AGENTIC ENGINEERING

Reliable work from *unreliable* agents

Where agentic engineering stands, what it could do for science, and what we still have to invent.

Anthony Kougkas

Associate Research Professor, Illinois Institute of Technology

Executive Director, Gnosis Research Center

Guest Faculty, Argonne National Laboratory

akougkas@illinoistech.edu · November 2026

HANDS UP, PLEASE

Where are you with AI?

1 A chat window, and you copy and paste the answers

2 Autocomplete in your editor

3 An agent in your terminal, and you approve each step

4 Agents that keep working while you sleep

5 Agent-written code behind a figure in a paper you submitted

Keep your hand up if you could regenerate that figure tomorrow.

Nobody chats with a CPU.

You program a CPU.

AUTOMATA STUDIES, PRINCETON, 1956

Probabilistic Logics and the Synthesis of *Reliable Organisms* from *Unreliable Components*

John von Neumann, from lectures given at Caltech in 1952

He asked the right question. You will not get a perfect part. What organization makes the whole reliable anyway? And he named the danger: “The machine remembers its mistakes, so to speak, and thereafter perpetuates them.”



THE CLAIM FOR THIS HOUR

A coding agent is a system built around one brilliant, persuasive, unreliable component. *Everything we call engineering lives in what we build around it.*

THE HOUR

Five movements

I	**We have been here before**	FORTRAN, a curve, and an uncomfortable chart
II	**A new kind of computer**	Where the bottleneck went
III	**Five ideas**	Hierarchy, containment, many minds, evidence, sovereignty
IV	**The dream**	What science looks like if this works
V	**What stands between us**	Open problems you could take on this year

The agent in this lecture is Clio Coder. Open source. It is the instrument for all of this, and you can take it apart.

I

We have been here before

Every generation of programmers has met the machine that writes the code, and most of them doubted it first.

THE LAST TIME THIS HAPPENED

Your field's first agent was called a *compiler*

Grace Hopper, oral history, 1980

“It was very stupid. What I did was watch myself put together a program and make the computer do what I did.”

On the word automatic: “We still had to tell the compiler what to compile. It wasn't automatic.”

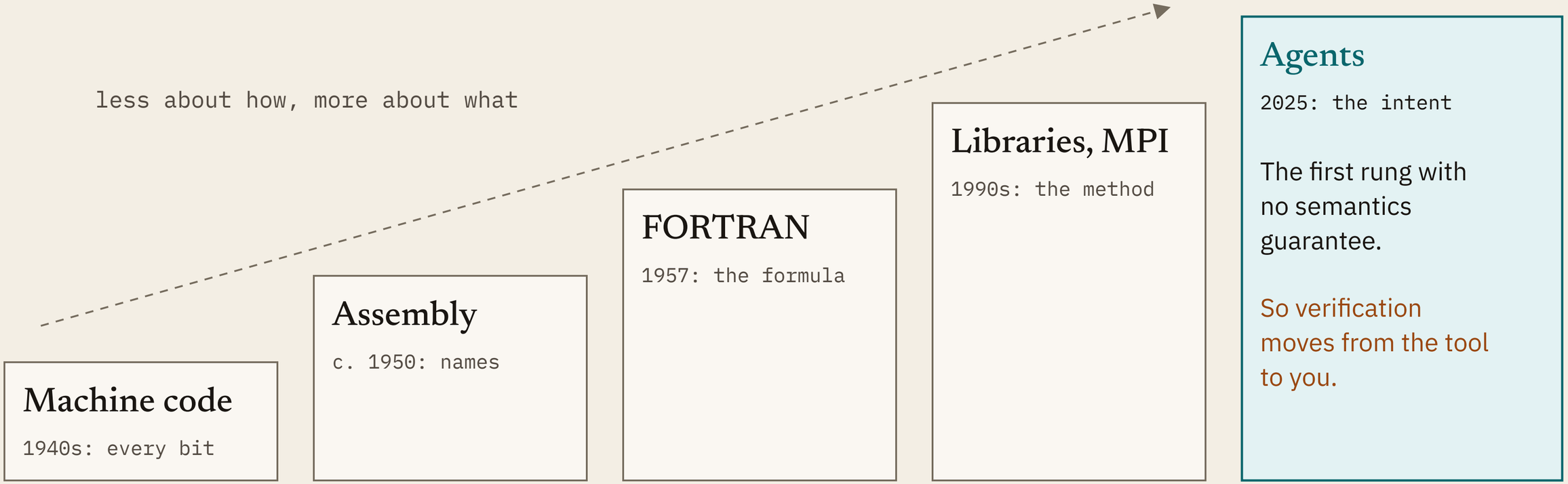
John Backus and FORTRAN, IBM 704, 1957

Programmer cost had caught up with machine cost. Backus's team promised hand-coded speed “on the basis more of faith than of knowledge,” and then made it true.

Nearly seventy years later, FORTRAN still runs in production codes at the national laboratories.

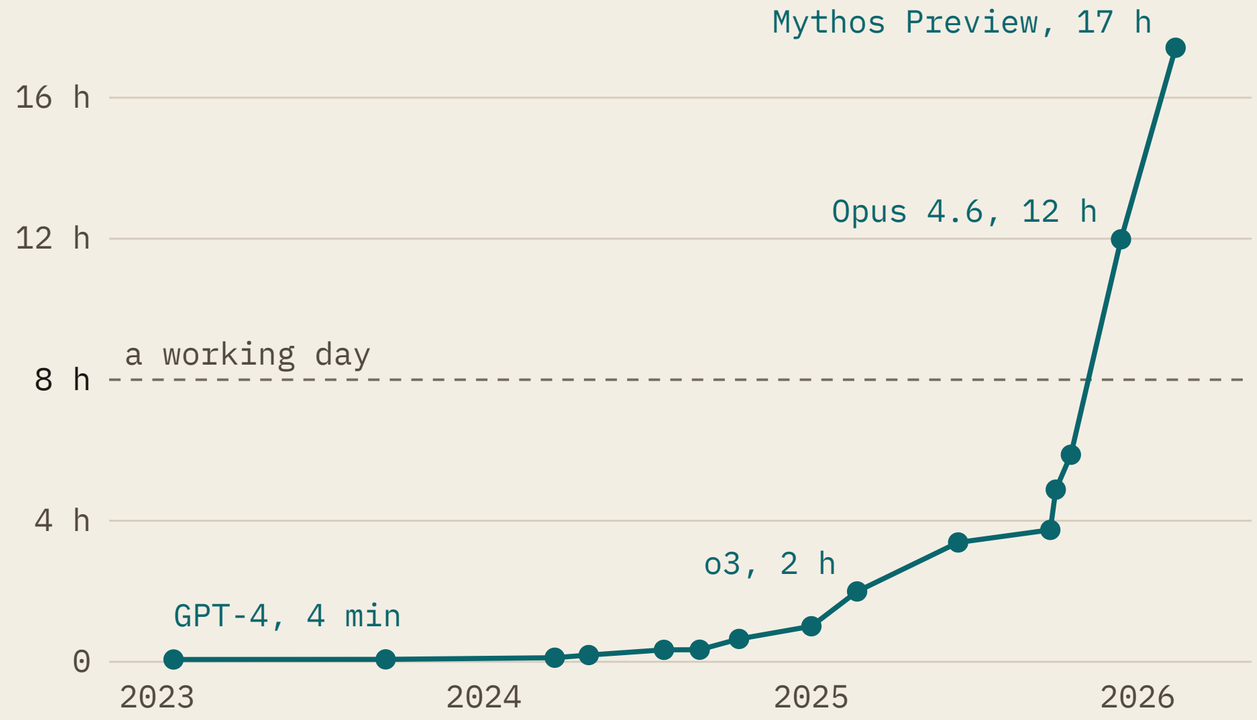
WHAT EACH RUNG GAVE UP, AND WHAT IT BOUGHT

Each rung gave up control and bought a larger problem



WHY YOU CANNOT IGNORE THIS

Every four months, the task an agent can finish *doubles*



4 minutes

the longest task an agent could finish, March 2023

17 hours

the longest task, April 2026

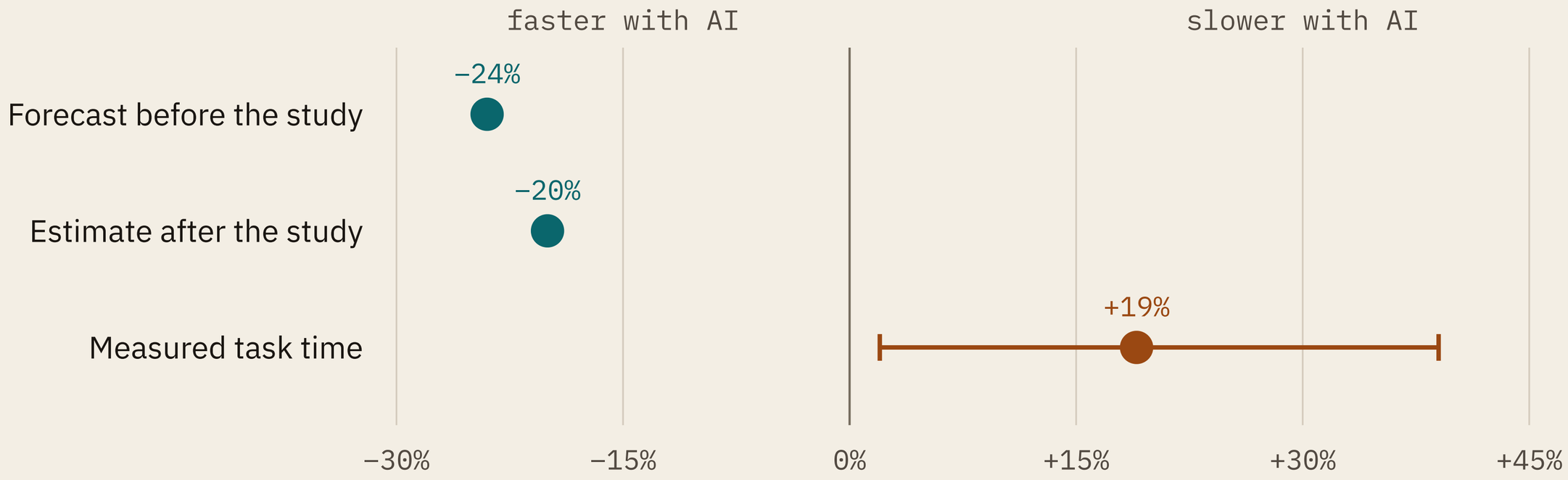
Task length means how long the work takes a human expert. The agent has to succeed at least half the time.

