



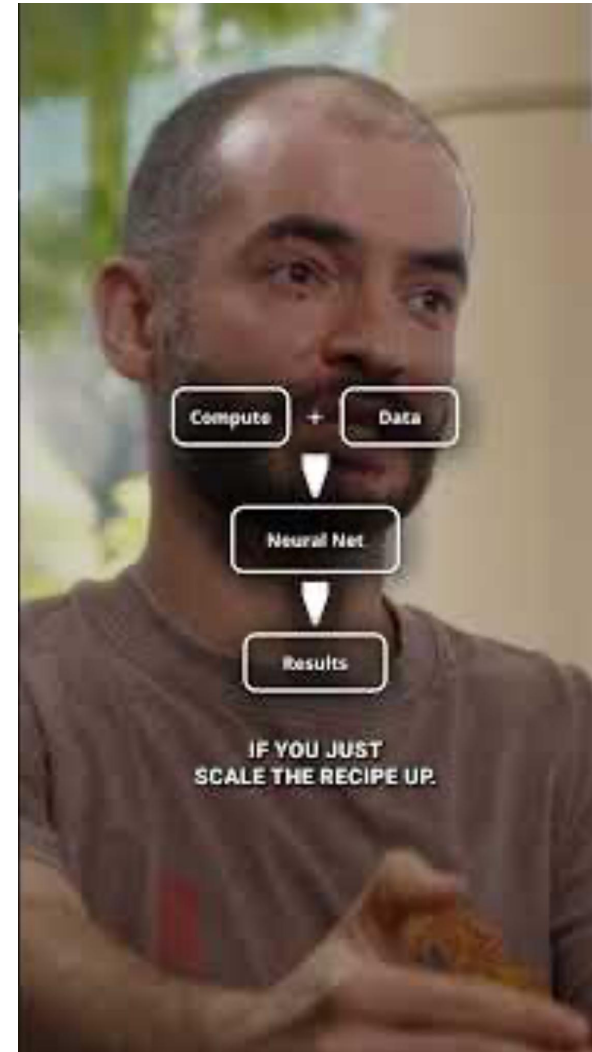
Agentic Engineering

AI agents that drive CAD, FEA, CFD and simulation tools: state of the art, parallelization,
and scaling laws

September 2026

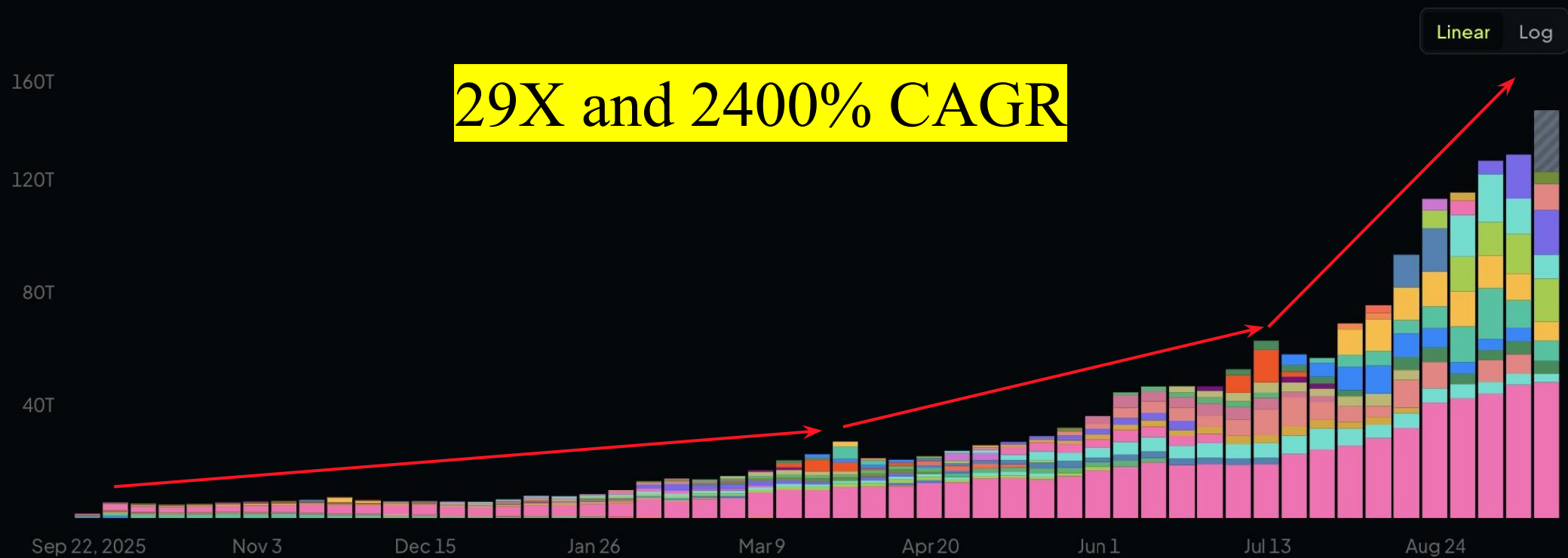
Time	Block	
0–5	Overview	Where intelligence in an agent loop comes from
5–17	I. State of the art	SOTA for CAD, CAE, system sim and HPC agents, and where they fail
17–32	II. Scaling laws	State the relation for training, test-time and agent/parallel compute
32–45	III. Parallelization	Choose single vs. parallel vs. orchestrator–worker with numbers; read the demo
45–55	IV. Doing it right	Build a verifiable, sandboxed, cost-accounted agentic workflow
55–60	Exercise + Q&A	Do the back-of-envelope design problem
	Cost today and cost tomorrow	

Math progress from high school math to long horizon math. Not perfect but exponential.



📊 Top Models

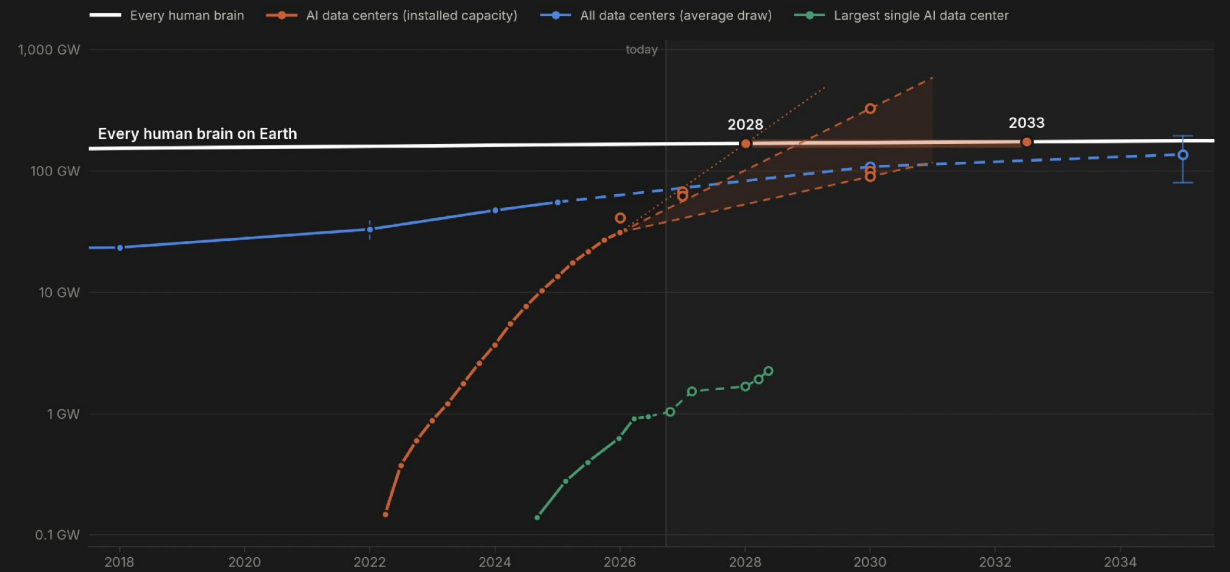
Weekly usage of models across OpenRouter





