@slowwww.ai

from Anthropic

How we built our multi-agent research system

< The missing manual >

SlowwAI
2025-06-16

The missing manual

@slowwww.ai

Why, what, and how

Why this manual?

- When I first picked up Anthropic's "How we built our multi-agent research system," it felt like a goldmine.
- It contains all the trial-and-error Anthropic had to go through when building a world-wide AI product.
- This means that this article contains all the information that can be used to further enhance my own agent.

What you can get out of this manual.

- I rearranged their agent and their system architecture so that you can pin point the EXACT location for improvement, and HOW to improve them.
- Thus, by the end of this manual, you will be able to
  1. Understand the whole architecture of how Anthropic builds it agents
  2. Pinpoint exactly where every "challenge" or "improvement" in the article lands inside the system.
- For anyone who likes to build with their own hands:
  - When you want the step-by-step process of building an agent from scratch
  - When your agent sits at 90 percent performance and you need that last ten

Overall architecture

@slowwww.ai

The architecture of Anthropic's research agent

The architecture can be restructured as follows :

Overall architecture

@slowwww.ai

Research agent rearranged

Strategies to enhance agent performance

Three-tier breakdown of areas to improve

@slowwwww.ai

Strategies to enhance agent performance

Three-tier breakdown of areas to improve

@slowwww.ai

Strategies to enhance agent performance

Areas for refinement - Agent

1. Understand your agents

- Watch agents work step-by-step to understand where they fail
- Generate logs in order to pinpoint optimization opportunities

2. Teach the orchestrator how to delegate

- Warning : each agent's task should not overlap
- Test the lead agent whether it clearly defines an objective, an output format, guidance on the tools and sources to use, and clear task boundaries for each sub agent

3. Scale effort to query complexity

- Simple fact-finding : 1 agent with 3-10 tool calls
- Direct comparisons: 2-4 subagents with 10-15 calls each
- Complex research: more than 10 subagents with clearly divided responsibilities

4. Let agents improve themselves

- A tool-testing agent would rewrite the description for each tool to avoid failures -> resulted in a 40% decrease in task completion time

@slowwww.ai

Strategies to enhance agent performance

Areas for refinement - Agent

5. [Prompt] Start wide, then narrow down

- Agents often default to overly long, specific queries that return few results.
- Counteracted this tendency by prompting agents to start with short, broad queries, evaluate what's available, then progressively narrow focus

6. [Prompt] Guide the thinking process

- Turn on "extended thinking" mode
- Extended thinking improves instruction-following, reasoning, and efficiency

7. Parallel tool calling transforms speed and performance

- (1) the lead agent spins up 3-5 subagents in parallel
- (2) the subagents use 3+ tools in parallel

@slowwww.ai

Strategies to enhance agent performance

Areas for refinement - Agent evaluation

Evaluation

Evaluating multi-agent systems presents unique challenges.

Traditional evaluations often assume that the AI follows the same steps each time: given input X, the system should follow path Y to produce output Z. But multi-agent systems don't work this way.

Even with identical starting points, agents might take completely different valid paths to reach their goal. One agent might search three sources while another searches ten, or they might use different tools to find the same answer. Because we don't always know what the right steps are, we usually can't just check if agents followed the "correct" steps we prescribed in advance.

Instead, we need flexible evaluation methods that judge whether agents achieved the right outcomes while also following a reasonable process.

@slowwww.ai

Strategies to enhance agent performance

Areas for refinement - Agent evaluation

1. Start evaluating immediately with small samples

- Start evaluating in the early stage of the project.
- Small changes will bring big improvements
- 20 queries would be enough

2. LLM-as-judge evaluation scales when done well

- Evaluate based on the following criteria: factual accuracy, citation accuracy, source quality, tool efficiency
- A single LLM call with a single prompt outputting scores from 0.0-1.0 and a pass-fail grade best simulates human evaluation

3. Human evaluation catches what automation misses

- Human evaluation is necessary, to find edge cases : hallucinated answers on unusual queries, system failures, or subtle source selection biases.

Evaluation

Evaluating multi-agent systems presents unique challenges.

Traditional evaluations often assume that the AI follows the same steps each time: given input X, the system should follow path Y to produce output Z. But multi-agent systems don't work this way.

Even with identical starting points, agents might take completely different valid paths to reach their goal. One agent might search three sources while another searches ten, or they might use different tools to find the same answer. Because we don't always know what the right steps are, we usually can't just check if agents followed the "correct" steps we prescribed in advance.

Instead, we need flexible evaluation methods that judge whether agents achieved the right outcomes while also following a reasonable process.

@slowwww.ai

Strategies to enhance agent performance

Production level

System

Areas for refinement - Production / system

Factors that influence performance

1. When it doesn't work as intended, check out the token usage first

- The easiest and most effective way to improve the overall performance of the agent is to increase token usage
- The more thought the agent put into the problem, the more better it gets
- The downside is that it burns through tokens fast

@slowwww.ai

Strategies to enhance agent performance

Areas for refinement - Production / system

@slowwww.ai

Strategies to enhance agent performance

Areas for refinement - Production / system

3. Subagent output to a filesystem to minimize the 'game of telephone

- Allow sub agents to call tools and store their work in external systems, then pass lightweight references back to the coordinator.
- This prevents information loss during multi-stage processing and reduces token overhead from copying large outputs through conversation history.
- The pattern works particularly well for structured outputs like code, reports, or data visualizations where the subagent's specialized prompt produces better results than filtering through a general coordinator.

Strategies to enhance agent performance

Production level

System

Areas for refinement - Production / system

4. Agents are stateful and errors compound

- When errors occur, we can't just restart from the beginning: restarts are expensive and frustrating for users. Instead, we built systems that can resume from where the agent was when the errors occurred
- We also use the model's intelligence to handle issues gracefully: for instance, letting the agent know when a tool is failing and letting it adapt works surprisingly well
- Adding try-except is another way

5. Debugging benefits from new approaches

- Try to identify the pattern of the agents. This will unlock a lot of areas for refinement. Must add Loggers to the system.
- Beyond standard observability, monitor agent decision patterns and interaction structures
- This high-level observability helped us diagnose root causes, discover unexpected behaviors, and fix common failures.

@slowwww.ai

Strategies to enhance agent performance

Production level

System

Areas for refinement - Production / system

6. Deployment needs careful coordination

- Whenever we deploy updates, agents might be anywhere in their process.
- We therefore need to prevent our well-meaning code changes from breaking existing agents.
- Thus, gradually shifting traffic from old to new versions while keeping both running simultaneously allows you to maintain well-working agents.

7. Synchronous execution creates bottlenecks

- Synchronously simplifies coordination, but creates bottlenecks in the information flow
- Asynchronous execution would enable additional parallelism: agents working concurrently and creating new subagents when needed. But this asynchronicity adds challenges in result coordination, state consistency, and error propagation across the subagents. As models can handle longer and more complex research tasks, we expect the performance gains will justify the complexity.