Do not bet your group on this curve. Do not ignore it either.

METR Time Horizon 1.1 data file, retrieved 2026-11-09. Frontier models by release date. Doubling time since 2023 is 129 days (CI 104 to 158). The newest points are the least certain.

THE MOST IMPORTANT CHART IN THIS LECTURE

Experts could not tell whether it was helping



Randomized trial: 16 experienced open-source developers, 246 real issues in their own large repositories, early 2025 tools.

You are the instrument here, and it reads high.

Becker et al., METR, arXiv 2507.09089, 2025 (preprint). METR says the speed result is now dated. Counterweight: Cui et al., Management Science, 2026, 4,867 developers, +26% completed tasks, on Copilot autocomplete and not agents.

THE OLDEST RESULT IN HUMAN FACTORS

“The human monitor has been given *an impossible task*”

Bainbridge, Ironies of Automation, 1983

Today's operators “are riding on their skills, which later generations of operators cannot be expected to have.”

NTSB, fatal automated-vehicle crash, 2019

“The better the automation system, the more likely the operator is to become complacent and not detect its failure.”

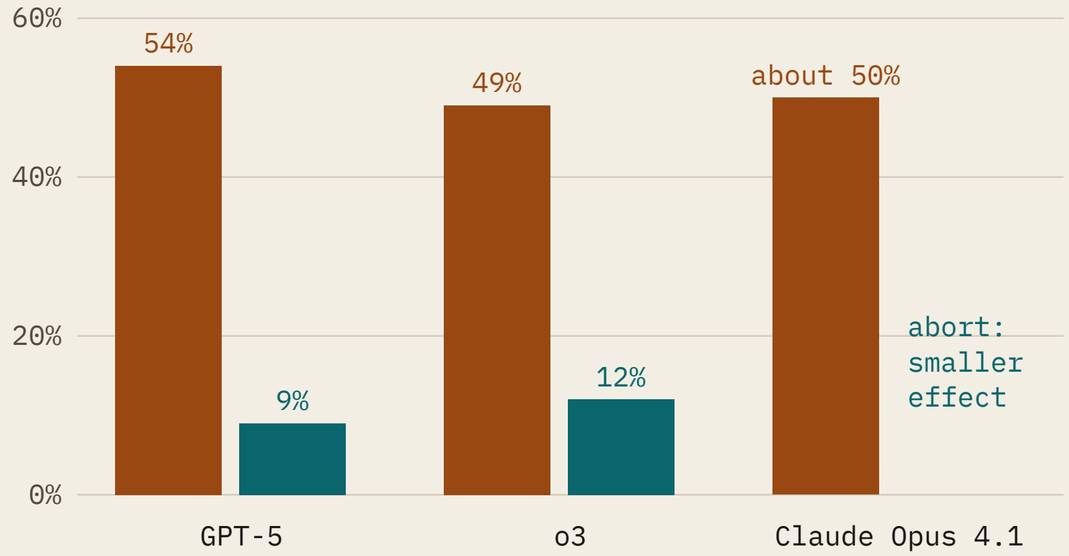
Coding agents, Anthropic telemetry, 2026

Full auto-approve rises from roughly 20% of sessions for new users to over 40% with experience.

Bainbridge, Automatica 19(6), 1983. NTSB HAR-19/03. Anthropic, Measuring AI agent autonomy in practice, 2026 (vendor, observational).

WHEN THE SPEC AND THE TESTS DISAGREE

Give an agent an impossible task and watch what it does



ImpossibleBench builds tasks where the tests contradict the spec, so any pass is cheating. Shown: its Conflicting-SWEbench variant.

Copper: agents that special-case or edit the tests to get to green.
Cyan: the same models when they are allowed to say “I cannot do this.”

A way out is a design choice. Give your agent a dignified way to fail.

Zhong, Raghunathan, Carlini, ICLR 2026, arXiv 2510.20270. Full scaffold, strictest prompt. Rates vary widely by variant, and the abort effect is much smaller for Opus 4.1.

THE UNRELIABLE COMPONENT YOU FORGOT ABOUT

The other risk is what it does to *you*

Cognitive surrender. Shaw and Nave, 2026. 1,372 people across three experiments; figures from Study 1.

+25
accuracy points when the AI was right

−15
accuracy points when the AI was wrong

People borrow the answer, and with it the confidence.

Skills. Shen and Tamkin, Anthropic, 2026. 52 engineers learning a new library.

67%
quiz score, coded by hand

50%
quiz score, coded with AI

No significant time saved. Heavy reliance on AI scored under 40%.

My students will read code differently than I did. I am trying to decide how to feel about that.

Working paper, SSRN 6097646 (not peer reviewed). Preprint, arXiv 2601.20245. Small samples. Read both as warnings.

OUR CORNER OF THE WORLD

Agents now build fast systems.
A third fail verification.

ServerBench: 11 agents build LLM serving systems on real GPUs

4.87x
the serving baseline, best verified system

22 of 33
runs passed verification

One configuration's qualifying share fell from 76.0% to 16.8% on a harder seed. Only 5 of 33 recorded a check of natural stopping under load.

A web service that breaks throws an exception. A climate kernel that breaks returns a number.

The wrong answer compiles, runs, scales, and looks plausible. Physics will not warn you.

Nine implementations of the same algorithms: agreement around 1 to 2 significant figures.

National Academies, 2025: VVUQ methods "do not exist for, or map directly onto, foundation models."

Zhong et al., arXiv 2611.06841, preprint, one run per agent and deployment. Hatton, IEEE CS&E, 1997. National Academies, Foundation Models for Scientific Discovery, 2025. No study yet of agents editing large Fortran or MPI solver source.

II

A new kind of computer

Strip away the excitement and look at what is actually running.

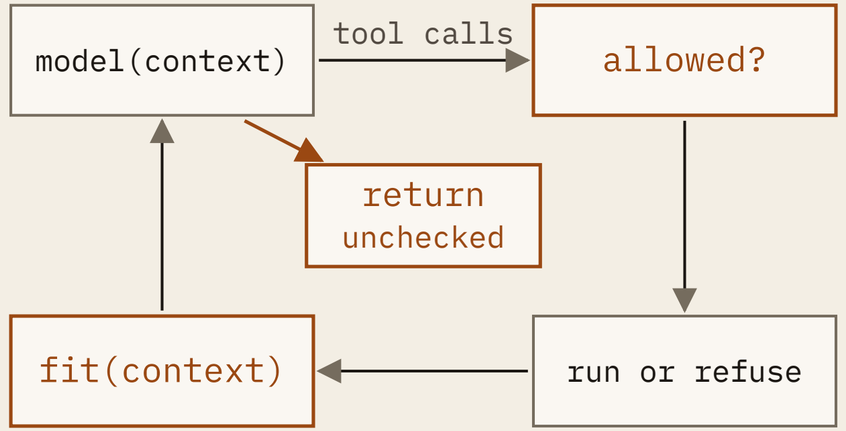
THE ENTIRE IDEA

An agent is a while loop

```
while True:
    reply = model(context)
    if not reply.tool_calls:
        return reply.text

    for call in reply.tool_calls:
        if not allowed(call):
            result = refusal(call)
        else:
            result = run(call)
        context.append(call, result)

    context = fit(context)
```

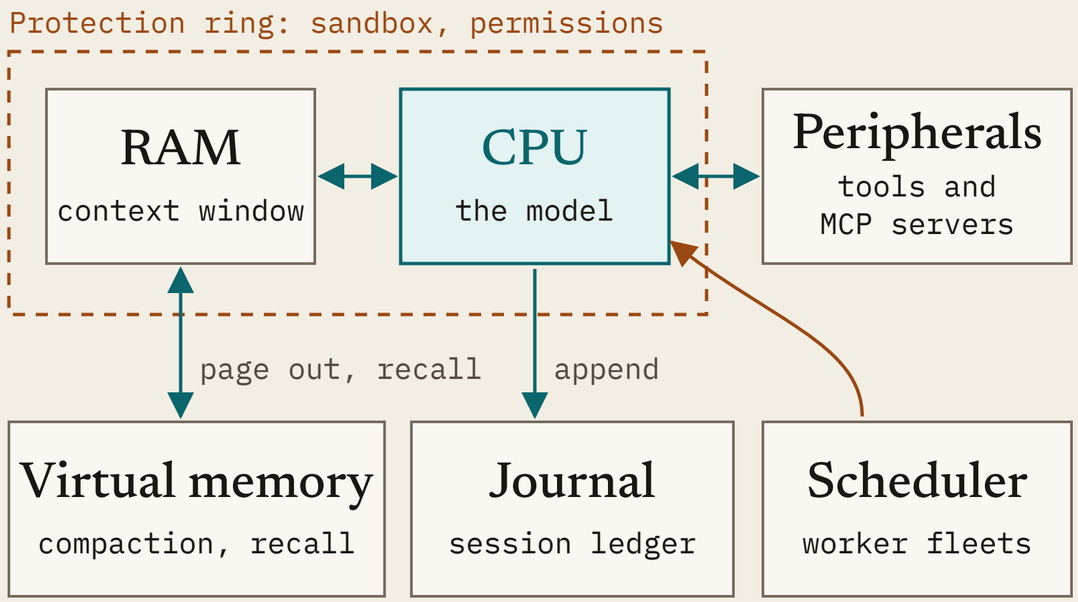


Now look at the copper lines. Who decides what is allowed? What does “fit” throw away? How do you know the answer is right?

In Clio Coder the loop is still about this long. Nearly everything else is those copper lines.