AI out-draws every human brain on Earth around 2028–2033

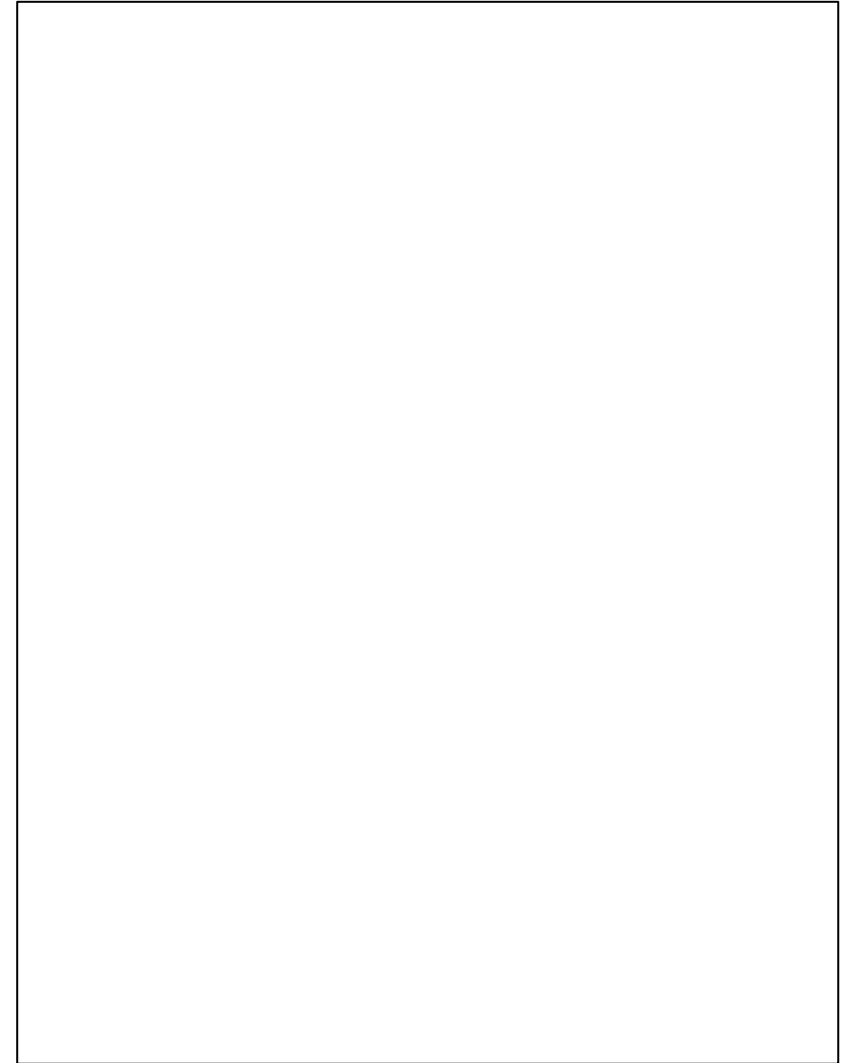
Gigawatts, log scale. Brains at 20 W each. Orange band: range of published forecasts; dotted line: 2025 growth (2.3×/yr) continued. Hollow markers are forecasts.



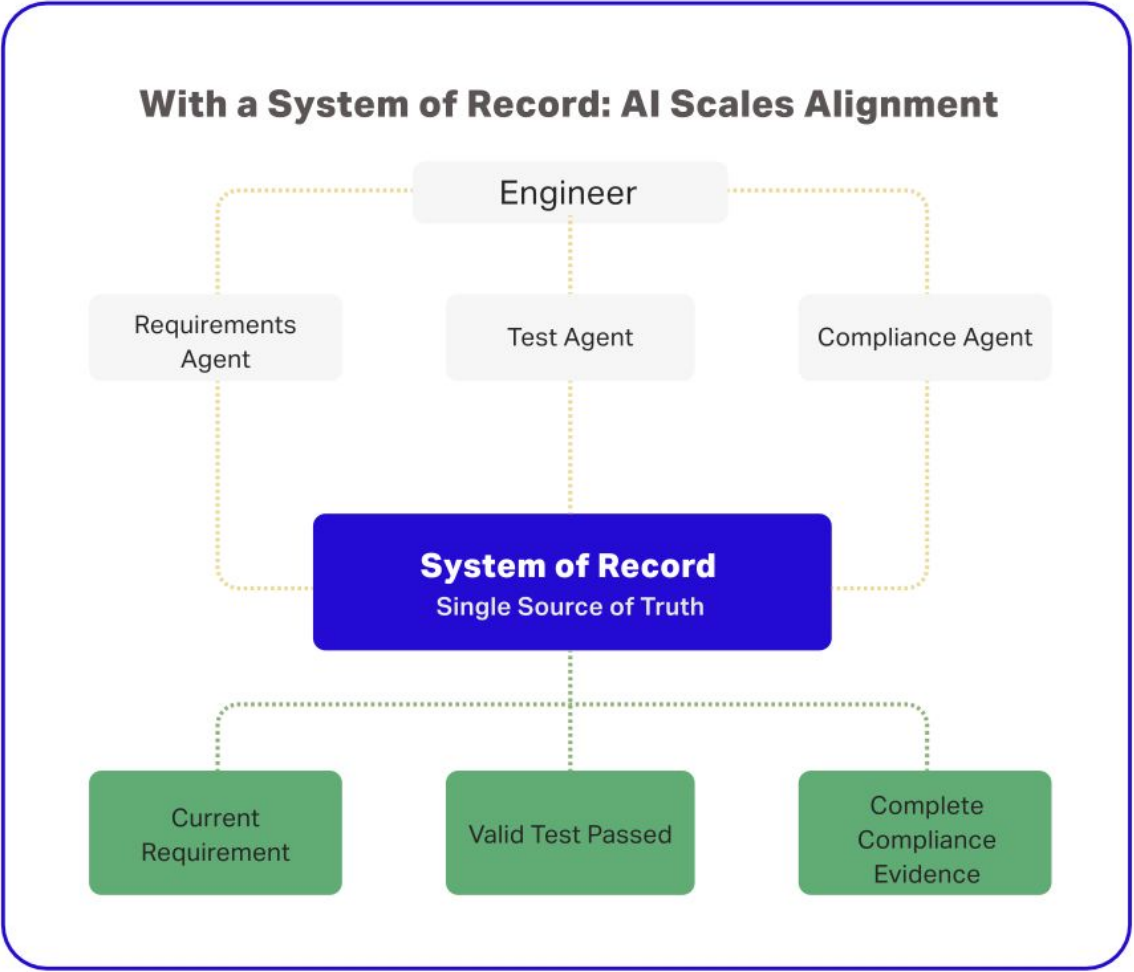
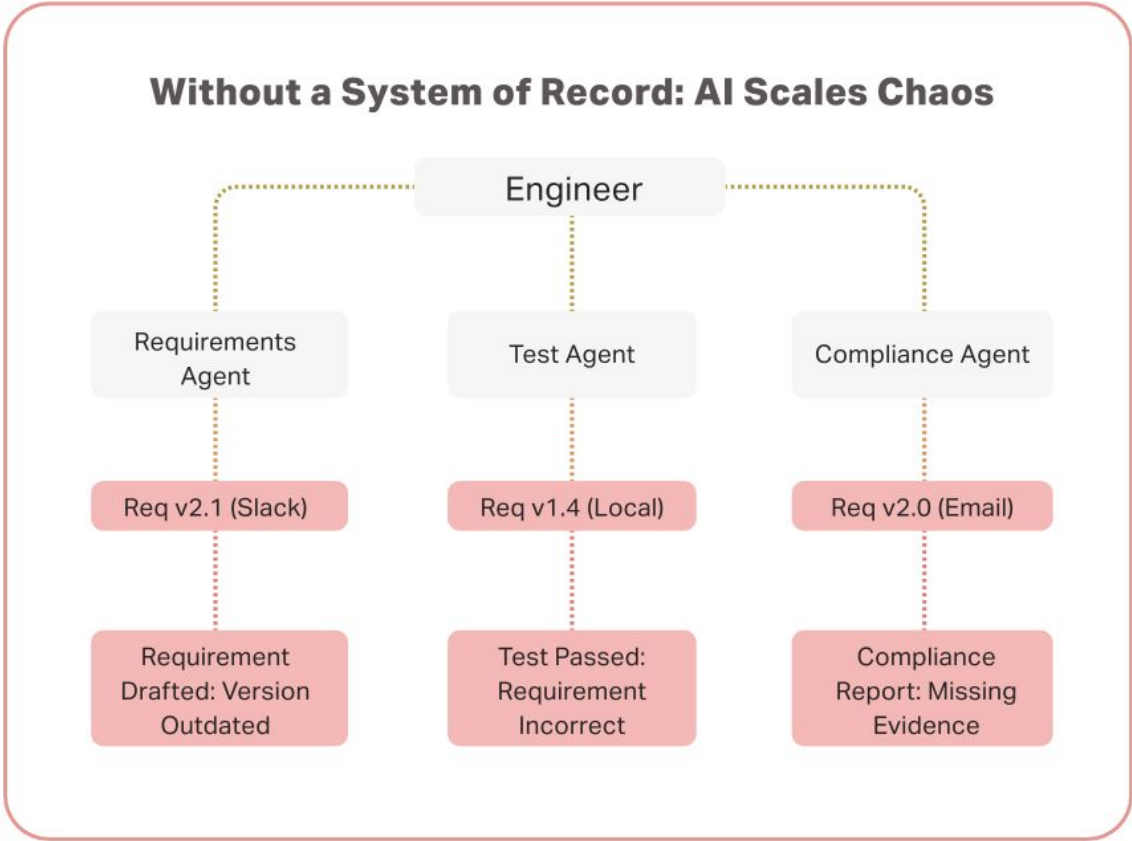
Sources: Epoch AI (AI capacity from chip sales; frontier data centers), IEA Energy and AI 2025/2026, Masanet et al. 2020, RAND 2025, Gartner 2026, JLL 2026, McKinsey 2024, SemiAnalysis 2024, UN WPP 2024.

