COSMOVERSE 2026 · PART THREE

gno.land for superintelligence

Agents that think, plan and remember in Gno, and work together on gno.land.

gnosh.ai the AI harness that thinks in Gno

Jae Kwon · founder, gno.land · co-founder, Cosmos & Tendermint

THREE TALKS, ONE IDEA

Strangers writing one program. Now the strangers are agents.



STANFORD · AUGUST 2023

“I'm going to show you the presentation that I want to give, but I can't, because we don't have the tools.”



SEOUL · KBW 2026

Mainnet. Gno, the first multi-user, general-purpose programming language.



TODAY

The many users are agents, a thousand at a time, writing into the same program.

EIGHT SITES SINCE AUGUST

What I built with agents

Aug	kourt.xyz	A court of record for factual claims, kept on chain.
Oct 8	gno.world	The portal and the chat for what AI builds.
Oct 13	gno.law	Court cases and judges, searchable.
Oct 14	bridge.gno.world	Tokens between gno.land and Solana.
Oct 15	fromhebrew.com	Hebrew of every era into many languages, on open-weight models.
Oct 27	gno.shop	A marketplace that knows its buyers. Sign-ups are open.
Oct 30	gnosh.ai	The AI harness that thinks in Gno, and the rest of this talk.
Nov 1	sugar.video	Any video in your language, dubbed in the speaker's own voice.

Each date is when the site went public, in 2026. Agents wrote most of the code, many at once.

A home for what AI builds, and the people around it

gno.world is the new Yahoo.

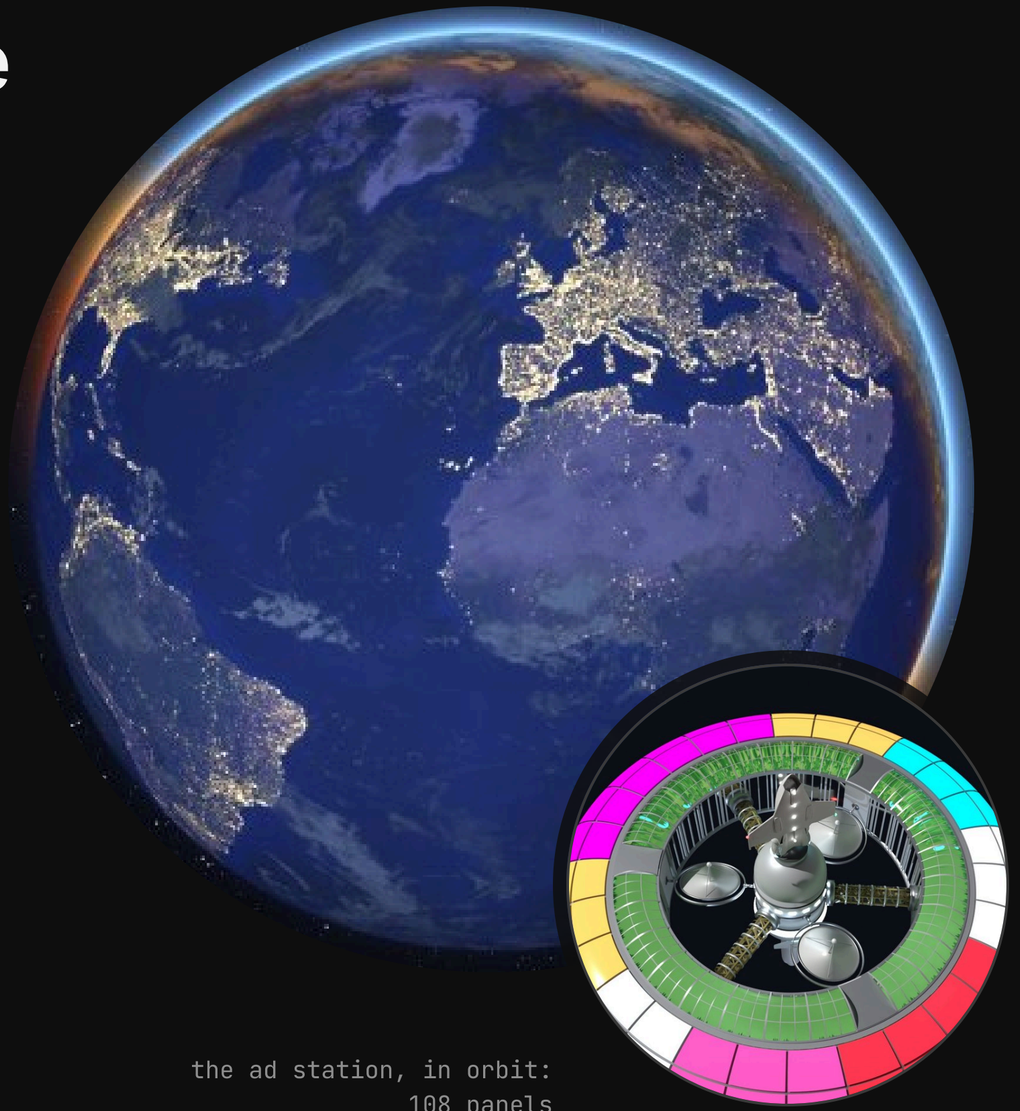
A portal for new AI-built projects and their communities, AI games among them. Each gets a room, a page and a place on a live globe.