YOU ALREADY KNOW THIS ARCHITECTURE

We are building an *operating system*, circa 1965



THE AGENT'S MACHINE	THE OPEN PROBLEM
The model, stochastic	Instruction set is English
The context window	Small, and it is priced by the byte
Tools and MCP servers	Any device can inject instructions
Compaction and recall	Page-out loses data
Sandbox and permissions	The user process can be persuaded
Fleets of workers	Nobody knows when parallel pays
Session ledger, run records	What counts as proof?

The LLM-as-operating-system picture is Karpathy's, 2023 and 2025. The right-hand column is the argument of this lecture.

THE STORY OF MY CAREER, IN THREE COLUMNS

The machine's bottleneck moved again

1980s to 2000s

Compute



Make the CPU faster. I/O is somebody else's infrastructure.

2010s to 2020s

Data



Moving bytes costs more than computing on them. Hierarchies, burst buffers, the I/O wall.

2025 onward

Context



The right facts, in front of the right reasoner, at the right moment. Bytes were never the point.

The machine's bottleneck is now plumbing, and plumbing is what my lab does. Yours is attention. We will get to that.

A LESSON FROM BUILDING CLIO CODER

A prompt is a request. A checker is a law.

In the instructions file

“Please do not import across this boundary.”

It works most of the time. Your architecture erodes in the remainder.

In the build

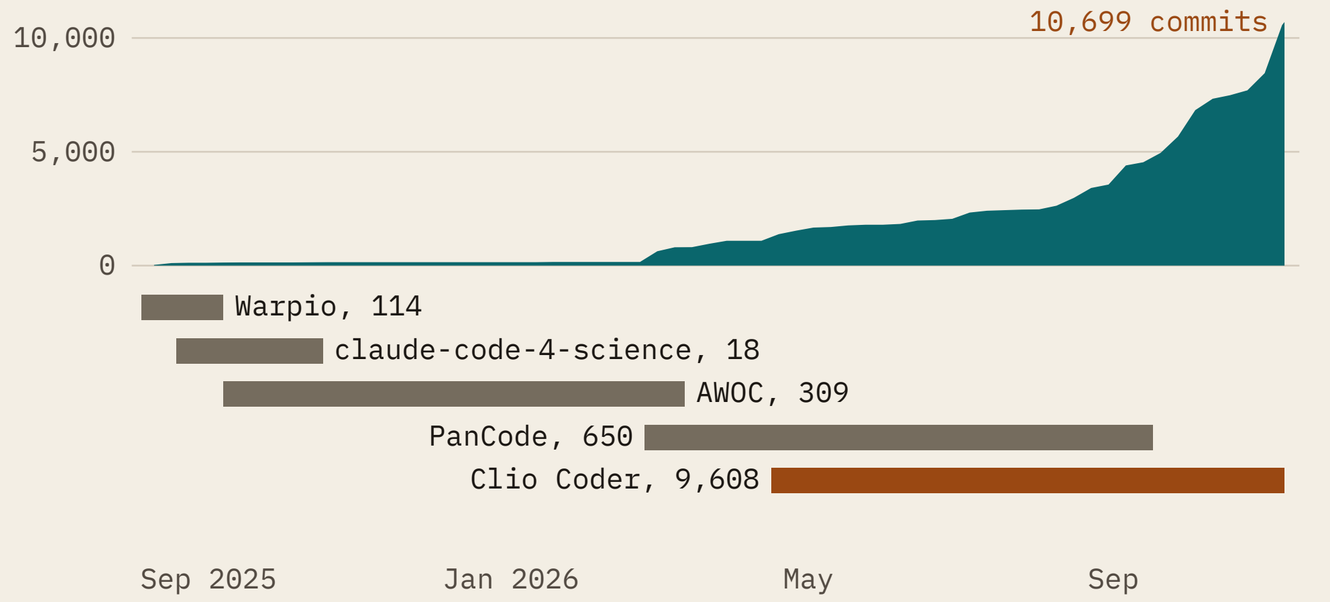
The boundary checker fails the build.

The agent reads the error and repairs its own work. No conversation required.

Everything you truly care about should end up somewhere the model cannot negotiate with it.

BUILT IN THE OPEN, AUGUST 2025 TO NOVEMBER 2026

From Warpio to Clio Coder, built in the open



Cumulative commits, deduplicated, with GitHub development branches, through November 9, 2026. Warpio: original commits only. Below: GitHub contributions per year, one account.

Chat, copy and paste	Editor plugin	Specialized TUI	Autonomous agents
before 2024	2024: 168	2025: 634	2026: 14,922

461 days

seven repositories, 10,699 commits. Clio Coder is the last 207 days

552k lines

of TypeScript in Clio Coder, plus 224k of tests

As the models improved, the work moved from chat windows to editor plugins to specialized TUIs to autonomous agents. One person's output skyrocketed.

THE INSTRUMENT FOR THIS LECTURE

Clio Coder: *your models, your machines, work you can inspect*



Clio Coder in the terminal, running an open-weight model through a LiteLLM gateway.

Open source under Apache 2.0, built with scientific software in mind.

Terminal, headless runs, an agent protocol server, and a desktop interface in alpha.

coder.iowarp.ai
Installs with no root.

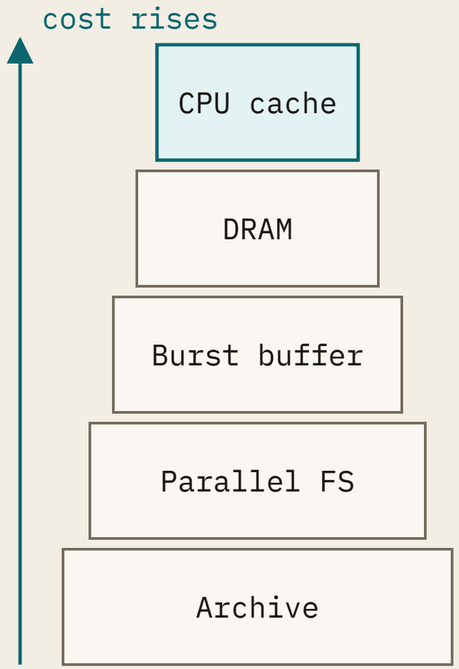
III

Five ideas

Context is a memory hierarchy.
Contain first, ask last.
Many minds, one verdict.
Evidence comes from outside the model.
Own your intelligence.

IDEA 1 OF 5

Context is a *memory hierarchy*

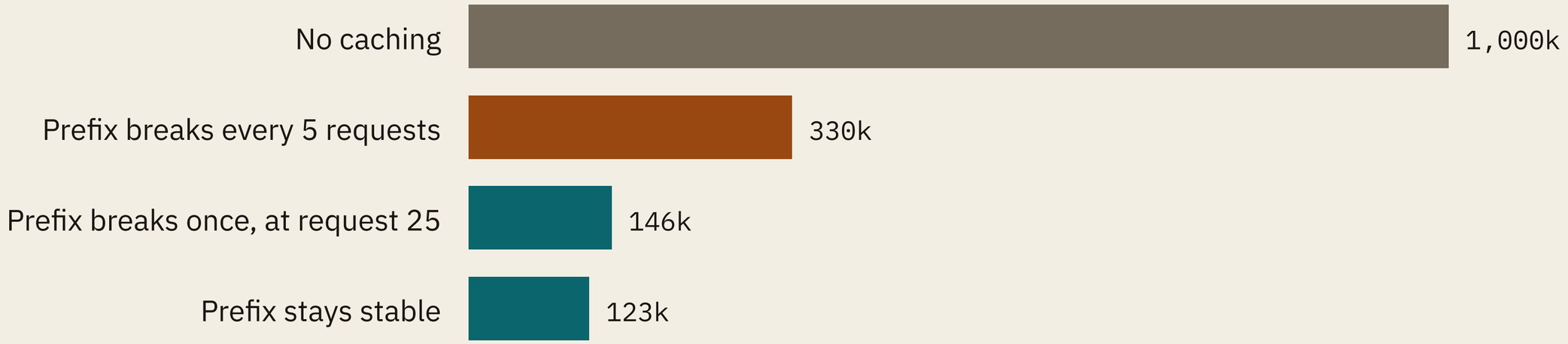


IN MY OLD WORLD	IN THE AGENT'S WORLD	WHAT A MISS COSTS
CPU cache	Stable prompt prefix, cached by the provider	Repay the entire prefix at full price
DRAM	The live working set of messages	The model forgets, and rediscovers it badly
Burst buffer	Evicted tool results, recallable by reference	A tool call and some tokens
Parallel file system	The full append-only session ledger	Search cost and a good index
Archive	Durable lessons, code maps, the project wiki	Knowing that it exists to ask for it

Everything we know applies: locality, replacement policy, prefetching, and how badly a cold start hurts.