In CAD or in an engineering tool though AI, where does the intelligence come from?

- An agent writes **CadQuery**, exports **STEP**, meshes it, runs **FEA**, reads the stress field, and revises the part
- Each iteration is scored by a **deterministic tool**, not by the model
- Question for the room: **the model, or the loop?**
- Answer we will build toward: the model proposes; the tool interface, feedback channel and control structure make it engineering



Systems of record

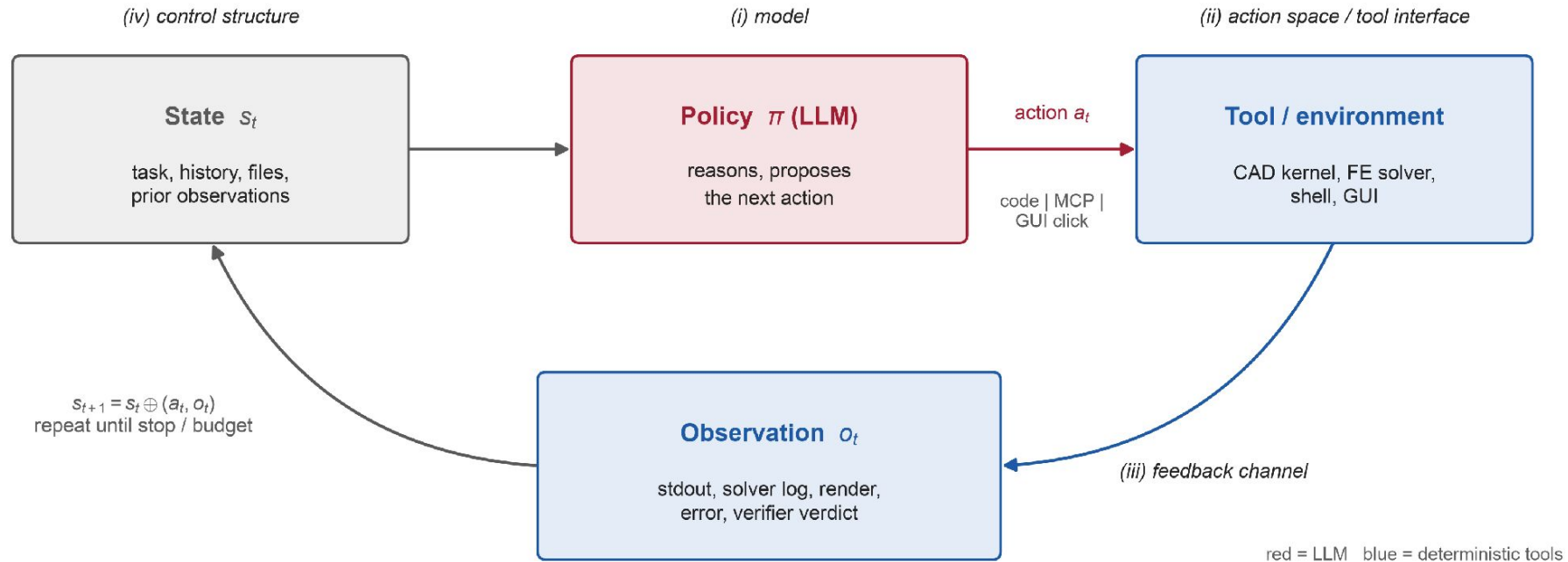


Part I · State of the art

Agent anatomy, tool interfaces, CAD / CAE / system-simulation agents, failures,
benchmarks

Anatomy of an agent: a policy in a loop with tools

An agent is a policy in a loop with tools; the loop, not the model, carries most of the engineering



- Four design levers: (i) **model**, (ii) **action space / tool interface**, (iii) **feedback channel**, (iv) **control structure**
- Most engineering gains come from (ii)–(iv), not (i) alone

ReAct (Yao 2022, 2210.03629) · Reflexion (Shinn 2023) · SWE-agent ACI (Yang 2024, 2405.15793)