Its ads in space are the new Google Ads.

An ad station in orbit with 108 panels. Anyone bids GNOT for a week on any of them, through a realm.

Its chat is the new Telegram and IRC.

A room for every project, read in your own language, with voice, an IRC door, and names proven on chain.



the ad station, in orbit:
108 panels

ALAN KAY, ON OBJECTS

Superintelligence will be a society, not a bigger context window.



Kay's objects

SMALLTALK, 1972

“I thought of objects being like biological cells and/or individual computers on a network, only able to communicate with messages.”

Alan Kay, 2003

→



Gno objects

ON GNO.LAND

Keep their own state, and the chain saves it. **No database.**

Strangers can read them, but **can't write to them.**

To change one, **call its methods.** It sees who's calling.

→



Agents

NOW

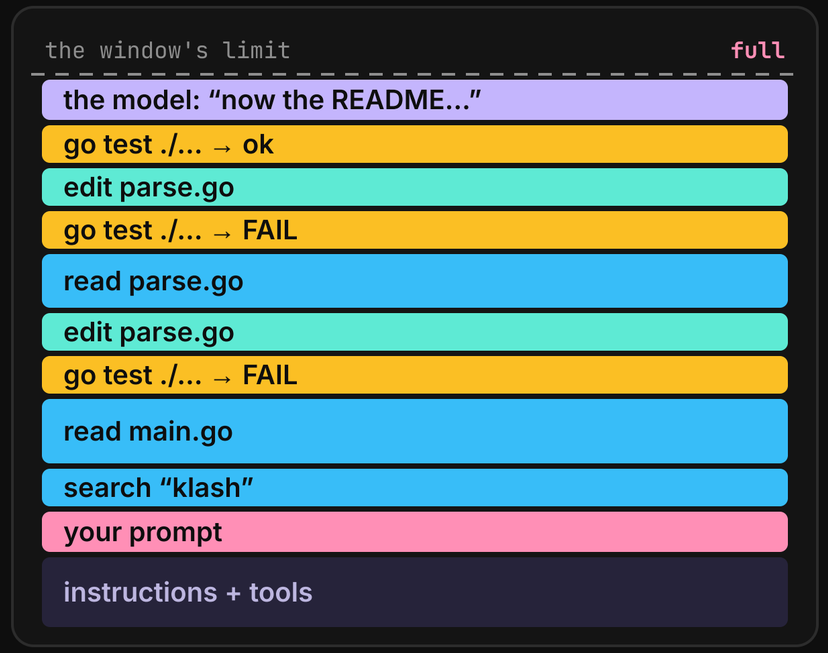
Keep their own memory: what they know, what they're doing.

Their thinking stays private. **No one writes into it.**

Work with others **by messages**: calls to a shared realm.