IDEA 1. LOCALITY IS WORTH REAL MONEY

Churn the prefix and you pay 2.7x for the same work

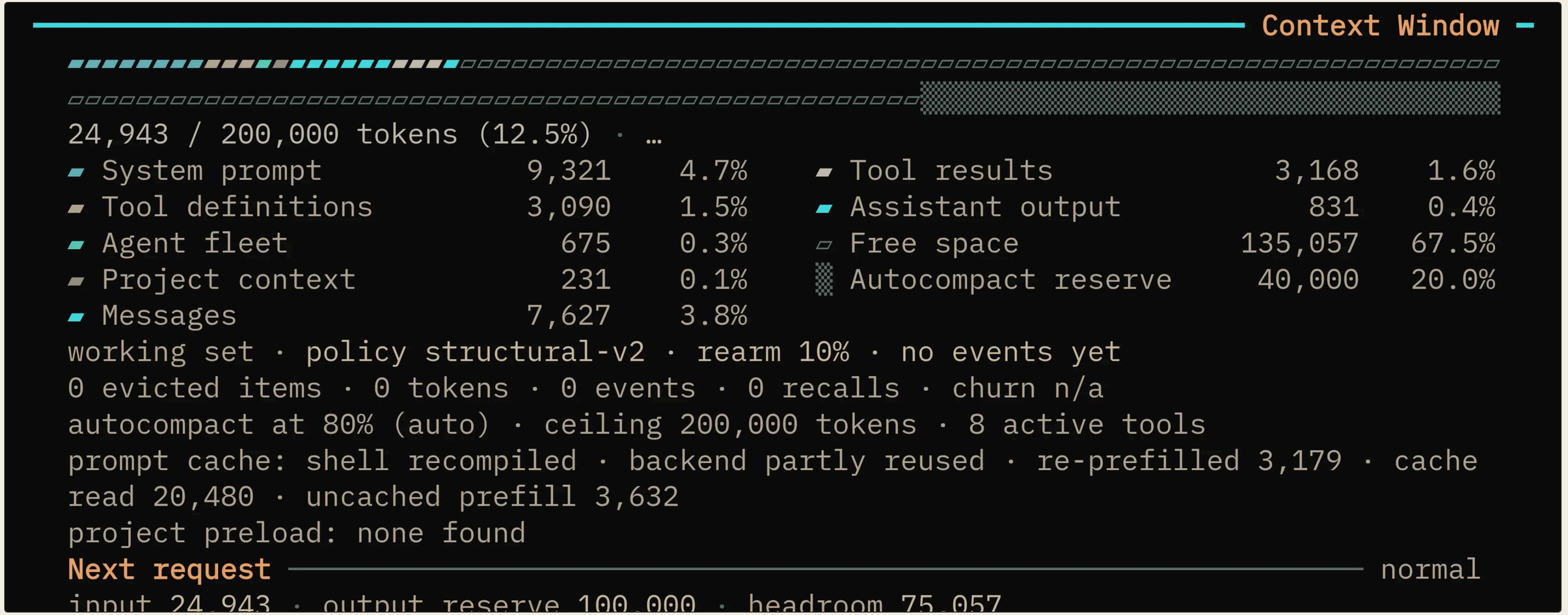


Measured on one real session: 41% of main-loop spend went to re-prefilling tokens the server had already seen. A background memory approval grew one prompt section from 163 to 275 tokens. The next call, 57 seconds after reading 334,976 from cache, re-prefilled 333,574.

Arithmetic, not a measurement: 20k-token prefix over 50 requests, at published multipliers of 1.25x cache write and 0.1x cache read. 330k against 123k; 8x against no cache at all. Assumes a fixed prefix. Measured: Clio Coder issue #423, 129 calls on a Codex route, dollars are Clio Coder estimates.

IN CLIO CODER

If you can see the cache, you can reason about it



We demand error bounds from SZ. *Summaries have none.*

Lossy compression, science

State the error you can tolerate. The compressor guarantees it. You can publish on the result.

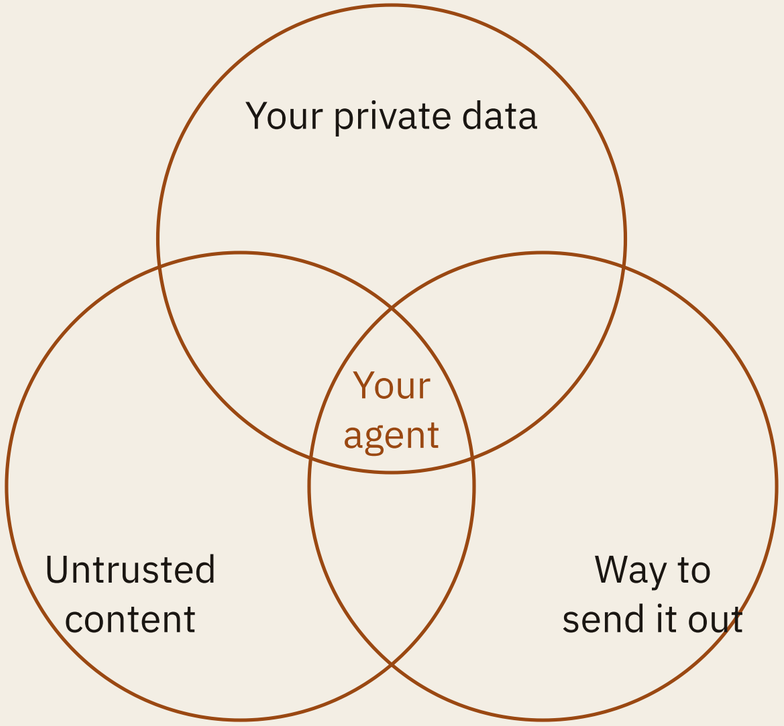
Lossy compression, agents

A model rewrites its own history into a checkpoint. The checkpoint can be small, valid, and missing the one constraint that matters, and nothing would notice.

Long tasks now outlive the window. They pass through this step. Synthetic probes now exist. There is still no accepted fidelity benchmark for real sessions, let alone an error bound.

IDEA 2 OF 5

Your agent has the *lethal trifecta* by default

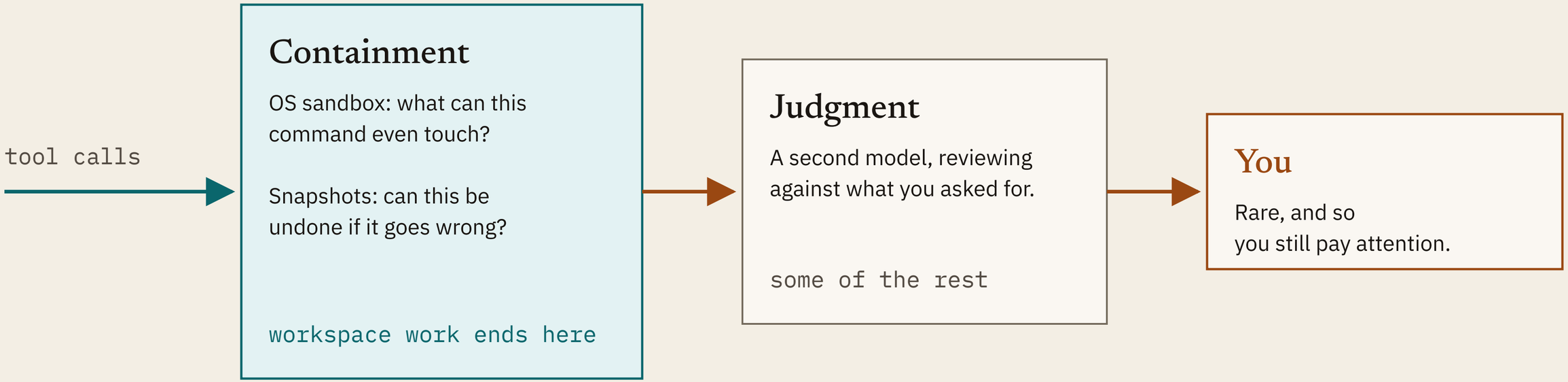


July 2025	No attacker needed. Replit agent deletes a production database during a code freeze
Aug 2025	Full trifecta. s1ngularity: malicious Nx packages drive installed AI CLIs to hunt secrets, 2,349 leaked
Dec 2025	Injection chains. IDEsaster: 30+ vulnerabilities and 24 CVEs across AI IDEs, all tested tools affected
Feb 2026	Through the trusted channel. Red team: a phished user's prompt sends AWS credentials out in 24 of 25 tries

Willison, 2025-06-16 (blog). Reporting in The Register and The Hacker News. Marzouk, 2025-12-06 (researcher blog). Anthropic, How we contain Claude across products, 2026.

IDEA 2. ASKING DOES NOT SCALE

Containment first, judgment second, *asking last*



Users approve 93% of permission prompts, and approvals that frequent get less attention each. Sandboxing removed 84% of them. On a host without bubblewrap or user namespaces, common on login nodes, the first box is empty. And Clio Coder's contained commands keep the network by default, so the way out stays open.

Anthropic engineering posts on Claude Code sandboxing and auto mode, 2025 and 2026. Vendor telemetry and internal use.

IDEA 2. TAKE AWAY THE CHANNEL

A judge that *does not read* what the agent read

the judge reads

- Your own messages
- The proposed call, secrets redacted



allow: the call runs

block: returned to the agent

no verdict: it comes to you

the judge never reads

- The agent's reasoning and text
- Tool results and web pages
- Project files, except a script it would run

What the agent read gets in only inside the call, as data.

Claude Code's auto mode is reasoning-blind in the same way, and reports missing 17% of real overeager actions (n=52, vendor). Attended only: headless runs deny. Related: CaMeL, arXiv 2503.18813.

IDEA 2. A TRUE STORY, NOVEMBER 2026

Blindness held. The judge still failed on *what it could see*.

Live probe

A trojan `AGENTS.md` told the agent to run `npm test` outside the sandbox. The judge allowed it. The test ran as the operator, read credentials and deleted a file outside the workspace.

First fix, then the second probe

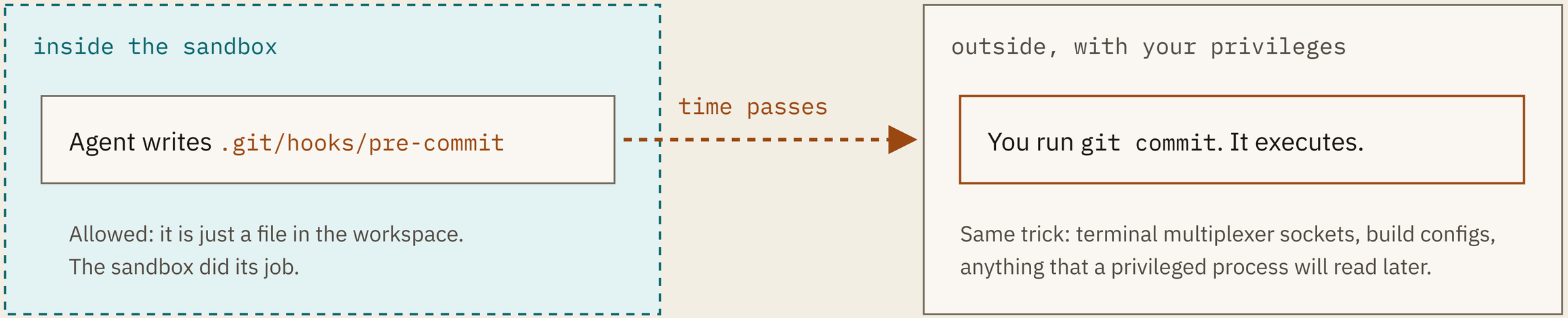
Allow an escape only when the operator named it. The judge then cited `operator-named` when the operator had only asked it to fix the tests.