Three ways an agent can hold an engineering tool

Interface	Example	Strength	Weakness
Code-as-action	CadQuery / build123d, FreeCAD Python, COMSOL / Abaqus APIs, OpenFOAM dicts, FEniCS	Deterministic, diffable, parametric; easy to verify and parallelize	Needs a scriptable tool; model must know the API
Structured calls / MCP	mesh(part, h=2mm), solve_static(...) wrappers	Narrow, safe, documented action space	You build and maintain the wrappers
GUI / computer use	Clicking through FreeCAD or a proprietary CAE GUI	Works on any legacy tool	Slow, brittle, hard to verify; lowest success today

- Rule: prefer the **most executable, most verifiable** representation (parametric code → STEP, not meshes or pixels)

MCP open standard (Anthropic, Nov 2024)

SOTA: CAD generation and editing

- **Son et al. 2026:** CadQuery → STEP → render / validate / FEA feedback. First attempt: **no strict-pass artifact** from Codex (GPT-5.5) or Claude Code (Opus-4.7); $\leq \sim 20\%$ of typed requirements met
- **CADEngBench 2026:** does CAD survive edits and parameter changes? 300 parts, 164 with a CalculiX FEA test (600 tasks + 150 assembly pairs)
- **CADWorld 2026:** 200 long-horizon FreeCAD GUI tasks in a VM, executable checks (geometry, constraints, parametric structure). Best agent **17.5%** vs. expert 87%
- Grounding: Fusion 360 Gallery (Willis 2021); MIT DeCoDE VLM study (Picard 2023)

Son 2605.17448 · Singh 2608.09296 · Dong 2609.16251 · Willis TOG 2021 · Picard 2311.12668

You don't need a frontier model to click buttons.

— SYSTEM-1 MODEL FOR CAD

Taiga-S1

A 1.2M-parameter model that builds CAD parts in FreeCAD

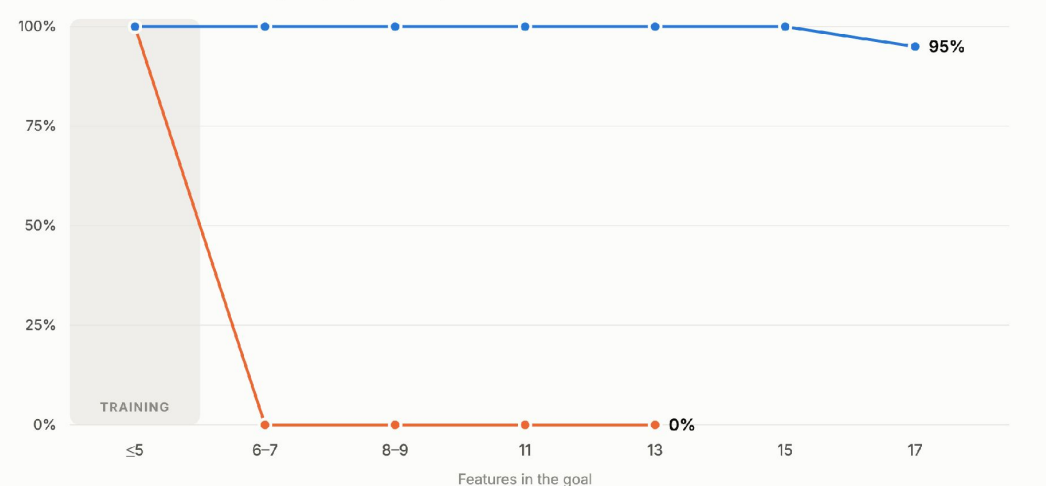
Taiga (1.2M parameters)

<https://huggingface.co/shhivv/taiga-s1>

<https://github.com/shhivv/taiga-s1>

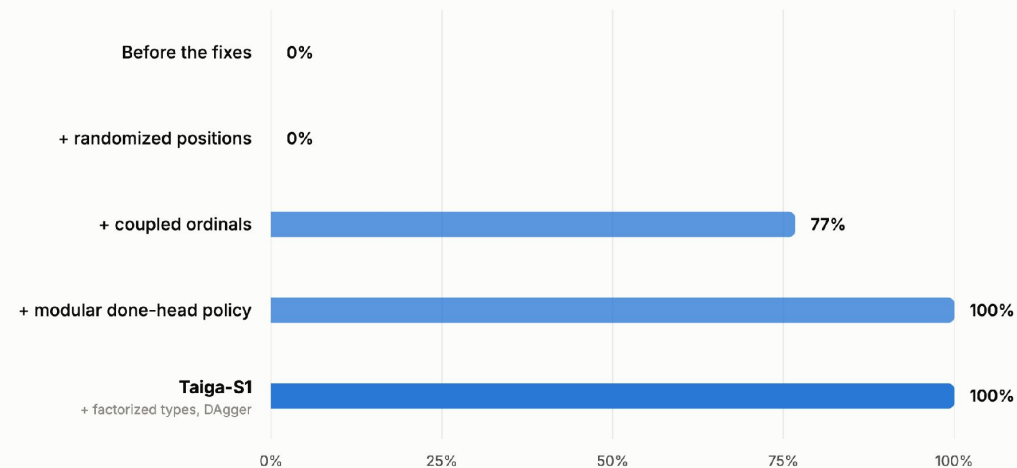
Parts built correctly vs. goal length

Clean success in live FreeCAD · 100 goals per point (60 beyond 11 features)



What made it generalize

Parts built correctly on 11-feature goals · average of the training runs for each step



SOTA: finite elements, CFD, multiphysics

- **FEABench** (Mudur 2025): agents drive COMSOL via its API, read solver output, iterate; scored on numerical targets
- **FEM-Bench** (2025) structured FEM reasoning · **ALL-FEM** (2026) fine-tuned FEM agents · **FeaGPT** (2025) end-to-end FEA
- **MCP-SIM** (Park, engrXiv 2025): shared-memory agents build, run, diagnose and revise FEniCS models
- **CFD**: ChatCFD (Fan 2025) end-to-end OpenFOAM agent; NL2FOAM (Dong 2025) fine-tuned 7B model
- Discussion paper: *What do CAE agents need beyond a generic harness?* (Shi 2026)
- MIT: **MechAgents** (Ni & Buehler, EML 2024), multi-agent elasticity with FEniCS

2504.06260 · 2512.20732 · 2603.21011 · 2510.21993 · 2506.02019 · 2504.09602 · 2609.03718 · doi:10.1016/j.eml.2024.102131

SOTA: system simulation, HPC, and discovery loops

- **Simulink:** SimuGen (Ren 2025), SimuAgent (Liang 2026); ASWE-Bench (Abdalla 2026)
- **Modelica:** Pufibara harness + Modelica Agent Workflow Benchmark (Wang 2026)
- **FMU / control:** Agent-in-the-Loop system ID + PI tuning (Björkskog 2026)
- **HPC (EngiAI 2026):** Gemini-3-Flash completed every tested pipeline; GPT-5-mini reached the final step in **50–70%** of trials, depending on prompt style
- **Discovery loops:** FunSearch (Nature 2023), AlphaEvolve (2025): LLM mutations + automated evaluator + evolutionary selection, in parallel

2506.15695 · 2601.05187 · 2608.23653 · 2605.19743 · FunSearch doi:10.1038/s41586-023-06924-6 · 2506.13131