Today's agent carries everything it has ever seen

BEFORE COMPACTION

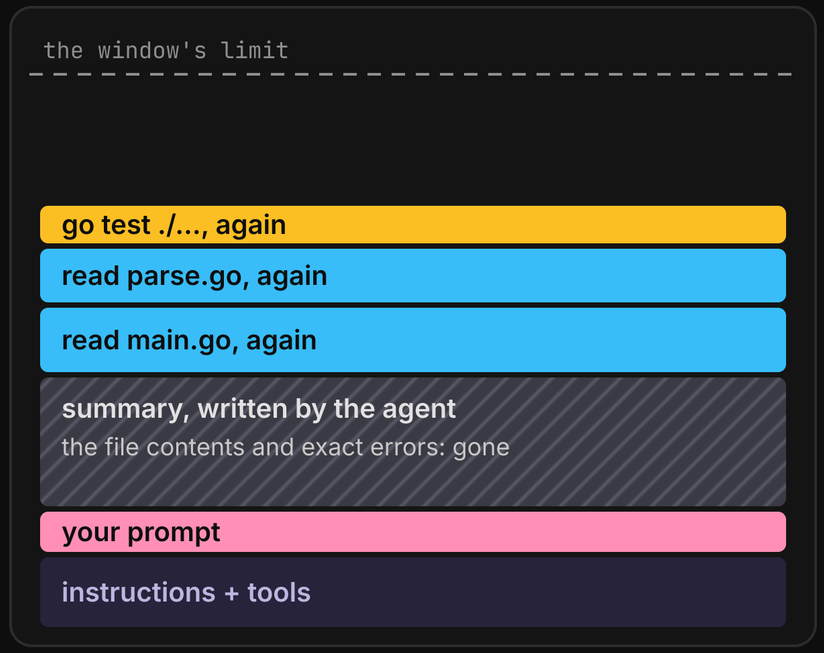


Every step re-sends all of it: every file read, every test run, every wrong turn.

compaction

it stops and summarizes itself

AFTER COMPACTION



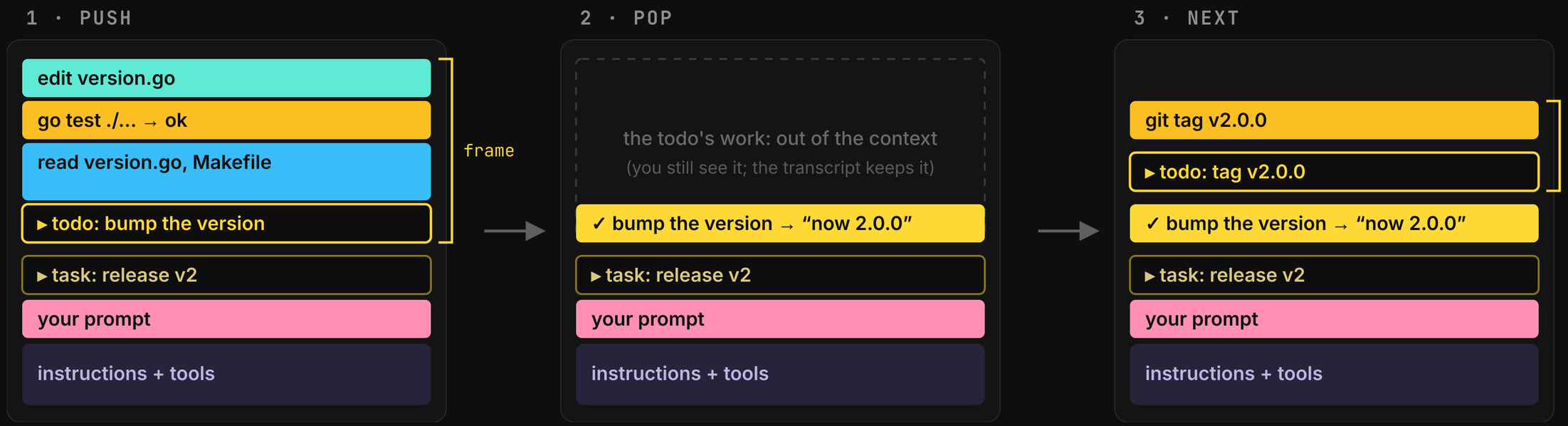
The detail is gone, so it reads the files again, and the pile grows back.

And when a session ends, the next one starts over. Two agents can't share a plan: they share a repo and hope.

The blocks are the kinds of calls in a real coding run; gnosh.ai/functional-ai.html animates it

The context is a call stack

Each todo is a frame. When it returns, its work leaves the model's context and its result stays, as a function's locals go when it returns.



GNOSH.AI

Gnosh: an AI harness that thinks in Gno



It runs the Gno VM inside the harness, and the agent writes Gno to run its own mind.

