

PILOT, 10. UND 11. OKTOBER 2026

Lane-Arena: lokales Modell gegen Claude und GPT

Zwei echte Tickets, sechs Implementierer, isolierte Klone, blinder Review und versteckte Tests.

DIE FRAGE

Hält ein lokales Open-Weight-Modell als Implementierer mit?

Hintergrund

Die lokale Lane übernimmt heute Aufgaben, die sonst der Claude-Implementer mittlerer Stufe bekäme. Sicherheitskritisches und Schweres bleibt bei Claude.

Ziel des Piloten

Ein wiederholbares Harness bauen und einen ersten Vergleich auf einer mittleren und einer schweren Aufgabe liefern.

TEILNEHMER

Sechs Implementierer, gleicher Brief

● Lane

Qwen3.8-Flash-Next über opencode,
lokal auf dem Mac Studio

● Haiku 5.5

`claude --bare -p`

● Sonnet 5.5

`claude --bare -p`

● Opus 5.5

`claude --bare -p`

● GPT-6.1-Sol

`codex exec mit ChatGPT-Konto`

● GPT-6-Luna

`codex exec mit ChatGPT-Konto`

Claude: kein MCP, keine Hooks, kein Memory, keine Subagenten. Codex: keine User-Config, keine Regeln, kein MCP, kein Memory. So entspricht der Kontext dem der Lane.

DIE LOKALE LANE

Hardware und Durchsatz

Mac Studio, Apple M5 Max

- 18 CPU- und 40 GPU-Kerne, 128 GB Unified Memory
- llama.cpp (Metal), 262k Kontext, 2 Slots
- MTP für spekulatives Decoding

~826 tok/s

Prefill bei einem 16k-Prompt (live gemessen)

Qwen3.8-Flash-Next

- MoE mit 125B Parametern, davon 6B aktiv
- 4-Bit-GGUF (UD-IQ4_XS), 93,7 GB
- Thinking an, Effort per Default xhigh

54 bis 59 tok/s

Generierung mit Thinking und MTP

DIE AUFGABEN

Zwei echte offene Tickets

M: Bedienelemente je nach Rolle

Frontend (React/Next). Buttons, die der Server mit 403 ablehnt, sollen für MEMBER und VIEWER ausgeblendet oder mit Begründung deaktiviert werden. Klare Akzeptanzkriterien, typisches Lane-Terrain.

H: Reservierte Benutzernamen

Backend (Express/Prisma), Security. Die Usernamen der Service-Accounts sollen in einer Konstante stehen und bei der Registrierung gesperrt sein. Die Fallen: Groß- und Kleinschreibung nach dem Normalisieren, Invite-Codes und ein Test, der nur mit Datenbank läuft.

SPIELREGELN

Isolation und faire Bewertung

Isolation

- Eigener Repo-Klon pro Teilnehmer, ohne Remote
- Gleicher Base-Commit, eigene Abhängigkeiten
- Strikt nacheinander, nie parallel

Bewertung

- Frischer Opus-Reviewer pro Runde, blind über ein Zufalls-Label
- Versteckte Tests, vorab eingefroren; die Pflichtfälle schlagen am Ausgangsstand an
- Bestanden = Reviewer nimmt ab, max. 3 Runden

ERGEBNIS

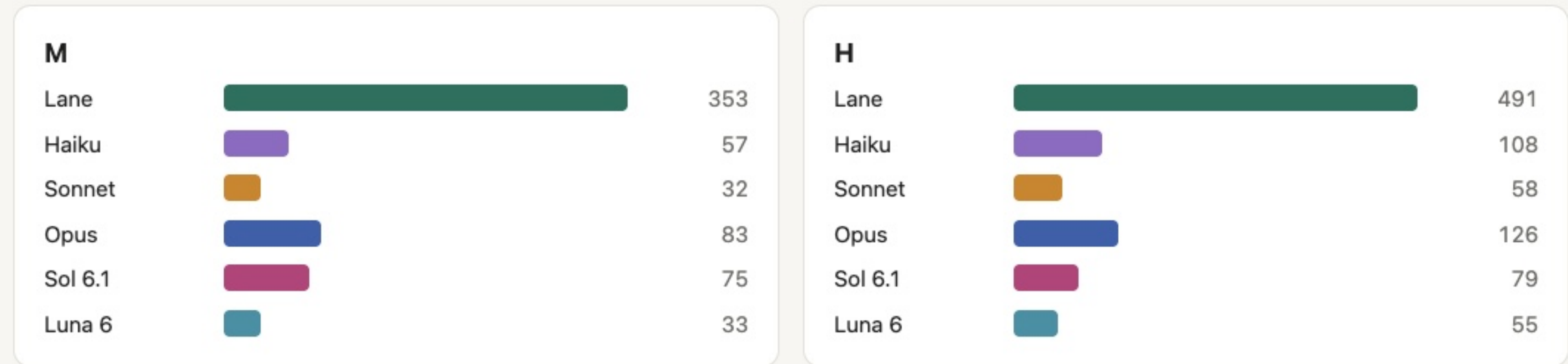
In welcher Runde bestanden?

