense access to execution traces. When GPT-4o can observe the raw JSON response from a prior tool call, it more reliably extracts correct parameters for subsequent

calls. These execution traces effectively function as implicit chain-of-thought (Wei et al., 2022), and abstracting them away removes signals the model has learned to exploit.

Role-separated architectures instead excel at *macro-stability*—maintaining consistent performance as noise scales, because the Planner's reasoning context is shielded from execution pollution. The 3-Server architecture's 1.38× latency scaling versus 3.24× implies that for long-horizon tasks where context windows approach capacity, decoupled agents will maintain usability far longer than monolithic ones. This trade-off means the optimal architecture depends on deployment context: low-noise environments with precision requirements favor monolithic designs, while high-noise, long-horizon settings favor role separation.

**The partial partitioning anti-pattern.** The 2-Server architecture consistently delivers the worst trade-off: lower tool reliability (94.29% vs. 96.99% monolithic, 96.83% three-server), efficiency costs comparable to full separation, and no compensating quality gains. Partial state sharing introduces synchronization overhead without the reasoning benefits of either full context access or full isolation. This finding is particularly relevant for practitioners who might consider partial partitioning as a pragmatic middle ground—our data suggests it is worse than either extreme.

**Implications for LLM reasoning.** The monolithic advantage at low noise suggests that current LLMs have not learned to reason effectively from abstracted summaries alone. The model leverages raw execution details—including error messages, response formatting, and status codes—in ways that sanitized summaries cannot replicate. This has implications for tool-use interface design: overly aggressive output filtering may inadvertently harm reasoning quality, and future work on context compression should preserve these reasoning-relevant signals.

## 6 Limitations and Future Work

**Single model.** All experiments use GPT-4o. Different model families (Claude, Gemini, Llama) may exhibit different sensitivity to context pollution due to variations in attention mechanisms, context window management, and training data composition. Cross-model replication is a critical next step to determine whether the context-reasoning trade-off is a general phenomenon or GPT-4o-specific.

**Synthetic noise.** Our distractors are syntactically valid but semantically irrelevant. Real-world noise includes partially relevant information that could either help or hinder reasoning in ways our protocol does not capture.

**Evaluation bias.** Using GPT-4o as both agent and judge introduces potential self-evaluation bias. Human evaluation or cross-model judging would strengthen the validity of our quality metrics.

**Future directions** include adaptive architectures that switch between modes based on detected noise, structured memory components that preserve reasoning-relevant signals while filtering noise, and extension to diverse agentic tasks such as code generation and multi-modal reasoning.

## 7 Conclusion

For tasks requiring high logical precision with limited noise, monolithic architectures are superior reasoning engines—they offer the lowest latency, fewest tokens, and highest fine-grained accuracy by giving the LLM dense access to execution traces. However, for long-horizon tasks where context pollution threatens the agent's reasoning coherence, role-separated architectures provide a necessary "cognitive firewall," maintaining stable performance (1.38× latency scaling vs. 3.24×) albeit at a high efficiency cost. Partial partitioning should be avoided: the 2-Server configuration consistently delivers the worst of both worlds. These findings challenge the prevailing assumption that modular agents are universally superior and ground the choice of context architecture in empirical evidence about its impact on logical reasoning.

### Acknowledgments

We thank the anonymous reviewers for constructive feedback that improved the experimental exposition and limitations discussion.

## REFERENCES

Anthropic. Model context protocol: Open standard for AI-native tool integration. https://modelcontextprotocol.io, 2024.

Krisztian Balog and Tom Kenter. Personal knowledge graphs: A research agenda. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, pp. 217–220, 2019.

LangChain. LangChain: Building applications with LLMs through composability. https://langchain.com, 2023.

Toran Richards. Auto-GPT: An autonomous GPT-4 experiment. https://github.com/Significant-Gravitas/AutoGPT, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Wei Zhang, Chen Liu, and Hao Wang. MCPBench: A benchmark for evaluating MCP-based agentic systems, 2025.