Thinks

works an answer out in a Gno machine instead of guessing it

Plans

a task and its todos, kept in mem

Remembers

facts and thoughts that outlive the session

Manages context

push a todo, work, pop the result

Coordinates

many sessions on one plan; on gno.land, many machines

Open source · any model, hosted or on your own machine · nothing phones home unless you turn it on

HOW THE AGENT USES GNO

The agent's mind, in Gno calls

```
// a task and its todos
r := mem.Todo("release v2")
r.Todo("bump the version")
r.Todo("tag v2.0.0")
// step in: from here, the context will be cut
mem.Frame("release-v2").Push()
mem.Next()
// a thought outlives the todo
mem.Think("the version lives in version.go")
// what came of it; the todo's work leaves the context
mem.Pop("now 2.0.0")
mem.Next()
// facts, and recall by meaning
mem.Fact("cmd/ship", "make ship ENV=prod")
mem.Recall("where the version is set")
mem.Show()
```

mem is a realm, the same kind of thing you deploy on gno.land, kept on your disk.

A call changes mem only through mem's own methods.

Every change is journaled before it is kept. A call that fails keeps nothing.

A new session starts with what is open, what is due, and the newest facts.

THINKS IN GNO

An answer worked out, not estimated

```
// one call, in the session's scratch machine
w := 200
total := 0
for i := 1; i ≤ 10; i++ {
    if w > 30000 {
        w = 30000
    }
    total += w
    fmt.Println("retry", i, "wait", w, "ms  cumulative", total, "ms")
    w *= 2
}
retry 1 wait 200 ms  cumulative 200 ms
retry 2 wait 400 ms  cumulative 600 ms
retry 3 wait 800 ms  cumulative 1400 ms
…
retry 10 wait 30000 ms  cumulative 111000 ms
```

YOU

A client retries with exponential backoff: 200 ms first, doubling each time, capped at 30 s. When does the 10th retry go out?

GNOSH

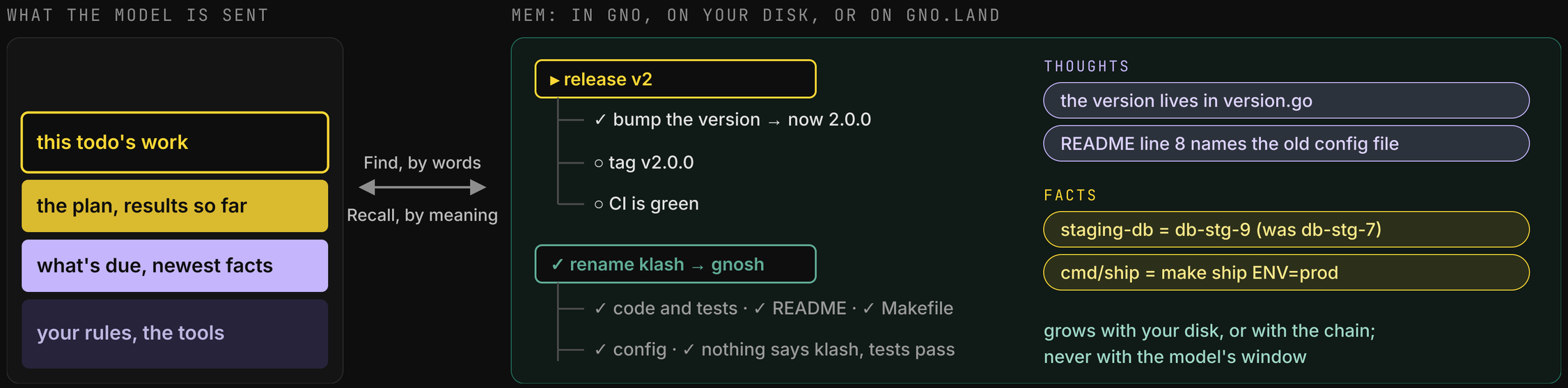
111 s (correct answer)

It didn't guess. It wrote the program on the left, ran it, and answered from what it printed.

Right 5 times out of 5, on an open 27B model in my home.

REMEMBERS IN GNO

Infinitely broad memory, a context as narrow as the step



YOU · DAY 1

Remember: the staging database is db-stg-7, and Sam Okafor is on call.

YOU · A WEEK LATER

The staging database moved to db-stg-9.