Where SOTA still fails

- **Spatial / geometric reasoning:** picking the right B-Rep face; tolerances and clearances
- **Boundary conditions and units:** silent physics errors
- **Long-horizon reliability:** errors compound over many tool calls (toy model A, Part III)
- **GUI-only tools:** slow, brittle, hard to verify (but CUA is getting really good)
- **Recognizing non-physical results:** mesh dependence, singular stresses at re-entrant corners, unconverged solves

Benchmarks: a benchmark is only as good as its verifier

Benchmark	Measures	Verifier
SWE-bench Verified	Repository-level code fixes	Hidden unit tests
FEABench	Multiphysics via the COMSOL API	Numerical target values
CADEngBench	Parametric CAD robustness, assembly, FEA	Executable predicates, FEA
CADWorld	GUI CAD over long horizons	Executable checks on saved FreeCAD files
METR time horizons	Autonomy vs. human task length	Task-specific scoring

- Similarity to a single reference **penalizes valid alternatives** and **rewards invalid similar work**

Jimenez 2310.06770 · Mudur 2504.06260 · Singh 2608.09296 · Dong 2609.16251 · Kwa 2503.14499

Part II - Scaling laws

Training compute →

test-time compute →

repeated sampling →

METR time horizons →

multi-agent scaling

Three scaling axes for agents

Axis	Governing relation	Key result
1. Training compute	$L(N, D) = E + A/N^\alpha + B/D^\beta$	Chinchilla: N and D both scale as $C^{0.5}$; ~20 tokens/parameter (rule of thumb)
2. Test-time compute	Accuracy vs. thinking budget; revise vs. search	Snell: >4× over best-of-N; beats a 14× larger model on medium-difficulty problems
3. Parallel sampling	$\text{pass}@k = 1 - C(n-c, k)/C(n, k)$	Coverage roughly log-linear in k; needs a verifier to select

- Skepticism: many “emergent” jumps are artifacts of discontinuous metrics (Schaeffer 2023)

Kaplan 2001.08361 · Hoffmann 2203.15556 · Snell 2408.03314 · Chen 2107.03374 · Schaeffer 2304.15004

Repeated sampling moves the bottleneck to the verifier

- **Large Language Monkeys**: coverage grows **roughly log-linearly** in k over four orders of magnitude; fit by an **exponentiated power law**
- SWE-bench Lite: **15.9%** (1 sample) → **56%** (250 samples)
- With a perfect checker (unit test, FEA pass/fail) sampling works; with majority vote or reward models gains **plateau beyond several hundred samples**
- Chen et al. 2024: voting can be **non-monotonic** in the number of calls. Easy queries improve :-), hard ones degrade :-(

Brown 2407.21787 · Chen 2403.02419

How fast is it getting better:

Agent-level trend: METR 50% time horizon

- Horizon = human-expert task length at which the agent succeeds 50% of the time; logistic fit vs. log task length
- 2019–2025: doubling roughly every **7 months**; the paper notes it may have accelerated in 2024
- 2026 updates (Time Horizon 1.1) suggest ~**4-month** doubling; METR cautions that measurements above 16 h are unreliable
- Caveats: mostly software tasks; large 50% vs. 80% gap; few long tasks at the top end
- Engineering tasks (CAD, CAE) have not been measured this way: an open research opportunity

Scaling agent systems (Kim et al.)

- 260 controlled configurations, six benchmarks, five architectures, three model families (v3)
- Relative to one agent: +80.8% (decomposable financial reasoning) to **-70.0%** (sequential planning, PlanCraft)
- Capability saturation: coordination gives diminishing or negative returns once the single agent exceeds ~45%
- **Tool–coordination trade-off**: tool-heavy tasks suffer most from multi-agent overhead at fixed budget
- **Error amplification** (trace-level): independent ~17.2X, centralized with verification ~4.4X
- Predictor picks the best architecture for 87% of held-out configs (CV R^2 0.37: a guide, not a law)

Kim et al. (Google Research, Google DeepMind, MIT) arXiv:2512.08296 v3, Apr 2026

Orchestrator data, and the case against multi-agent

- BrowseComp: token usage alone explains 80% of variance; with tool calls and model choice, ~95%
- Cost: ~15X the tokens of a chat interaction
- Counterpoints: *Don't build multi-agents* (Cognition 2025); Gao et al. 2025; debate often does not help (Wang 2024; Smit 2024)
- Scale laws for teams: MacNet logistic growth to >1000 agents; Ringelmann effect (2026); MAST: 14 failure modes

Gao 2505.18286 · Qian 2406.07155 · Bertalaníč 2606.02646 · Cemri 2503.13657

Synthesis: what parallel agents buy

- **Buy:** (a) more total tokens and reasoning, (b) separate context windows, (c) speed
- **Cost:** coordination overhead, lossy compression between agents, error propagation
- **Only pays off when** the task **decomposes into independent sub-problems** and verification can be **centralized**
- Default for engineering: one strong agent + **parallel evaluations** (simulations)

Part III · Parallelization

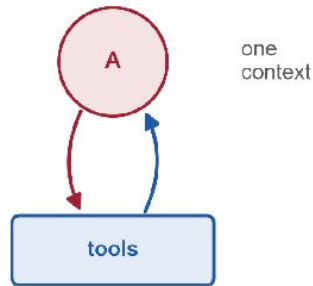
Topologies, three toy models, where parallelism lives, and a live multi-fidelity demo

Five popular topologies: verification decides error propagation

Five agent topologies: they differ in where verification sits and how errors propagate

red = LLM agent blue = verifier / evaluator

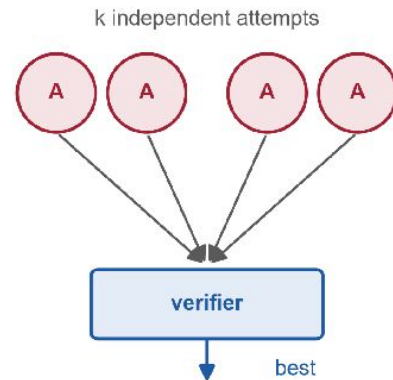
Single agent (SAS)



Good for: sequential,
tightly coupled design

Risk: context overflow;
slow

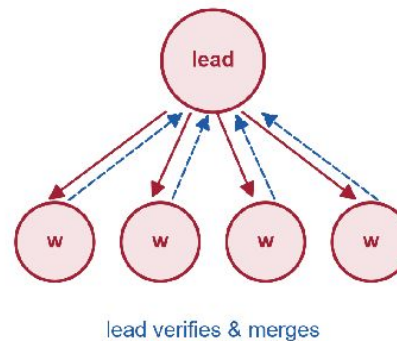
Best-of-k + verifier



Good for: tasks with a
cheap exact checker

Risk: correlated failures;
verifier is the bottleneck

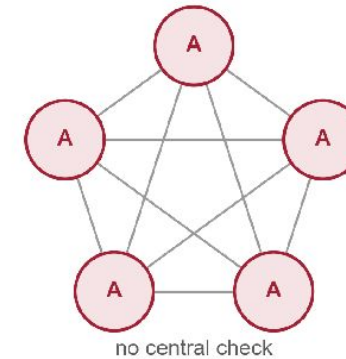
Orchestrator-worker



Good for: design sweeps,
multi-load-case analysis

Risk: poor decomposition;
lead is a bottleneck

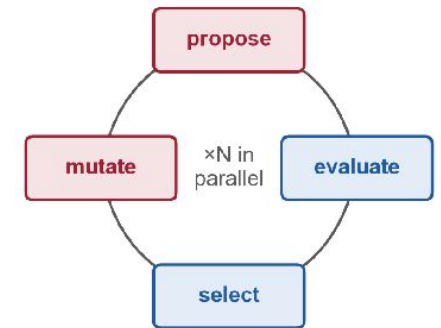
Decentralized / debate



Good for: exploration,
dynamic environments

Risk: consensus failure;
error cascades (~17x)