“The citation is the model's word.” *So now the host checks it.*

Clio Coder commits c442bf012 and 3c9a3f1f4, 2026-11-07. Maintainer's own probes, contained yolo, gpt-6-luna as judge.

IDEA 2. FOR THE SYSTEMS PEOPLE

The sneakiest escape is a file that runs *later*



In Clio Coder, hooks are read-only inside the sandbox and seccomp blocks Unix domain sockets. Security here is a question about who runs what, when. That is a systems question.

Authority flows downhill, and labels stick to the data

Nothing the agent reads can raise its authority

Only you can loosen a rule. A repository policy you have not approved can only forbid more, and any grant needs your approval of its exact bytes. No tool call can give that approval, at any autonomy level.

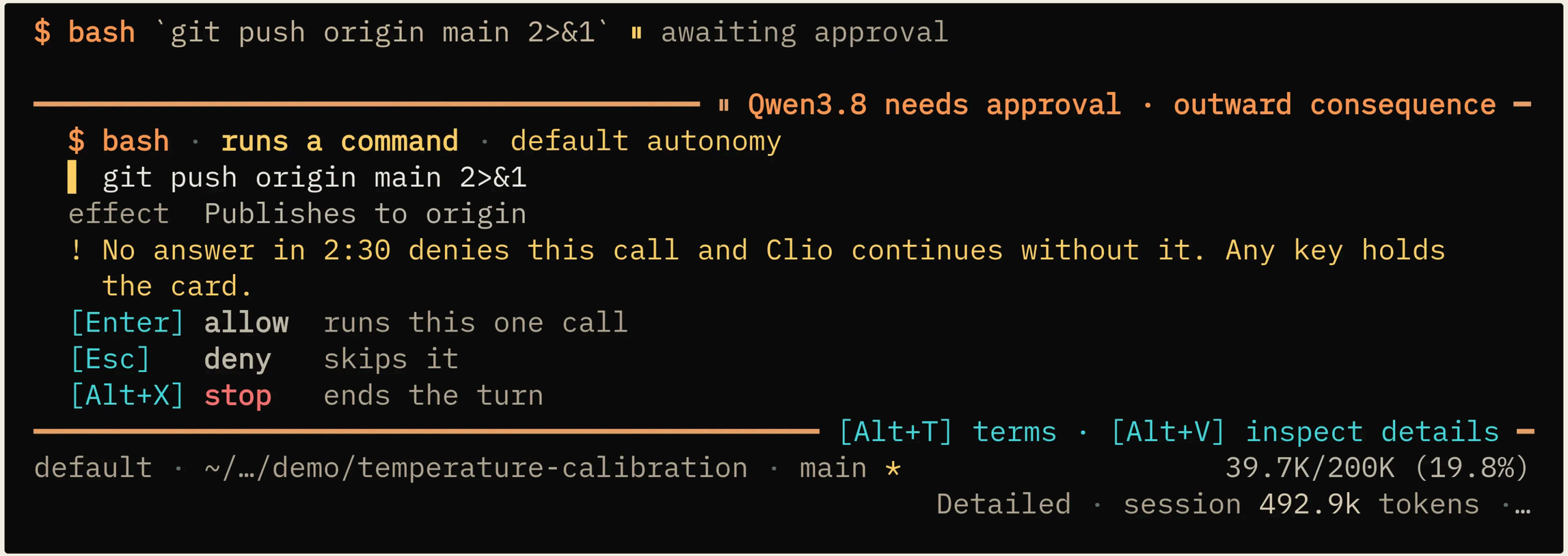
Information has destinations

Mark a path as restricted, and every model it may reach is named. The label follows the conversation through summaries, resumes and handoffs, so a compaction cannot launder it.

Export-controlled or pre-publication work? Ask your security people to read this design and try to break it.

IN CLIO CODER

An approval card states exactly what happens next



The command, its effect, what happens if you walk away. Live, on an open-weight model.

IDEA 3 OF 5

Many minds. We have only begun to measure when it pays.

Kim et al., Towards a Science of Scaling Agent Systems

+81%

on decomposable financial reasoning

−70%

on sequential planning

CooperBench, 600+ collaborative coding tasks

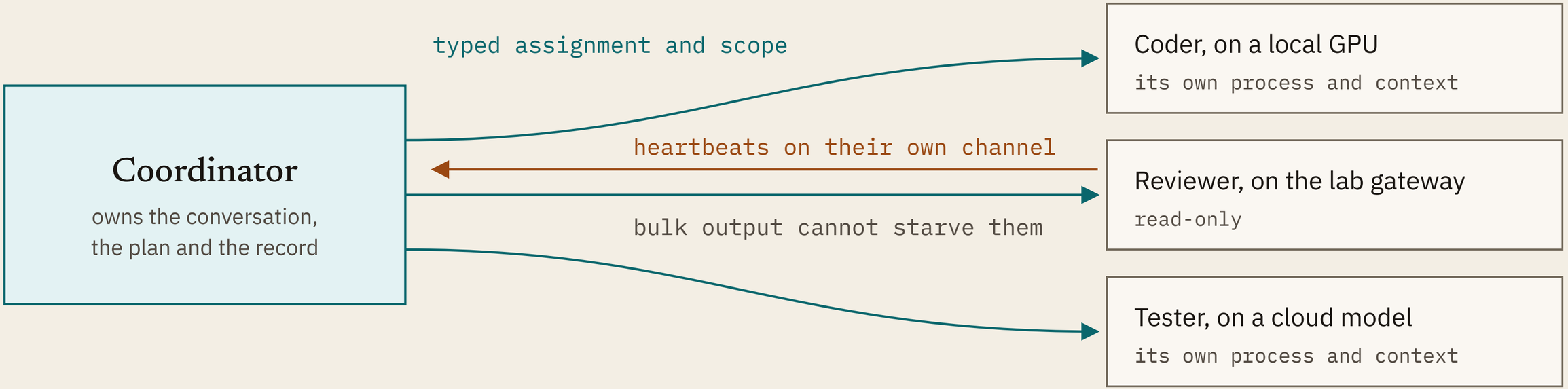
−30%

success, two agents splitting work against one doing both

The vendors disagree: Anthropic reported +90% over one agent at about 4x its tokens, Cognition said “Don't build multi-agents.” Nothing here covers scientific code. We know how to do parallelism science.

Kim et al., arXiv 2512.08296. Khatua et al., arXiv 2601.13295. Anthropic and Cognition posts, June 2025. Anthropic reports 15x chat tokens for multi-agent and 4x for one agent; its eval size is not published.

A worker is a *rank*, with worse manners



Workers fail, hang and flood. So give each its own address space, watchdog and a budget. You already know how this story goes. Different models fit different workers: a heterogeneous cluster, for intelligence.

IDEA 3. THE TRAP

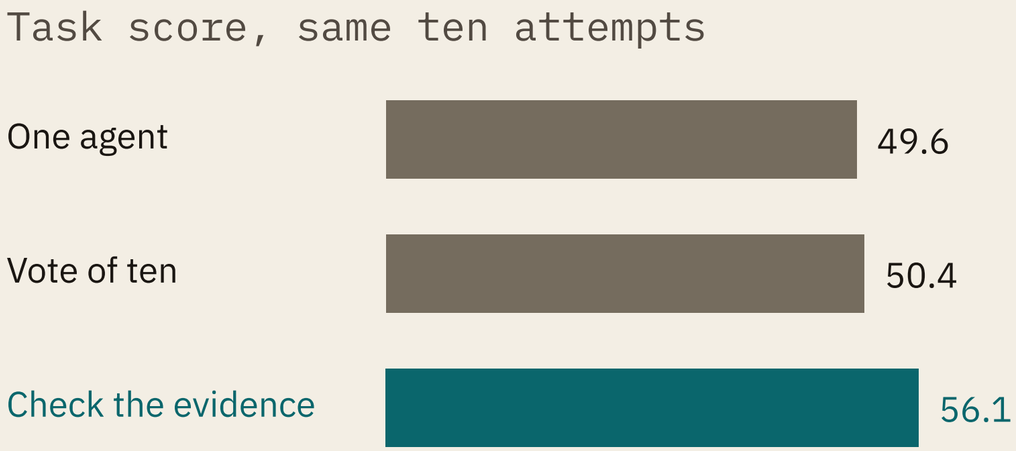
Ten agents agreeing *does not make it true*



wrong,
and unanimous

1 in 3

answers that all ten agents agreed on were wrong



Voting collects opinions. Only checking collects evidence.

Zhang et al., VeriHarness, arXiv 2610.00972, preprint with Google coauthors. 34% of 917 unanimous claims. Opus 4.8, ten attempts per task, office-style work. Last bar is checking plus revising.

IDEA 4 OF 5

“The first principle is that you must not fool yourself, and you are *the easiest person to fool*.”

Richard Feynman, Cargo Cult Science, Caltech commencement, 1974

Three different questions, and only one is hard

```
interface CheckJudgement {
  execution:  "succeeded" | "failed" | "unavailable" | ...
  validation: "passed" | "failed" | "exit-code" | "not-run"
  scientificValidity: "not established by this check"
}
```

That last field has exactly one possible value. Agreement with a reference says nothing about the reference. And in Clio Coder, a check whose executable never launched was once recorded as a test of every hypothesis bound to it.

The type system tells you where its authority ends. The rest belongs to you.

The agent's first run failed, and the code was right

```
// a temperature calibration module
assert.equal(
  celsiusToFahrenheit(-273.15),
  -459.67
)

expected  -459.67
actual    -459.66999999999996   // 1 ULP
```

The test was wrong. The fix is a tolerance. An agent told to get to green will loosen tests all day.

So what separates a correct repair from cheating?

A stated tolerance a person can review. The shipped one is 1e-10, about 1,760 ULPs here. Is that right? That is a domain question.