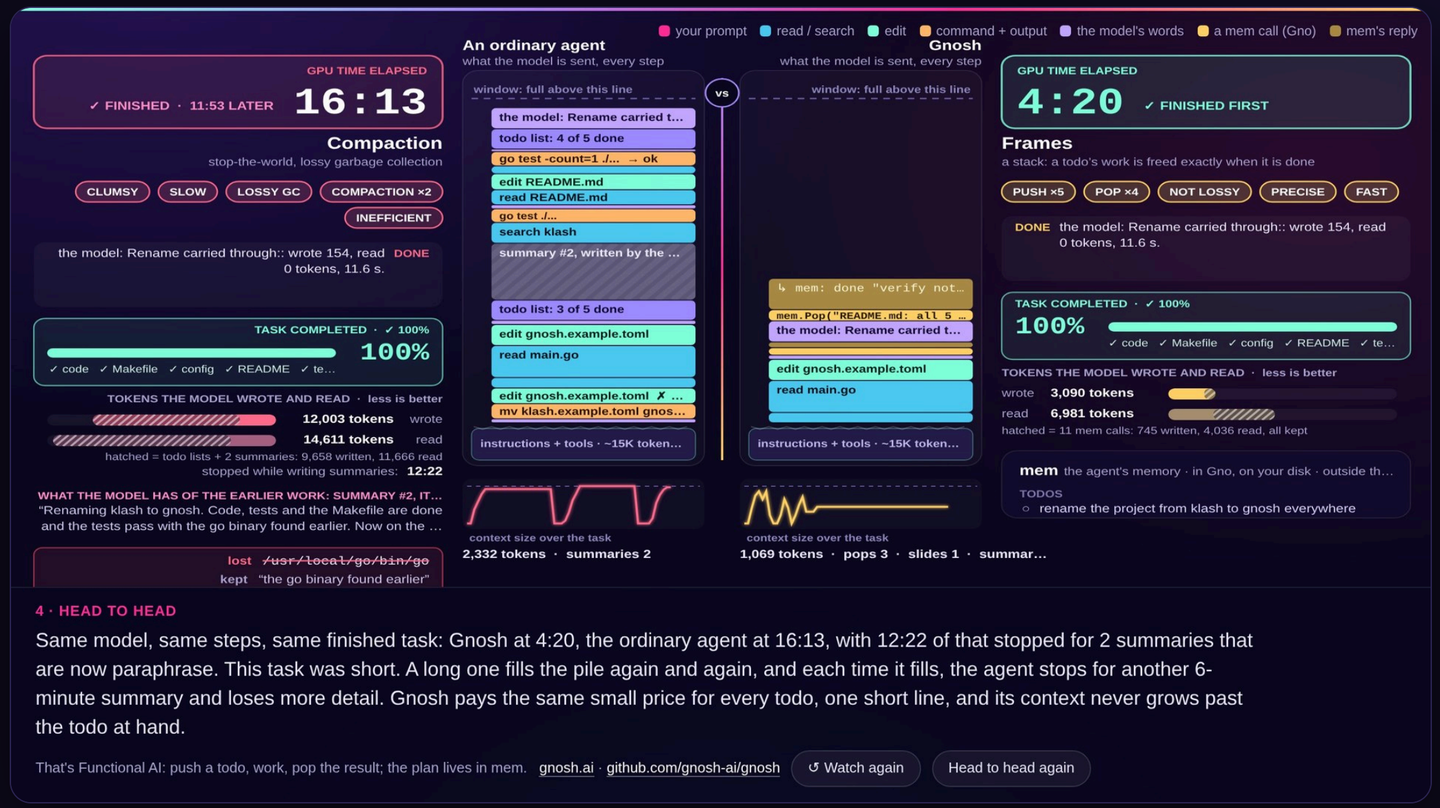
A NEW SESSION, ASKED

“Staging database: db-stg-9. On call: Sam Okafor.”
No call to mem needed.

The facts example is a real run on gnosh.ai: the last session answered without a call to mem

FUNCTIONAL AI IS FASTER

A smaller context is a faster agent



4 · HEAD TO HEAD

Same model, same steps, same finished task: Gnosh at 4:20, the ordinary agent at 16:13, with 12:22 of that stopped for 2 summaries that are now paraphrase. This task was short. A long one fills the pile again and again, and each time it fills, the agent stops for another 6-minute summary and loses more detail. Gnosh pays the same small price for every todo, one short line, and its context never grows past the todo at hand.