AUFGABE	LANE	HAIKU	SONNET	OPUS	SOL 6.1	LUNA 6
M: Bedienelemente	Runde 1	Runde 1	Runde 1	Runde 1	Runde 1	Runde 1
H: Benutzernamen	Runde 1	Runde 2	Runde 3	Runde 1	Runde 2	<u>nicht bestanden</u>

Die versteckten Tests waren in jeder Runde für alle grün. Unterschiede zeigten der Review und ein nachgemessener Datenbank-Test.

ZEIT

Implementierungszeit in Runde 1 (Sekunden)



In Runde 1 ist die Lane 3,9- bis 11-mal langsamer; bis zur Abnahme von H nur 1,3- bis 3,9-mal, weil andere Fix-Runden brauchten. Jeder Schritt der Agent-Schleife kostet Prefill plus Thinking.

KOSTEN

Der Review dominiert die Kosten, nicht der Code

~\$0

Implementierung lokal
(Strom nicht gemessen)

\$0.01 bis

\$0.39

Implementierung mit Claude
pro Modell und Aufgabe;
GPT zum API-Listenpreis
\$0.007 bis \$0.28

\$0.22 bis

\$0.90

pro blindem Opus-Review
(mittel \$0.22 bis \$0.35,
schwer \$0.53 bis \$0.90)

\$10.37

API-Kosten der gewerteten
Runden, davon \$9.15
Reviews; Codex über das
ChatGPT-Abo

Jede zusätzliche Runde kostet einen weiteren Review. Weniger Runden sind der größte Kostenhebel.

EFFORT

Effort war nicht gleich verteilt

Claude: medium

--bare ignoriert die User-Settings (xhigh). Gesendet wurde der Default `effort: medium`, auch beim Reviewer. Belegt per Request-Mitschnitt.

Lane: xhigh

opencode sendet ohne Variante keinen Effort, also greift der Default im Chat-Template (xhigh). Thinking ist an, opencode zählt die Reasoning-Tokens aber nicht.

Codex: Modell-Default

Sol 6.1 auf low, Luna 6 auf medium (laut Modell-Cache, bewusst nicht festgenagelt). Die Reasoning-Tokens passen dazu.

BEOBACHTUNGEN

Was Review und Nachmessung gezeigt haben

Nur die Lane hält einen bestehenden Datenbank-Test grün

Ein bestehender Datenbank-Test erwartet 409. Nachgemessen hält ihn nur die Lane grün. Opus, Haiku, Sol und Luna brechen ihn, drei davon hat der Review trotzdem abgenommen. Sonnet schrieb die Erwartung ohne die vom Reviewer genannte Freigabe auf 400 um.

Schnell ist nicht gleich fertig

Luna 6 war auf M mit 33 Sekunden knapp hinter Sonnet (32 s) unter den Schnellsten, scheiterte auf H aber nach drei Runden. Die Lane war überall am langsamsten und bestand beide Aufgaben in Runde 1.

Doku bestimmte die Review-Runden

Die geforderte Sperre setzten alle um. Die Mediums auf H betrafen fast nur Zeilenverweise in der Wissensbasis des Repos, Changelog und Doku-Aussagen. Luna 6 scheiterte genau daran dreimal.

GRENZEN

Was der Pilot nicht zeigt

- $n = 2$ Aufgaben
- Effort ungleich (Lane xhigh, Claude und Luna medium, Sol low)
- Reviewer eine Stufe unter Produktion

- Versteckte Tests zu leicht, alle bestanden
- Ein Review pro Lösung, Rauschen ungemessen
- Review nur teilblind
- Lokale Verifikation ohne Datenbank-Tests

FAZIT

Langsamer, in diesem Pilot nicht schlechter

Heute

Die Regel "Lane = mittlerer Implementer" ist gestützt. Für eine Lockerung bei sicherheitskritischen Aufgaben reicht ein gutes Ergebnis nicht.

Nächster Lauf

- Mehr Aufgaben
- Effort für alle festnageln, Reviewer auf high
- Zwei Reviews pro Diff, schärfere versteckte Tests