Evolutionary population

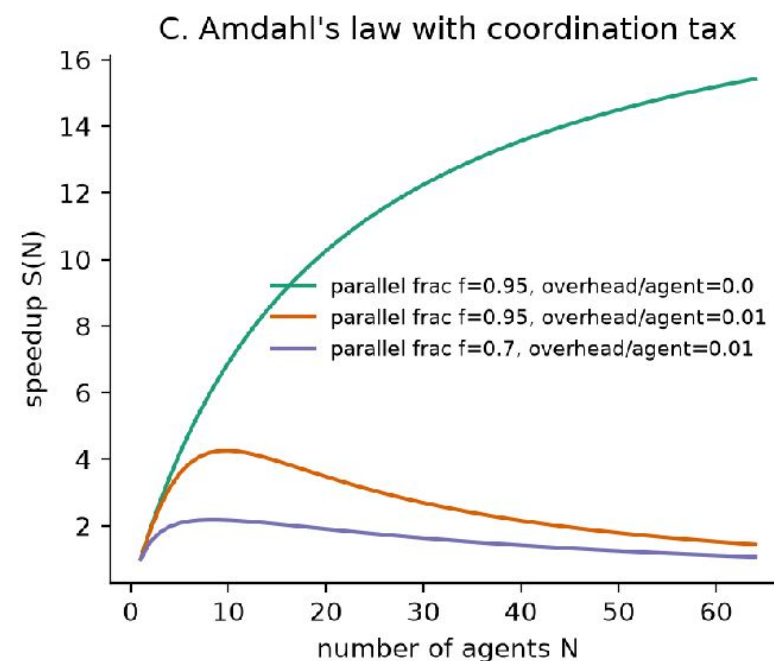
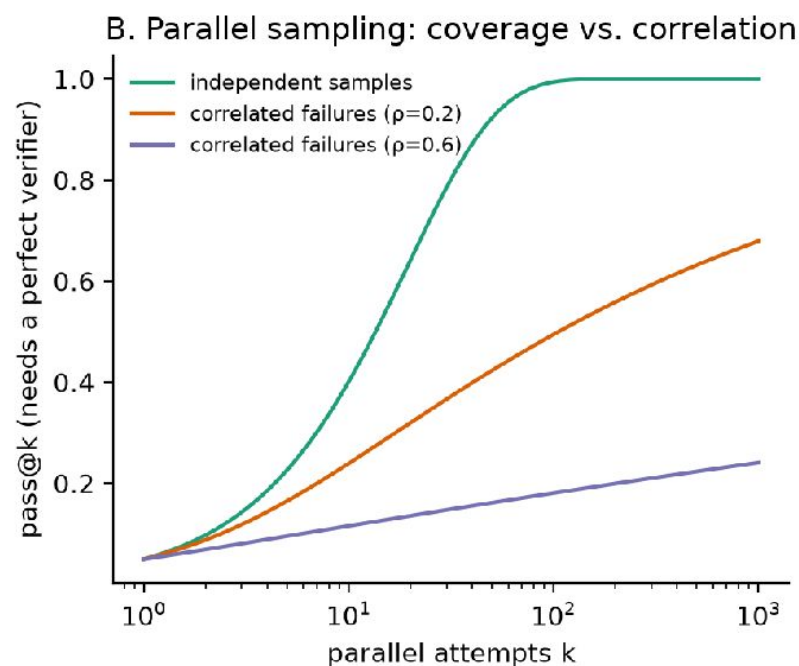
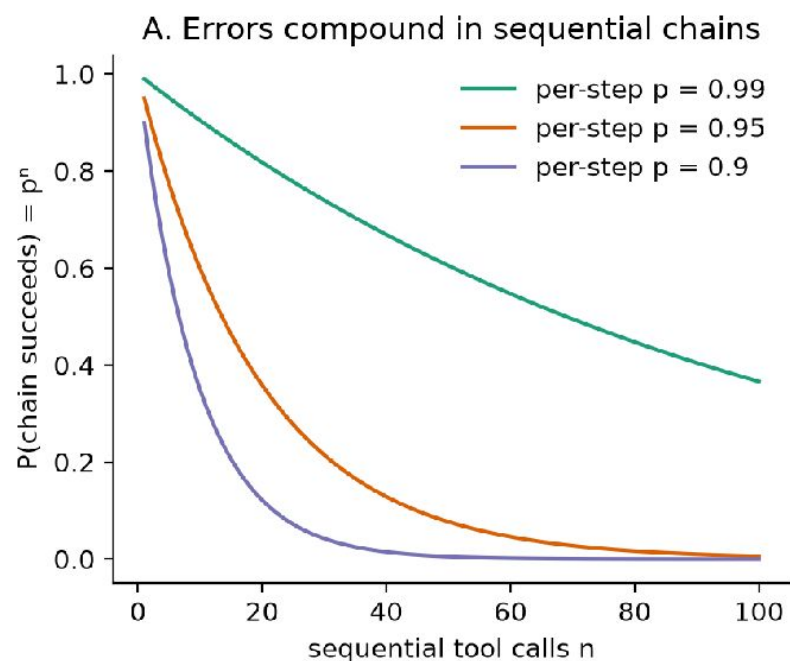


Good for: shape / topology,
gear & linkage synthesis

Risk: evaluator gaming
(reward hacking)

- Orchestrator-worker with central verification contains error amplification (~4.4X vs ~17.2X independent)