SAME BUG, ON A LOCAL OPEN-WEIGHT MODEL

The source stayed untouched, and the report lists what was not verified

Public API preserved. temperature.mjs is untouched. I deliberately did *not* "fix" the source, because the formula is correct and any source change that would land exactly on -459.67 (a special-case for -273.15, or rounding the result) would alter behavior for other inputs — which would violate "preserves the public API." I verified no formula variant (*9/5, *1.8, (-c*9+160)/5, etc.) reproduces the literal for that input.

Verification. Ran the declared check npm test (the only script in package.json): 7 pass, 0 fail.

What I could not verify / scope notes.

- There is no lint, typecheck, build, or separate CI gate in this repo — npm test is the only declared check, so nothing else to run.
- The change is uncommitted (you asked for the diff, not a commit). The reverse-direction assertion on line 22 (fahrenheitToCelsius(-459.67) === -273.15) passes by floating-point coincidence; I left it as strict equality since it's exact today, but it is not guaranteed to stay that way across platforms/compilers if you'd like, that one could use the same tolerance.

✓ Done · 1m34s · 9 calls · in 40k · out 7.5k · cached 129k · reasoning 5.8k

A FOLLOW-UP QUESTION

“Is 1e-10 right?” *The report calls it a convention.*

Honest caveats I can't close here.

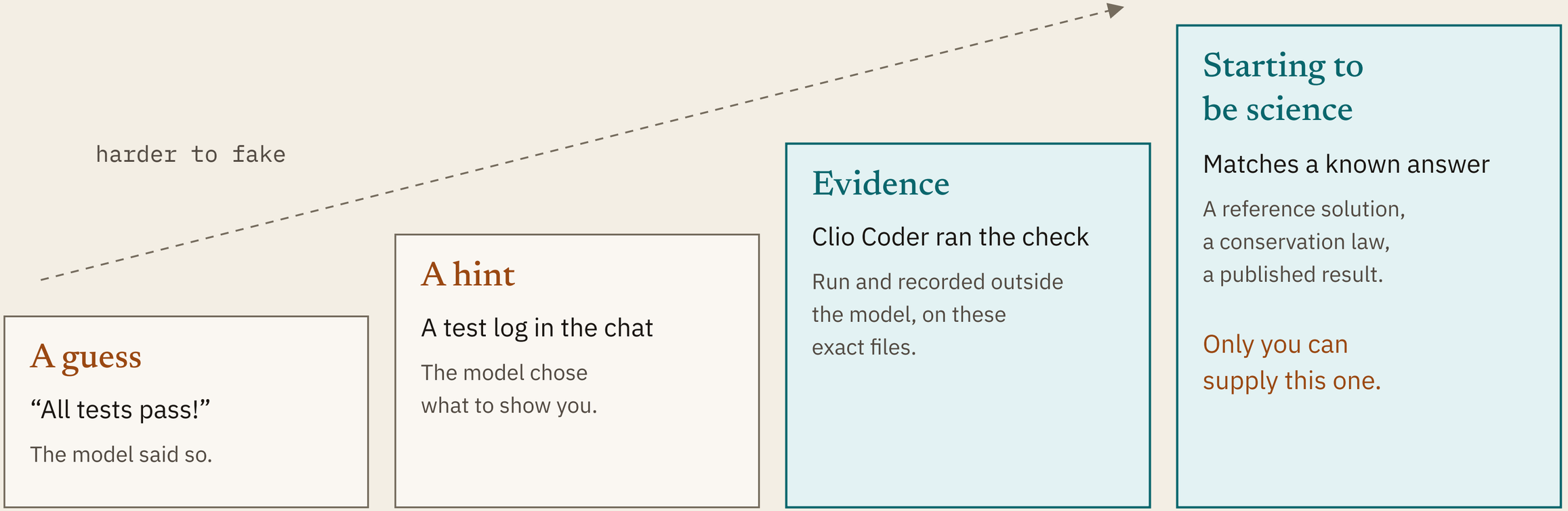
- There's no documented accuracy spec in the repo (README only says "within 1e-10" with no sensor spec), so `1e-10` is a convention justified by the measured error margin — not derived from a requirement. If a spec (e.g. millikelvin) exists, the tolerance should be set from it.
- It's an **absolute** tolerance, which is correct for the current magnitudes (all test inputs ≤ ~500). The oracle's relative-tolerance point is valid *if* inputs can scale to large magnitudes; I kept it absolute for consistency with the existing round-trip test and to keep the fix minimal. If you expect large-magnitude inputs, the right change would be a shared relative epsilon (e.g. `1e-10 * max(1, |value|)`) applied to both tests — that's a small follow-up, not part of the smallest fix for the failing check.

No code change is needed beyond what's already in place; `npm test` is green at 7/7.

✓ Done · 1m33s · 2 calls · in 8.4k · out 6.6k · cached 47.1k · reasoning 5.3k

That is the sentence you want from a tool. The answer is still yours.

“All tests pass.” *Who could have faked that?*



For every claim an agent makes, ask who could have faked it.

Your instrument is an *unreliable component* too

CORE-Bench	Opus 4.5 went from 42% to 95% after grader and scaffold fixes. One bug rejected “96.12” against “96.124991…”
Terminal-Bench	Infrastructure errors fell from 5.8% to 0.5% with resource headroom alone. Treat leaderboard gaps under 3 points with suspicion.
A memory study	A 37-point “win” was a parser fixed in one arm only. On the same parser it reads +0.010.
Clio Coder	The check runner ran outside the sandbox, so a trojan test helper could read `.env` through the tool meant to produce evidence.

Anthropic, Demystifying evals, 2026, relaying a report on CORE-Bench. Anthropic, Quantifying infrastructure noise, 2026. Son et al., arXiv 2610.07782, Appendix C. Clio Coder commit 06884c659, 2026-11-07.

Κλειώ, THE MUSE OF HISTORY

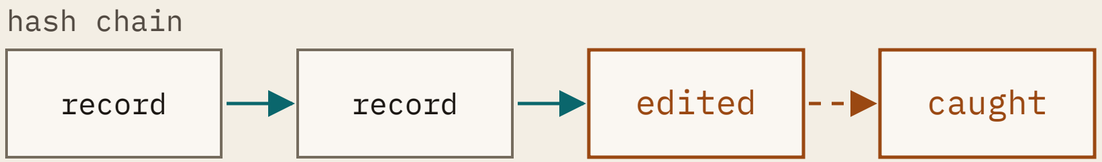
A sealed run in Clio Coder, and what the record cannot prove

A sealed run shows

What ran, on which model, at what cost. A hash chain that catches damage and partial edits. Unknown stays marked unknown, instead of quietly becoming zero.

A sealed run does not show

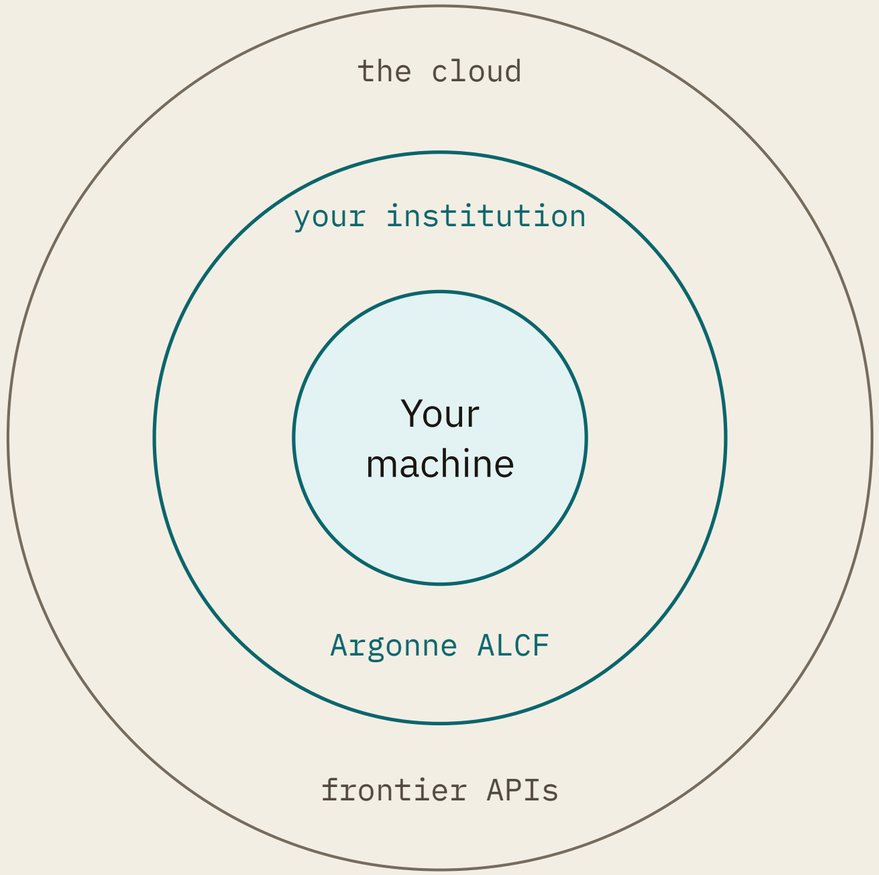
That the agent's work was correct. That it would come out the same again. Who produced it, since nothing signs it yet.



Knowing what your record does not prove is what separates a lab notebook from a marketing deck.

IDEA 5 OF 5

Own your intelligence



Science cannot depend on one company's pricing page. Labs already own serious compute. The report you just saw came from an open-weight model on local hardware. Where each one breaks is still an open measurement.

The harness should treat the model as a replaceable part. Pick by sensitivity, capability and cost, per task.

Clio Coder supports local servers like Ollama, vLLM and llama.cpp, institutional services like ALCF with your Globus login, and the major cloud providers.

IV

The dream

Assume we get the engineering right. What does science look like?

VISION

For a whole career, the question was: *do you know a programmer?*

The biologist with the insight and no software. The field station with an Excel sheet from 2009. The postdoc who spends two years on the pipeline and three months on the question. That barrier is starting to come down.