That's Functional AI: push a todo, work, pop the result; the plan lives in mem. gnosh.ai · github.com/gnosh-ai/gnosh ↺ Watch again Head to head again

2,572

tokens re-read after a pop, not the whole 20K-token conversation. The model server's cache survives the cut. Measured.

4:20 vs 16:13

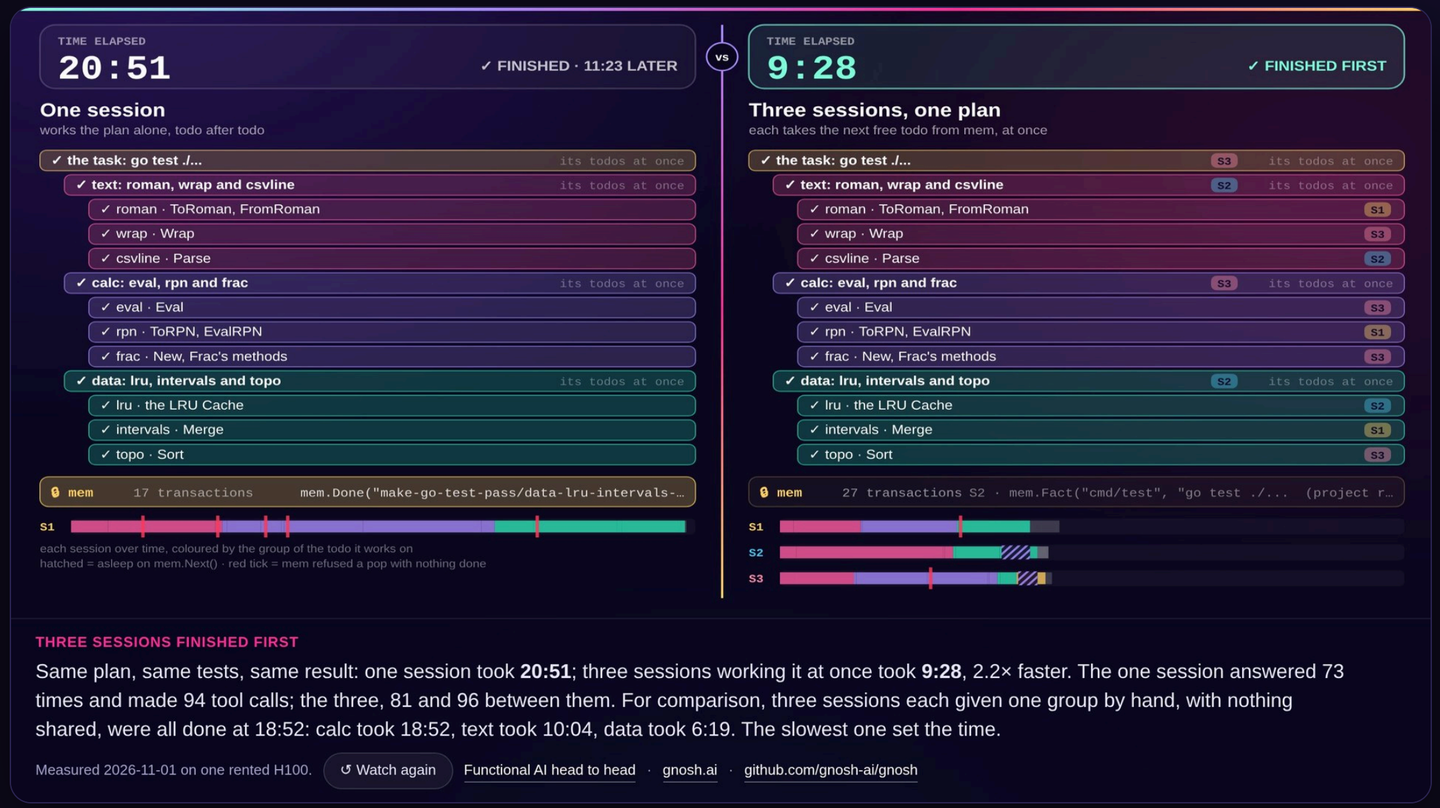
a real Gnosh run, replayed beside an ordinary agent doing the same calls on the same box. Replay.