## A INFERENCE-TIME COMPUTE AND INTERNALIZED REASONING

An important observation from our results is the strong parity between single-server and three-server tool success rates (96.99% vs. 96.83%). Modern inference-time compute LLMs such as GPT-4o perform substantial agentic looping internally—these models effectively internalize the scaffolding of memory, execution, and planning through their extended reasoning capabilities. This may explain why the monolithic architecture achieves reasoning performance comparable to explicit external role separation without the overhead of distributed systems: the model already implements an implicit form of role separation within its own reasoning process.

This raises a key question for the logical reasoning community: as models become more capable of internal chain-of-thought and self-correction, does the marginal benefit of external architectural decoupling diminish for reasoning quality, persisting only for scaling stability? Our data suggests the answer is nuanced—monolithic reasoning benefits from raw trace access in ways that even strong internal reasoning cannot fully compensate for at low noise, but this advantage erodes as context pollution grows.

## B COMPLETE PER-LEVEL DATA

### B.1 TOOL CALL SUCCESS RATES

Table 4: Tool call success rates (%) across all distractor levels.

| Distractors | Monolithic | 2-Server | 3-Server |
|---|---|---|---|
| 6 | 97.39 | 94.15 | 95.70 |
| 10 | 97.58 | 93.78 | 97.78 |
| 14 | 97.06 | 94.55 | 96.25 |
| 18 | 96.18 | 94.93 | 98.05 |
| 22 | 97.09 | 94.25 | 96.66 |
| 25 | 96.63 | 94.06 | 96.51 |
| **Mean** | **96.99** | 94.29 | 96.83 |

### B.2 Scaling Behavior

Table 5: Scaling behavior from 6 to 25 distractors (ratio of high-noise to low-noise values). Values near 1.0× indicate stability.

| Metric | Monolithic | 2-Server | 3-Server |
|---|---|---|---|
| Execution Time | 3.24× | 2.27× | **1.38**× |
| Token Usage | 2.56× | 1.81× | 2.00× |
| Task Completion | 0.997× | 1.008× | **1.008**× |
| Tool Success Rate | 0.992× | 0.999× | **1.008**× |

### B.3 Benchmark Reliability Summary

Table 6 confirms that all architectures achieved 100% task success across all six studies, demonstrating that functional reliability is robust to architectural choice under the tested conditions.

Table 6: Benchmark reliability across all studies. ✓ indicates 100% task success.

| Study | Distractors | 1-Server | 2-Server | 3-Server |
|---|---|---|---|---|
| Dec 2025 (A) | 6 | ✓ | ✓ | ✓ |
| Dec 2025 (B) | 10 | ✓ | ✓ | ✓ |
| Dec 2025 (C) | 14 | ✓ | ✓ | ✓ |
| Jan 2026 (D) | 18 | ✓ | ✓ | ✓ |
| Jan 2026 (E) | 22 | ✓ | ✓ | ✓ |
| Jan 2026 (F) | 25 | ✓ | ✓ | ✓ |

## C Metric Definitions

We provide detailed definitions for all evaluation metrics used in this study.

**Functional Reliability.** *Task success rate*: binary pass/fail indicating whether the agent completed the assigned task. *Completion score*: continuous score (0–10) measuring the degree of alignment between agent behavior and task requirements, assessed by GPT-4o-as-judge.

**Quality and Alignment.** *Planning effectiveness*: evaluates whether the agent's strategy was optimal and the task was decomposed into appropriate sub-goals. *Dependency awareness*: measures whether the agent respected logical prerequisites between tool calls (e.g., querying before updating). *Parameter accuracy*: assesses whether API parameters were correctly inferred and populated, scored programmatically. *Tool appropriateness*: evaluates whether the correct tools were selected for each sub-task. *Grounding*: measures the factual accuracy and relevance of agent outputs relative to task context.

**Technical Robustness.** *Tool call success rate*: percentage of tool invocations that completed without errors. *Schema compliance*: whether tool calls conform to the expected API schema. *Valid tool identification*: whether the agent correctly identified available tools from among valid options and distractors.

**Efficiency.** *Execution time*: wall-clock seconds from task initiation to completion. *Total tokens*: sum of input and output tokens across all LLM calls for a given task, reported in thousands. *Tool calls per task*: number of distinct tool invocations required to complete a task.