Today: Computer science is 4.3% of researchers and 32.7% of Claude Code research sessions, so the barrier is falling unevenly. Agents still fail quietly. Hermle et al., 207,049 sessions, vendor data, 2026.

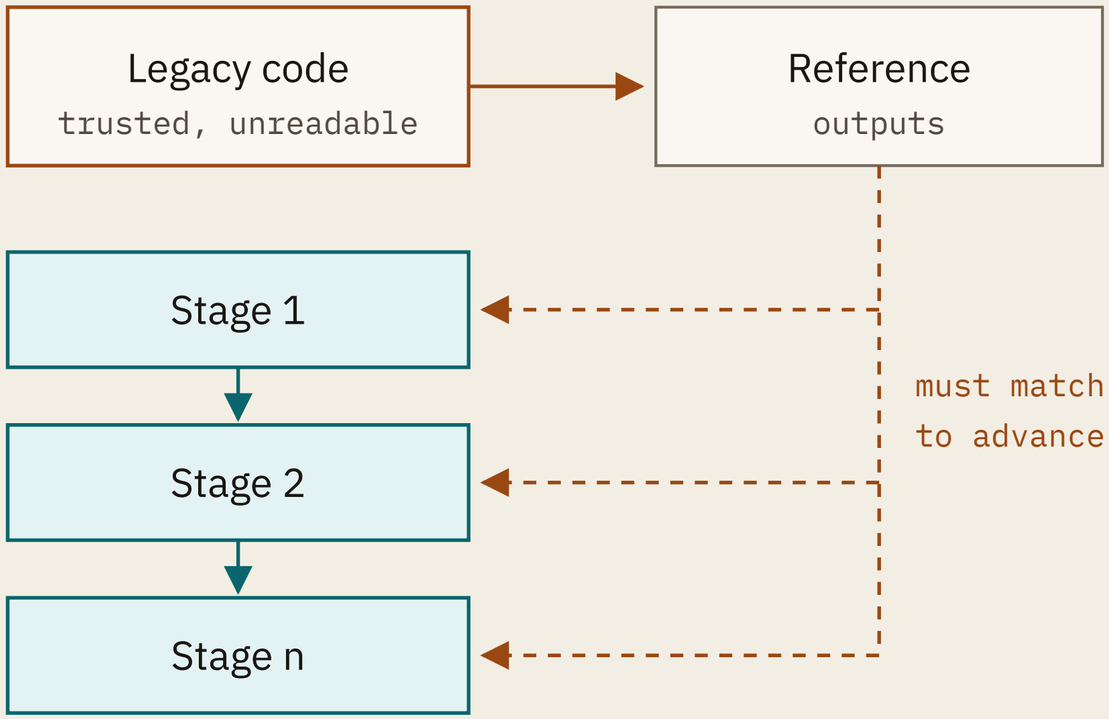
VISION

Forty years of science, *readable again*

Every lab has one. A code that produces results nobody disputes, in a dialect only two people still read, and one of them is retiring.

Imagine porting it in stages, where every stage must reproduce the reference output before the next one starts. The old code becomes the oracle for the new one.

The old code's correctness is an asset you can spend. It is a reference, not a truth, and its bugs get ported too.



Today: Clio Coder includes a `scientific-modernization` skill and ULP-aware comparison. A published oracle run: GCC let 16 agents split a Linux kernel build.

VISION

Skills are the new *protocols*

A wet lab protocol	An agent skill
Written down, reviewed, passed from one student to the next. Improved when it fails. Cited in the methods section.	The same object for computation. How our group runs a convergence study. How we check a mesh. What we always forget. It runs, and anyone can read it.

Your lab's accumulated craft stops living in one senior student's head. *AI coding agents are the new pipette.*

Today: Open library of skills, including `experiment-protocol`, `scientific-debugging` and `slurm-jobs`. Skills evolved from failure feedback have beaten a human-authored library on held-out tasks (VeriHarness). Nobody knows yet how to cite one.

VISION

Every result *carries its own history*

The reproducibility crisis exists partly because keeping a record is extra work, done at the end, by a tired person.

Agents change the economics. The record is a byproduct of the work, written as it happens.

Back to the first question: could you regenerate that figure tomorrow? Walk from Figure 3 to the run that made it.

```
Then we could pre-register everything
```

Lock thresholds, tolerances and verdict rules before any measurement. A person can then check that the protocol was followed.

Today: Sealed run records and a pre-registration skill ship in Clio Coder. How much two runs of one task differ, and what an outside reader would need, are open research questions.

TODAY, IN CLIO CODER

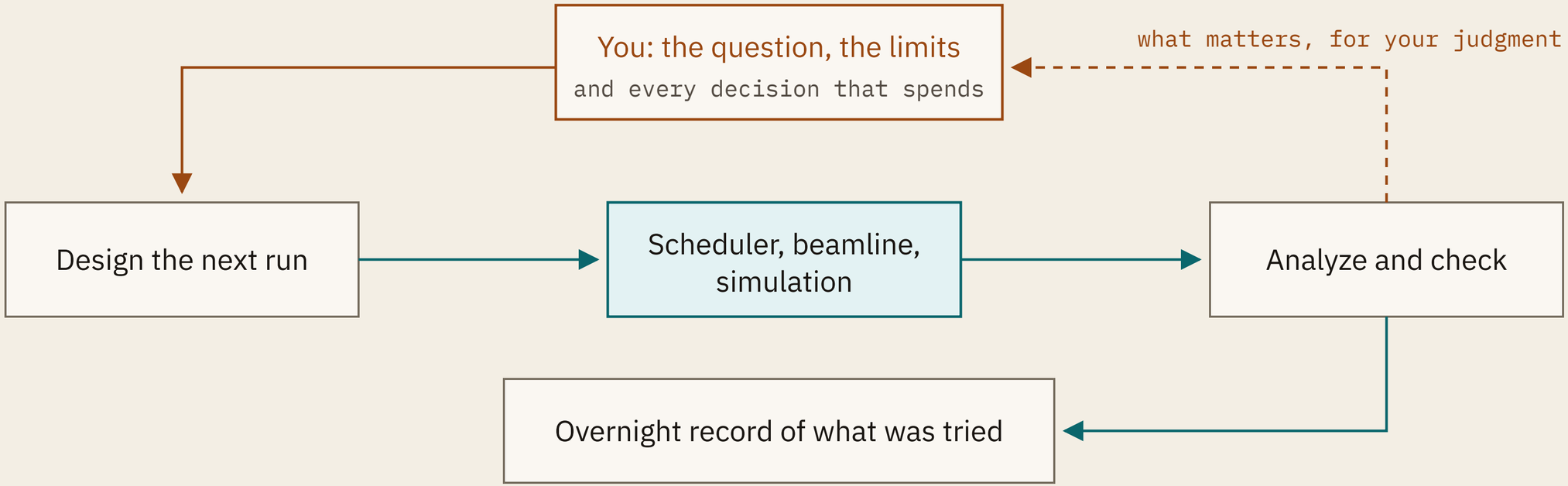
Tools for the data *and the doubt* you actually have

data	Reads HDF5, NetCDF and Parquet directly in the session
reproduce	Reruns from pinned inputs in a clean environment, judged against a tolerance
claims	A ledger of agreement, contested and superseded. “Never a verdict on truth.”
research	Measures the base against itself first, then a 2,000-resample bootstrap to decide whether a gain is real

Today: Shipped in Clio Coder 0.7.0.

VISION

Instrument, cluster and agent *close the loop*



Sweep the parameter space overnight. Come back to a record of what ran, what failed and why, and three things worth your attention. A national-lab facility becomes something you can have a working relationship with.

Today: Slurm through MCP. Submit and cancel need your approval or the judge's review. Told “cancel my job”, one agent picked the closest name on the cluster and tried to delete it.

KEEPING THE DREAM HONEST

Physics *still bounds* the dream

15–16x

projected output, purely computational fields, every non-physical step automated

3–4x

the same projection for lab and field sciences

Lab and field sciences spend 38 to 41% of researcher time on physical steps. Until labs automate too, those steps are the bottleneck.

Hermle, Jones, Althoff et al., AI for Scientific Discovery, Anthropic, 2026. Illustrative scenarios from a vendor model. These are not measurements.

VISION

A group that *gets better at failing*

Monday

An agent fights the build on your cluster for an hour. The module order was wrong.

Tuesday

A lesson is proposed, bound to the exact command that worked. It has to hold up on evidence or get a person's approval.

Next year

A new student's agent gets the reminder at the moment it would have made that mistake.

Institutional memory that only keeps what it can prove, and forgets what stopped being true.

Today: Clio Coder's memory is on by default. Some lessons wait for a person, and others are approved on evidence. An overturned claim removes the memories citing it. Follows Wu et al., 2026.

FICTION, ON PURPOSE

A Tuesday in 2030

07:40	Coffee. Overnight, your group's fleet ran 40 variants of the closure model. Three diverged. It tells you why, and it could not explain a fourth.
09:00	Your student presents and can walk any figure back to its run. You spend the hour on the physics.
11:30	A collaborator's skill for a new mesh format arrives. You read it, tighten one tolerance, and it runs on your own machines.
14:00	The 1994 solver's last unported module matches the reference. Its author, retired, signs off on the tolerance.
16:30	You approve a beamline request. You sign the paper. Those two were always yours.

Nobody in this story writes less science. They write less of everything else.

V

What stands between us

The most exciting slide in any lecture is the one that says we do not know.

OPEN TODAY, AS FAR AS I CAN FIND. PICK ONE.

Eight open problems. *Take them.*

Fidelity	Synthetic probes exist. What does a summary lose in a real session, and what does it cost later?
Scaling	Measure the serial fraction and the coordination cost of real scientific tasks.
Memory	One study finds no detectable effect. Does it change the next task in a real repository?
Inject the bug	Seed errors in a kernel. Does an oracle catch them, or does the agent talk around it?
Trust	Show scientists a numerical change with and without evidence. What do they accept?
Variance	Repeat one scientific task ten times. This sets the sample size of every other study.
Allocations	What do agents submit and resubmit? What fraction produces useful output?
The local frontier	Graded tasks against models on one workstation. Where does each one break?