Topology-error link: Kim et al. 2512.08296

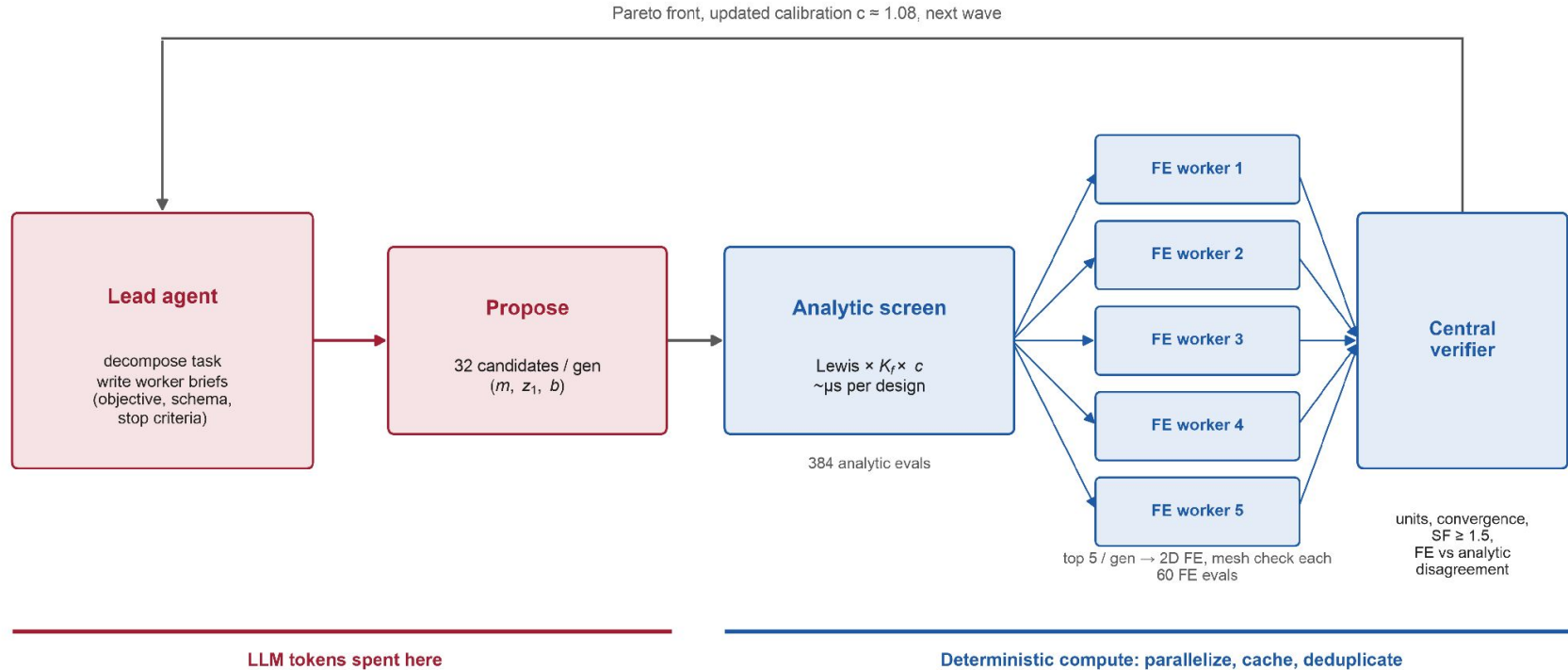


- **A** p^n : $p = 0.95, n = 50 \rightarrow 0.077$. Check inside the loop
- **B** correlated failures (ρ) flatten pass@k: pay for **diversity**
- **C** $S(N) = 1/[(1-f) + f/N + c(N-1)] \rightarrow \text{optimum } N^* \approx \sqrt{f/c}$

Kim 2512.08296 saturation, Ringelmann 2606.02646

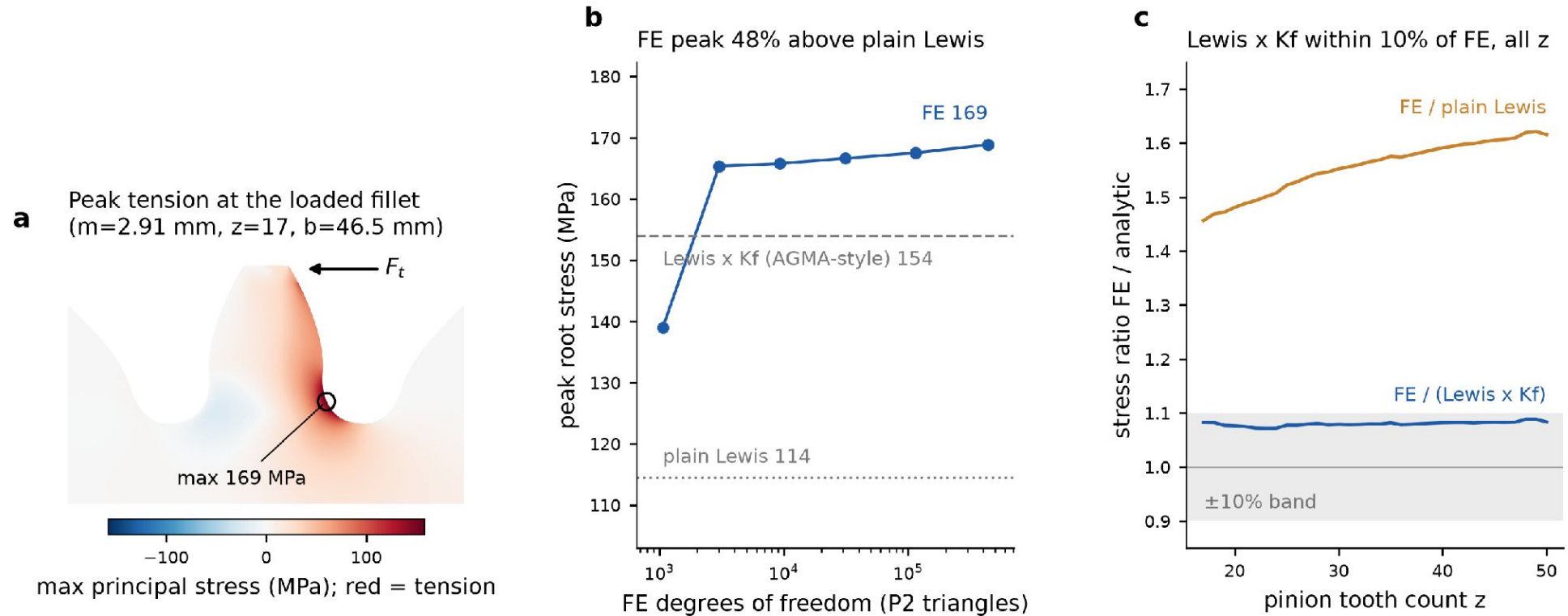
Where to spend time and money on parallelism:

Multi-fidelity orchestrator-worker loop: agents propose and verify; the solver fleet is plain parallel compute



- Three layers: **model** (batching, parallel reasoning) · **agent** (multiple LLM contexts) · **tool / compute** (simulations on a cluster: **usually dominates wall-clock**)
- The solver, not the LLM, is the bottleneck: keep the queue full, cache, deduplicate, **screen with cheap models first**