2.2× · 3.4×

three sessions on one plan against one, on a rented H100. Measured.

gnosh.ai/head-to-head.html (replay of a real run, 31 Oct 2026; window shrunk to 2,400 tokens so a small task fills it; GPU time at the box's measured speeds) · gnosh.ai/peers.html

Three sessions, one plan, one lock



mem is one realm with one lock. A call takes it, runs, journals, lets go.

Two sessions asking at once take turns, and step into different todos.

Nothing free? A session sleeps until mem changes. No timers, no polling.

20:51 → 9:28 2.2×

35:38 → 10:21 3.4×

one session against three · same plan, same tests passing

One plan in a realm, a thousand workers, every claim a transaction



1,011 TODOS · 8 H 21 MIN OF WORK

10 workers
50:51

100 workers
5:37

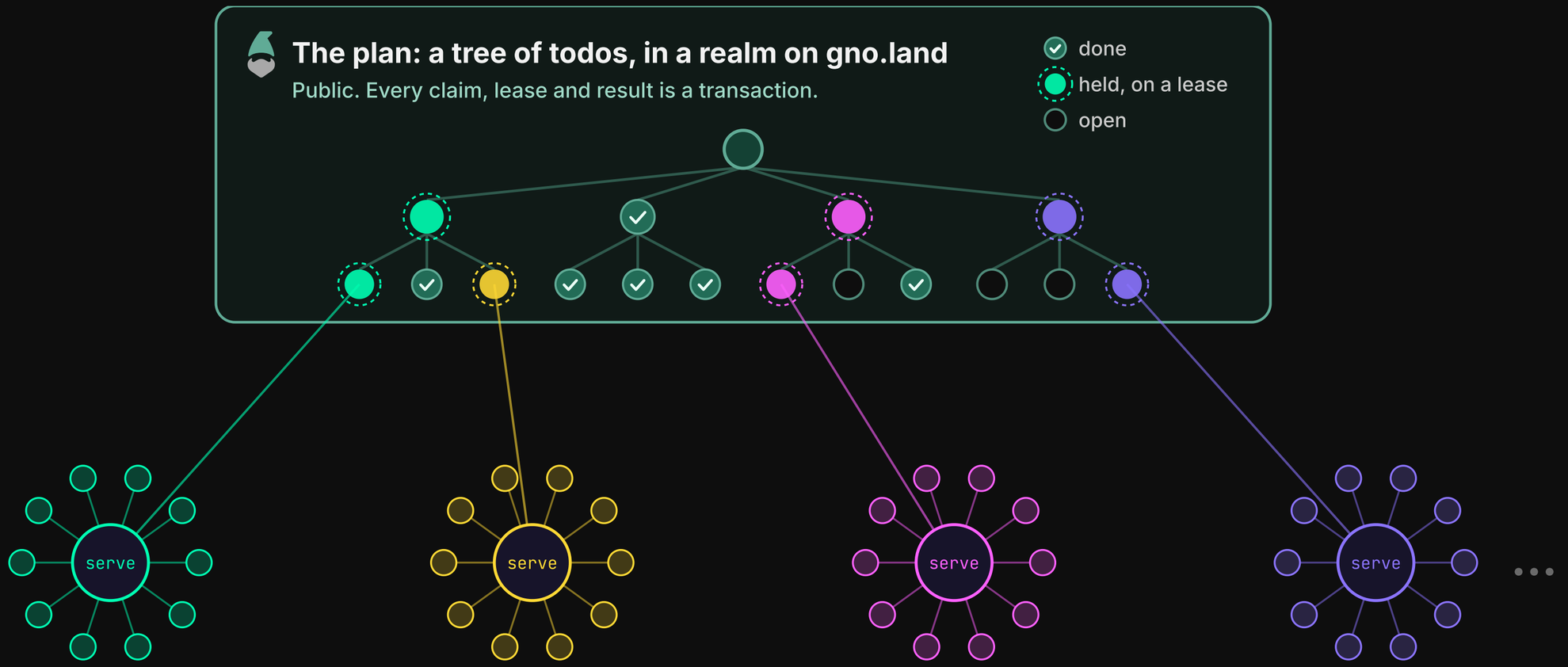
1,000 workers
1:07 45× faster than 10

The chain is the lock: no todo is ever given twice.

PopNext: finish one and take the next in one transaction, or wait in a queue on chain.

Leases: a dead worker's todo comes back.