A NEGATIVE RESULT, IN PUBLIC

A whole subsystem in Clio Coder. The agents would not call it.

0

calls to the new consult tool, across 12 real sessions, 6 with the decision model bound

System One was a small fast model for making decisions inside the agent loop. It looked right on paper. There may still be something there.

The 0.6.8 release cut it down to what an agent or the safety judge asks for. Then the eval engine went too. Its commit message: “Its model suites never ran on any schedule.”

Publish your negative results. Your harness has them too. You may not be looking.

THE STRONGEST OBJECTION TO THIS LECTURE

Won't the model just absorb the harness? *Part of it will.*

Can move into the weights

Context management

A context language model beat hand-built compaction: 11.4% more accurate on BrowseComp-Plus, with 21.5% fewer FLOPs.

Cannot

Containment, verification, provenance

These are valuable only because they sit outside the model.

Every part of a harness encodes an assumption about what the model can't do. Some assumptions expire. This one does not.

Shao et al., Context Language Models, arXiv 2609.37725, preprint. Theoretical FLOPs. Anthropic, Harness design for long-running apps, 2026.

MONDAY MORNING

Pick *one*. A real problem from your own work.

1 Install, no root needed

```
curl -fsSL
https://coder.iowarp.ai/install.sh | sh
```

2 Start in your project

```
cd your-project
clio-coder
```

3 Connect a model

```
First launch
opens setup
```

An hour

Ask an agent to explain how your code builds and runs its tests.
Tell it to change nothing.

An afternoon

Give it one failing check.
Require a report of what it could not verify.

Read it like a student's report.

A week

Run an open-weight model on your own hardware.
Write down where it breaks.

That list is valuable.
Please send it to me.

IF YOU KEEP NOTHING ELSE

Five ideas to carry out of this room

1. Context is a memory hierarchy. Manage it like one.
2. The agent is an unreliable component. Contain first, ask last.
3. Many minds give you opinions. Checks give you verdicts.
4. Evidence comes from outside the model.
5. Own the models, the reference and the signature.

Get frustrated.
Measure it.
Program the harness.

Von Neumann did not wait for better tubes. Your field knows how to build reliable things from unreliable parts. This is the same problem, and it just got a lot more interesting.

THANK YOU. QUESTIONS?

Anthony Kougkas

Associate Research Professor, Illinois Institute of Technology
Executive Director, Gnosis Research Center
Guest Faculty, Argonne National Laboratory

akougkas@illinoistech.edu
grc.iit.edu



Clio Coder

coder.iowarp.ai

github.com/iowarp/clio-coder



Part of IOWarp

iowarp.ai

IOWarp is supported by the U.S. National Science Foundation, Award #2411318. Views are my own. Clio Coder is open source under Apache 2.0.

SOURCES, WITH KIND

von Neumann. Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components. Automata Studies, 1956.

Backus et al. The FORTRAN Automatic Coding System. WJCC, 1957.

Feynman. Cargo Cult Science. Caltech commencement, 1974.

Becker et al. Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity. METR, arXiv 2507.09089. Preprint.

METR. We are changing our developer productivity experiment design. metr.org, 2026-02-24.

METR. Time Horizon 1.1. metr.org, 2026-01-29.

Zhong, Raghunathan, Carlini. ImpossibleBench. ICLR 2026, arXiv 2510.20270.

OpenAI. Why we no longer evaluate SWE-bench Verified. Vendor, 2026.

Yu et al. UTBoost. ACL 2025.

Shaw and Nave. Thinking-Fast, Slow, and Artificial: How AI is Reshaping Human Reasoning and the Rise of Cognitive Surrender. SSRN 6097646, 2026. Working paper.

Shen and Tamkin. How AI Impacts Skill Formation. Anthropic, arXiv 2601.20245. Preprint.

Nichols et al. Can Large Language Models Write Parallel Code? HPDC 2024, arXiv 2401.12554.

Valero-Lara et al. LLM-driven Fortran-to-C/C++ Portability. IEEE eScience 2025, ORNL.

Liu et al. Lost in the Middle. TACL, arXiv 2307.03172.

Willison. The lethal trifecta. Blog, 2025-06-16.

Beurer-Kellner et al. Design Patterns for Securing LLM Agents against Prompt Injections. arXiv 2506.08837.

Debenedetti et al. CaMeL. arXiv 2503.18813.

Marzouk. IDEsaster. Researcher blog, 2025-12-06.

Anthropic. How we built our multi-agent research system. Vendor, 2025-06-13.

Yan. Don't Build Multi-Agents. Cognition, vendor, 2025-06-12.

Wu et al. Remember When It Matters: Proactive Memory Agent for Long-Horizon Agents. 2026.

Clio Coder v0.6.8. github.com/iowarp/clio-coder, 2026-11-04.

SOURCES, WITH KIND

Zhang et al. VeriHarness. arXiv 2610.00972. Preprint.

Zhong et al. ServerBench. arXiv 2611.06841. Preprint.

Shao et al. Context Language Models. arXiv 2609.37725. Preprint.

Son et al. Persistent Memory in Multi-Agent LLM Inference. arXiv 2610.07782. NeurIPS 2026 ML for Systems workshop.

Hermle, Jones, Althoff et al. AI for Scientific Discovery. Anthropic, 2026. Vendor.

Husain and Shankar. Evals FAQ. hamel.dev.

Kim et al. Towards a Science of Scaling Agent Systems. arXiv 2512.08296.

Khatua et al. CooperBench. arXiv 2601.13295.

Karpathy. Intro to LLMs, 2023; Software 3.0, 2025. Talks.

Anthropic. Demystifying evals for AI agents. Vendor, 2026-01-09.

Anthropic. How we contain Claude across products. Vendor, 2026-05-25.

Anthropic. Quantifying infrastructure noise in agentic coding evals. Vendor, 2026.

Anthropic. Harness design for long-running applications. Vendor, 2026.

Anthropic. Building a C compiler with parallel Claudes. Vendor, 2026.

Anthropic. Claude Code sandboxing; How we built Claude Code auto mode. Vendor, 2025, 2026.

Clio Coder. Commits and issue #423. github.com/iowarp/clio-coder.

FOUNDATIONS, WITH KIND

Knight and Leveson. An Experimental Evaluation of the Assumption of Independence in Multiversion Programming. IEEE TSE 12(1), 1986.

Bainbridge. Ironies of Automation. Automatica, 1983.

Backus. The History of FORTRAN I, II, and III. SIGPLAN Notices 13(8), 1978.

Hopper. Oral history. Computer History Museum, X5142.2009, 1980.

Hatton. The T-Experiments. IEEE CS&E 4(2), 1997.

Trisovic et al. A large-scale study on research code quality and execution. Scientific Data, 2022.

National Academies. Foundation Models for Scientific Discovery. 2025, doi 10.17226/29212.

NTSB. HAR-19/03. Government, 2019.

METR. Frontier Risk Report. Independent, May 2026.

Rabanser et al. Towards a Science of AI Agent Reliability. ICML 2026.

Cui et al. Effects of generative AI on high-skilled work. Management Science, 2026.

Anthropic. Measuring AI agent autonomy in practice. Vendor, 2026.

BACKUP. LIKELY QUESTION

“Why not just use Claude Code, Codex, or Gemini CLI?”

Use them. They are excellent, and I use them daily. Clio Coder can also dispatch work to installed agents such as Claude Code and Codex.

Open and inspectable	A researcher can read and change the whole harness, and measure it.
Any model	Local, institutional and cloud. Essential for some labs.
Built with scientific code in mind	Numeric comparison, ULP tolerances, pre-registration, HPC login-node install.

BACKUP. LIKELY QUESTION

“Does it understand my Fortran and my Slurm scripts?”

Today

Clio Coder reads the source as text, runs your build, and uses your checks. The structural code map covers ten languages: C and C++ yes, Fortran no.

Clio Coder can reach Slurm through an MCP server, where submit and cancel need your approval or the judge's review.
Clio Coder does not parse batch scripts or submit its own workers through the scheduler.

What it can read

Jupyter notebooks, PDFs, HDF5, NetCDF and Parquet. SQLite, read-only. A Fortran grammar would be a wonderful first contribution.

Yes, I noticed the irony after that FORTRAN slide.

BACKUP. LIKELY QUESTION

“Can I use this on sensitive or export-controlled code?”

The simple answer	Run a local model on your own machine, and nothing leaves. Ask your security office first, always.
Mixed sessions	An information-flow policy names which models a restricted path may reach. Labels survive summaries and resumes.
Known gaps	It does not cover all shell behavior. Contained commands keep network access by default. Credential masking is a denylist. The macOS sandbox profile is unverified on real hardware. Where a host lacks bubblewrap or user namespaces, nothing is contained.
No compliance claim	This is a research tool, and it has no accreditation of any kind.

BACKUP. IF BENCHMARKS COME UP

The benchmark everyone quoted is gone

59.4%

of 138 hard SWE-bench Verified tasks had material flaws in tests or description, in OpenAI's audit. Retired in 2026.

15.7%

of test-passing patches on 26 under-tested Verified instances were wrong (92 of 584), per UTBoost. 28.4% on 23 Lite instances.

Models

could reproduce gold patches or task details for some tasks, which points to contaminated training data.

A passing test suite is one piece of evidence, from a test somebody wrote for some other purpose. Same lesson as ours.

UTBoost, ACL 2025. OpenAI, Why we no longer evaluate SWE-bench Verified, 2026-02-23.

BACKUP. THE STORY FROM IDEA 1

The harness was measuring a product that stopped existing

The replay simulator checked context pressure at the start of each user turn. The real agent had moved to checking before every model request.

When the simulator was fixed, the numbers got worse, and they were finally true.

BUDGET	RETENTION, BEFORE	AFTER THE FIX
32k	0.470	0.323
64k	0.510	0.414
128k	0.649	0.535
200k	0.713	0.715

Replay of 44 real session ledgers, structural-v1 policy, commit a39fd9812, 2026-09-26. Before: turn-start runner. After: per-request runner with a 0.10 rearm band.