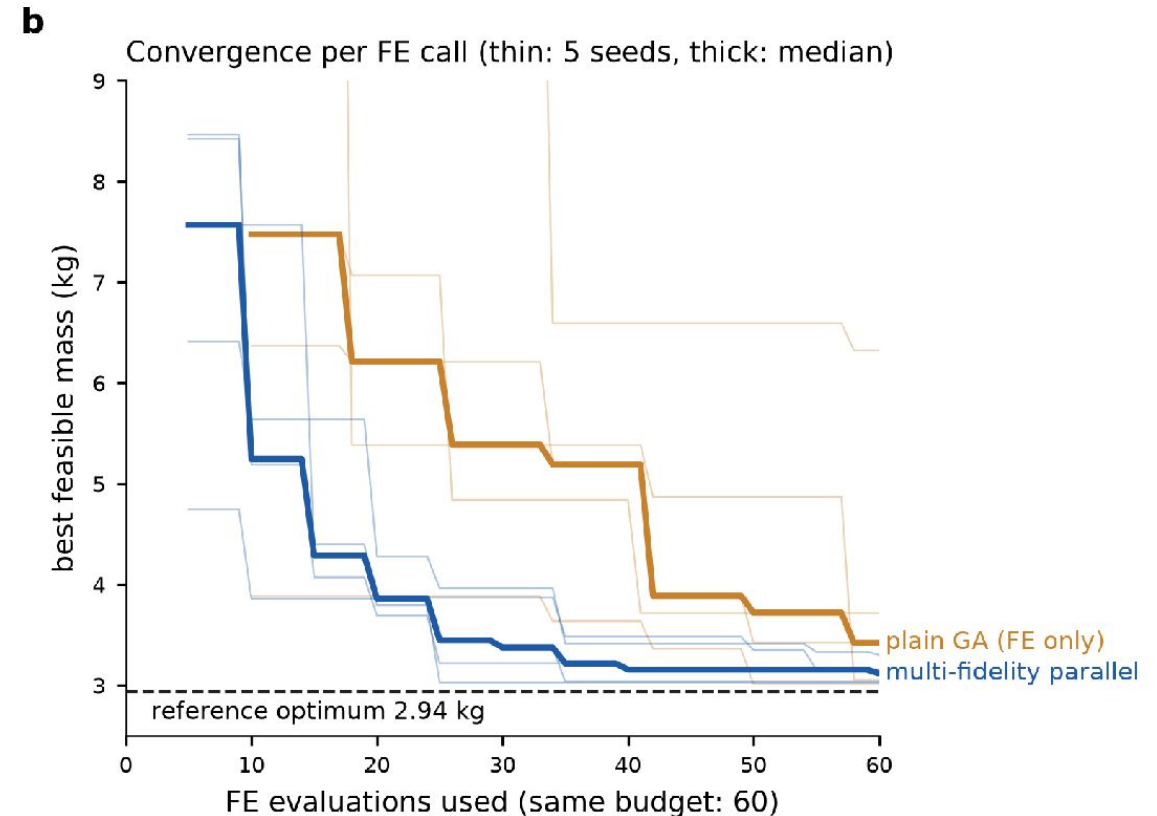
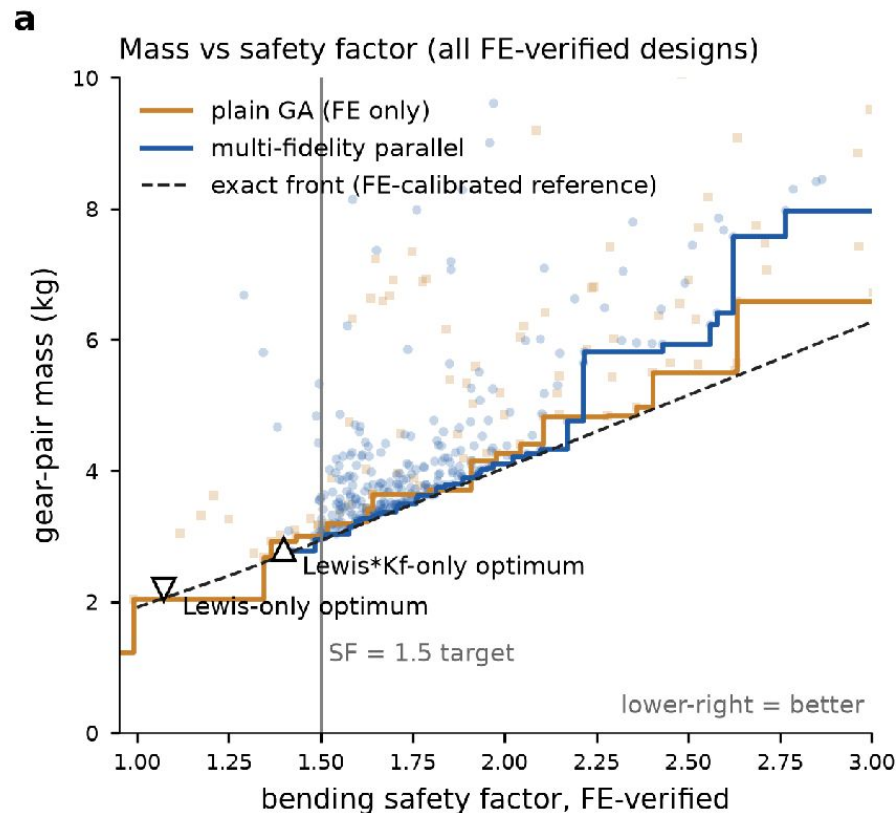
Demo (1/3): gear pair, two evaluators



- Minimize gear-pair mass: **100 N·m**, ratio **2**, $S_t = 250$ MPa, $SF \geq 1.5$. Lewis ($\pm K_f$, $\sim \mu s$) vs. 2D FE (< 1 s)
- FE peak **46–62% above plain Lewis**; Lewis $\times K_f$ **7–9% unconservative**, learned online as $c \approx 1.08$

Companion notebook gear_demo.ipynb (scikit-fem); teaching demo, static tip load

Demo (2/3): multi-fidelity parallel vs. plain GA, same 60 FE calls



- Multi-fidelity: median **3.12 kg** (+6.1% above the 2.939 kg optimum) · plain GA: **3.42 kg** (+16.4%) · 5 seeds each
- Designs from an analytic model alone **fail FE**: Lewis-only SF 1.04–1.08; Lewis × K_f SF 1.40–1.42 (target 1.5)

demo_results.json · reference optimum from a parallel FE sweep over z ($m = 2.91$ mm, $z_1 = 17$, $b = 46.5$ mm)

Demo (3/3): lessons, and where an LLM would slot in

- **Parallelism buys wall-clock, not CPU efficiency:** 5 workers 10.8 s vs 1 worker 27.8 s (**2.6X**, not 5X); CPU 44 s vs GA 24 s
- **Multi-fidelity wins** at equal expensive-evaluation budget; **verification** caught SF 1.04–1.42 designs
- This loop is scripted: 0 LLM tokens. An agent would: write / repair the generator and FE setup; propose structural changes (profile shift, helix, material); judge when to trust the cheap model; write the report
- **Agents for proposal and verification; plain parallel compute for evaluation**

Part IV · Doing it right

A checklist for agentic engineering

Checklist (1/2): formulation and verification

- Write objective, constraints and acceptance criteria **before** the agent. **No checker, no scaling**
- Use the most executable representation (parametric code > mesh > image)
- Deterministic verifiers in the loop: geometry validity, clearance, units, convergence, mesh study, analytic cross-check (beam, Lewis / AGMA, Hertz)
- **Keep the verifier separate and hidden from the proposer (prevents reward hacking)**
- Final design: independent solver or hand calculation

Checklist (2/2): architecture, reproducibility, evaluation

- Estimate f , c , p and real diversity, then pick the topology. Default: single agent + parallel evaluations
- Centralize verification; avoid unverified peer-to-peer propagation (~17X)
- Log tokens, dollars, wall-clock, solver-hours; report **cost-normalized** results
- Sandbox every tool call; version artifacts; record lineage; human sign-off before fabrication
- Report pass@1 with pass@k and run-to-run variance; hold out problems; beat strong non-LLM baselines (CMA-ES, GA, SIMP)

Kapoor et al., AI Agents That Matter, 2407.01502 · Kim 2512.08296

Core reading

- Kim et al., *Towards a Science of Scaling Agent Systems*, arXiv:2512.08296
- Anthropic, *How we built our multi-agent research system* (2025); *Building effective agents* (2024)
- Brown et al., *Large Language Monkeys*, arXiv:2407.21787
- Kwa et al. (METR), *Measuring AI Ability to Complete Long Software Tasks*, arXiv:2503.14499
- Mudur et al., *FEABench*, arXiv:2504.06260; Son et al., *Self-Improving CAD Generation Agents with FEA as Feedback*, arXiv:2605.17448