

Anthropic's Claude speed sprint: give agents a benchmark that can only improve, then fan out

Coding Agents Alpha Tracker

2026-10-05

Anthropic's Claude speed sprint: give agents a benchmark that can only improve, then fan out

By Coding Agents Alpha Tracker • October 5, 2026

Theo breaks down how Anthropic used Claude to make Claude.ai about 3x faster, and the benchmark guardrail pattern he reuses in T3 Code. Also: DHH on choosing Rust when agents write the code, Opus 5.5 vs GPT-6 Astra, and always-on “AI sheds”.

How to run agent-driven performance work

Theo went through Anthropic's write-up of a two-week sprint that made the core Claude.ai and desktop experience about 3x faster. The sprint covered four journeys that make up 95% of user activity. At p75, a fresh load went from 3.1s to 0.55s, and starting a Claude Code session went from 0.8s to 0.3s [1]. The method carries over to other codebases:

1. **Measure from the user's side first.** Claude analyzed usage through the Datadog MCP server and picked the highest-impact journeys. Each measurement started at a user interaction, ended when the result was rendered, and separated client work from server work [1].
2. **Use deterministic proxies, but check them.** Wall-clock time is noisy, so the team used instruction counts under Valgrind for pure JS paths. For the browser they used React commits, V8 call counts, style recalculations and DOM mutations [1]. Every benchmark had two jobs: give Claude a number to lower in the lab, and act as a CI guardrail that could only go down. Benchmarks that were flaky or didn't track user latency were thrown out [1].
3. **Ratchet.** After Claude cut instructions on two hot paths by 48% and 31%, any PR that raised those counts failed CI, and a daily job lowered the ceiling whenever the count dropped [1].

4. **Loop in a shared Slack channel.** Someone opens a thread about a slow step. Claude traces it, builds a benchmark, and sends PRs sized for risk, with anything user-visible behind a flag. It then reads field data and either ratchets the benchmark or turns the flag off [1]. About 200 flags were added over the sprint, and more than half were cleaned up by the end [1].

Theo's caveat: optimizing for counts alone pushes agents to cut network requests that don't matter. He suspects this caused a bug he found live, where deleted threads stayed in Claude's cached sidebar after a refresh [1]. He still keeps performance work in the loop. He asks the agent for theories, has it build demos of each, then tests which feels faster himself [1].

His own version: an agent-written GitHub Action benchmarks T3 Code's request formats against stored thread data and comments on PRs when results drift from the baseline, so coding and review agents can fix regressions themselves [1]. With that in place, he fanned out agents and 38 of 40 performance PRs auto-merged [1]. For exploratory work he says to explicitly ask for "extreme" options with several proposals ("Don't be afraid to boil the ocean"). That prompt led him to fork Rusty V8 into a custom runtime, which he reports gave a 2-4x improvement [1].



How Anthropic made Claude 3x faster (28:40)

DHH: if you don't read the code, reconsider the language

DHH had agents implement and optimize Basecamp's Campfire in Elixir, Go and Rust. He asks whether Ruby's slowness still matters "if you're no longer reading the code" [2]. Agents have also finished Laravel and Django ports. He frames the exercise as a test of how frontier agents handle each environment out of the box [3]. The Rust version is more than 10x the lines of code of Rails, which he says mattered far more when humans wrote the code [4]. He also says agents write much faster Rust than Elixir or Go [5]. Caveats: Rust compiles slowly, and he says most apps don't need Rust-level performance [6, 7]. His sharper claim is that any language agents can't write well out of the box "is going to have a hard time" [8]. This echoes Huntley's argument from the last brief that compiler back-pressure keeps agents on the rails.

Opus 5.5 vs GPT-6 Astra: daily driver vs reviewer

Theo now says preferring OpenAI models for code "makes almost no sense." In his view Opus 5.5 is fast and surprisingly cheap, while OpenAI models are slow without fast mode and burn through a \$200 plan in hours [9]. His vibe rankings: Opus 5.5 scores 9/10 on code and 9/10 on understanding intent; Astra scores 7.5/10 and 3/10 [10].

A narrower pattern has support from others. Robin Ebers uses Opus day to day but has Astra run an adversarial review of plans Opus wrote. About 30% of Astra's feedback is over-engineering, and he says 70% points to real holes [11]. Riley Brown keeps Astra for complex, high-stakes tasks and uses Opus for frontend, documents and most coding, which feels unlimited on the \$200 plan [12].

Where agents run

DHH recommends an "AI shed": an always-on machine on your Tailscale network that runs most of your agents. Compiles happen there, so a modest laptop is enough and closing the lid doesn't stop the agents [13, 14]. His main shed is a Minisforum MS-A2, and he says a Beelink box with an 8745HS or similar handles most work [15]. Theo announced t3os on the same theme: a headless, mostly Ubuntu-based OS for "personal servers," built from tech choices agents prefer, with no ISO and not meant for a computer you use [16]. OpenAI's Tibo said in a podcast that specialist dots get extra guardrails and run on their own hardware, sometimes Mac minis, and that one dot's harness can control many devices [17].

Smaller items

- **Verified hot-reload:** Geoffrey Huntley had an OpenAI model inspect a running SBCL process and add **reverse-string** through a tool call. The change was saved only after it passed 21 fixed cases and 1,000 generated

ones. A fresh SBCL process passed the same checks, and rolling back and restoring the revision both worked [18].

- **Codex:** Tibo says that for the next 28 days, Codex/Work will ship one clear improvement relevant to most users each day, or do a full usage reset [19].
- **Agent count:** Tibo now runs fewer agents in parallel because ultrafast mode lets him stay in flow. He grows his team of agents on frontier work and shrinks it again after model jumps [17].
- **Durability:** Armin Ronacher says making an agent “somewhat durable” is easy, but a harness that is durable by design without becoming “a crazy mess” took many iterations [20].
- **Outputs as artifacts:** Building on Karpathy’s writing tip, Omar Sar has his agent produce artifacts suited to the task, on Notion-like pages, instead of plain text. He reviews them through comments and suggestions, including for code review [21].

Sources

1. How Anthropic made Claude 3x faster
2. X post by @dhh
3. X post by @dhh
4. X post by @dhh
5. X post by @dhh
6. X post by @dhh
7. X post by @dhh
8. X post by @dhh
9. X post by @theo
10. X post by @theo
11. X post by @robinebers
12. X post by @rileybrown
13. X post by @dhh
14. X post by @dhh
15. X post by @dhh
16. X post by @theo
17. OpenAI’s Head of ChatGPT: We’re entering a new era of AI (again) | Tibo Sottiaux
18. X post by @GeoffreyHuntley
19. X post by @thsottiaux
20. X post by @mitsuhiko
21. X post by @omarsar0