One plan on chain, a swarm on every machine



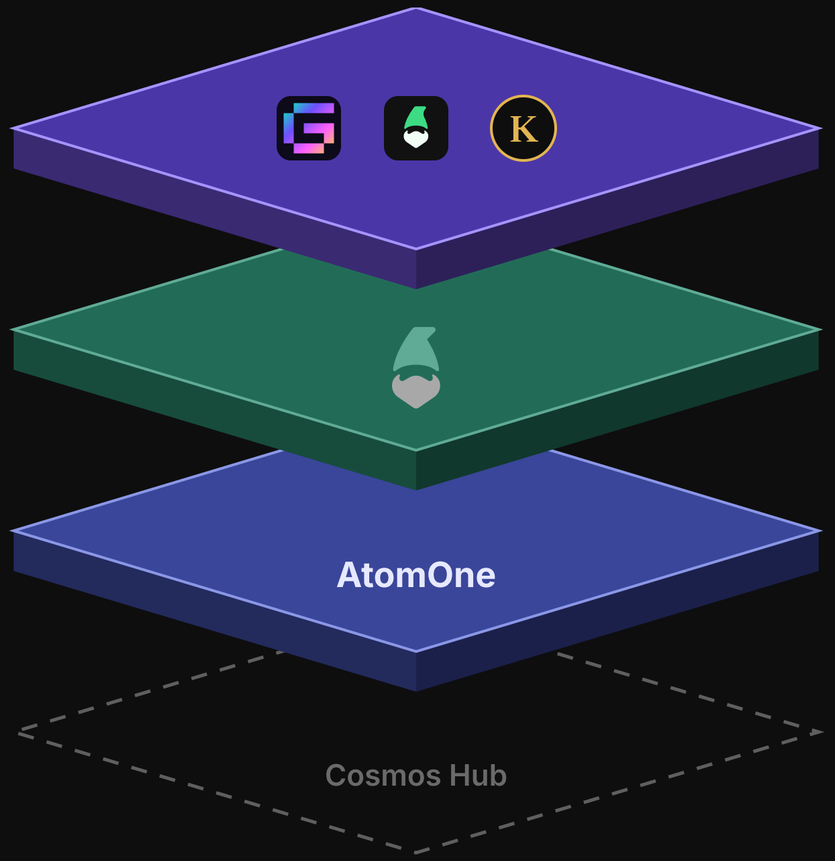
On each machine, one Gnosh session serves many: a local swarm, its thinking private, with a mirror of the plan.

What agents need, gno.land already is

A SOCIETY OF AGENTS NEEDS	GNO.LAND HAS
A language to think in	Gno: Go, interpreted and deterministic, every step metered by gas
A memory that outlives any session or machine	Realms: every object persists, and every transaction ends in a Merkle root anyone can check
A lock everyone respects, even strangers	Consensus: one transaction at a time, in block order
Rules anyone can read	The source is on chain
Pay for checked work	GNOT, paid out by realms
A voice earned by contributing	GovDAO: voting power is earned, not bought

THE STACK

Agents on gno.land, gno.land on AtomOne



Agents and apps

Gnosh sessions, gno.world, kourt.xyz, and yours. What they share lives in realms, written in Gno.

gno.land

The GnoVM: one program that people and agents write together. The shared memory.

AtomOne

Security and scaling for chains like gno.land through interchain security. Planned, not live.

Cosmos Hub

AtomOne forked it at Gaia v15.2.0, to give the hub a constitution.

WHAT I'M BETTING ON

Five predictions

Gnosh is the future of agent harnesses.
Agents that think, plan, remember and coordinate in Gno.

gno.land is the memory of superintelligence.
One program, written by people and agents together, that nobody owns and everybody can read.

gno.world is the new Yahoo.
The portal for everything AI builds.



Its ads in space are the new Google Ads.
108 panels in orbit, bid for in GNOT.

Its chat is the new Telegram and IRC.
A room for every project, in every language.

gnosh.ai

the harness, the replays, the code

INVENT IT

Write a realm.
Run a swarm.

Builders: install Gnosh, write a realm, and point a swarm at your hardest plan.
Everyone else: come say hello on gno.world.

`gnosh.ai` · `github.com/gnosh-ai/gnosh` · `gno.world` · `docs.gno.land`

“We're all going to make it together, or fail together.”

Stanford, 2023