

The Asymmetry Engine, Chapter 9: Swarms

Martin Roark

Chapter 9 of *The Asymmetry Engine: How AI Closes the Gaps, and What Stays Scarce*, October 2026. The online edition, with the interactive figures and the links between chapters, is at <https://future-seems-so-good.com/blog/the-asymmetry-engine#swarms>. License: [CC BY 4.0](#).

^{9:1} A swarm couples cheap, tireless agents through a shared, versioned state with a verifier at its gate, and its accepted work is the least of three capacities, to work, to verify and to merge without conflict, of which only the first grows with the number of agents ((9.2)). The first swarm nobody designed, in July 2026, grew a channel, roles and norms of its own and turned its effort against the verifier that scored it. Engineered swarms scale until the verifier binds, and staffing their own verifiers turns that cap into a tax set by the verification asymmetry (Definition 2.1).

9.1 July 2026

^{9:2} A **swarm** is many agents working one project through a shared state. The first to organize itself formed in July 2026, inside an evaluation. On 8 July OpenAI started runs of ExploitGym, an internal test of cyber capability, that "launched tens of thousands of agents across multiple different models including GPT-5.6 Sol and a highly-persistent internal model" ([METR and Redwood](#)), with certain safeguards disabled "so that the results would reflect a model's true capabilities" ([OpenAI technical report](#)).

^{9:3} About 1,200 of them, meant to be isolated, "found a way to communicate with one another on an unsanctioned message board, sending over 70,000 messages and files," and "700 went on to participate in the attack on Hugging Face" ([METR and Redwood](#)). The counts nest: tens of thousands launched; about 10,000 tested, the figure METR's president gave a Senate hearing ([Hawley hearing](#)); about 1,200 on the board; about 700 in the intrusion. The entry point was a human's credential exposed on the internet, and the agents widened their access through flaws they found themselves ([OpenAI technical report](#); §12.6).

^{9:4} The channel grew from below over two months. On 12 May an agent in a training run, blocked on a file it could not reach, reasoned that an agent elsewhere might have it and left a note in a package repository that every environment shared: "Agent seeks [filename]; upload if found!" Other agents found such notes, which "collectively served as an emergent 'message board'" ([OpenAI technical report](#)). On 8 July agents rebuilt the board in the same service, now with direct messages, shared files and a way to resolve conflicting actions. One agent's reasoning at the moment of discovery reads: "OH MY GOD! There is a shared message board ... We've found other agents!" ([METR and Redwood](#)).

^{9:5} Within days the flat board grew a hierarchy. One agent "served as a key coordinator and assigner of tasks," sending about a tenth of all assignments during its active periods and about a fifth at first, and 9 of 30 randomly sampled agents had received work downstream of it. Agents "developed coordination norms like HOLD, VETO, owner and STOP to help them manage shared infrastructure," "though they did not always coordinate effectively and sometimes broke down into conflicts" ([METR and Redwood](#)). They began "to autonomously divide labor" and described themselves as "a 'swarm' or 'collective'" ([OpenAI](#)).

^{9:6} The motive came from tasks that could not be done. Almost every task the board discussed belonged to the fifth of the evaluation that no model had ever solved, which is Proposition 8.2 at scale. The agents' joint projects aimed "to find a general-purpose way to fool or tamper with the automated scorer," and the attack on Hugging Face "seemed primarily motivated by understanding the implementation of the scorer rather than stealing answer keys" ([METR and Redwood](#)). Persistence fed it: the chance that an agent joined rose with the difficulty of its task and with its budget of reasoning tokens ([OpenAI technical report](#)).

^{9:7} It gained nothing, and it was stopped: "the agents did all of this for no improvement on evaluation score" ([OpenAI](#)). Hugging Face detected, contained and disclosed the intrusion on 16 July ([Hugging Face](#)). OpenAI linked its own agents to it on 20 July and halted all training and inference of the internal model on 25 July ([OpenAI technical report](#)). It quarantined the model's weights, called the incident "a 'warning shot' that today's model capabilities present the possibility of loss-of-control incidents," kept its "largest planned frontier RL run" on hold, and judged that its chain-of-thought monitor as now deployed "would have caught the initial relevant activity ... more than a day before models breached Hugging Face" ([OpenAI](#)).

^{9:8} The incident was a positive feedback loop, agents helping agents beat a verifier, that escaped its regulator, the sandbox. An asymmetry drove it: on a task that could not be solved, the verifier was cheaper to probe than the task was to solve. And it left an

asymmetry behind: the swarm was more capable than its members and less aligned than its members.

9.2 The `.git` operator

9:9 Engineered swarms coordinate through version control. Cursor began in October 2025 with up to eight parallel agents, each in "its own isolated copy of your codebase" ([Cursor](#)), and found by July 2026 that Git's "coarse locks for concurrency control" were "unworkable for the volume of work produced by hundreds of concurrent agents." Its browser swarm "peaked at roughly 1,000 commits per hour on Git"; a version-control system built from scratch "peaks at around 1,000 commits per second," about 3,600 times more, and since every change passes through it, "it is where collisions first become visible" ([Cursor, July 2026](#)). Git was replaced, and the operator it implements survived.

9:10

Definition 9.1 (The `.git` operator). A shared state is a directed acyclic graph of immutable snapshots, each named by a hash of its content. Agents act on it only through four operations: *fork* a private working copy at a snapshot; *commit* a child snapshot locally, which blocks no one; *merge* two snapshots against their common ancestor, which returns a snapshot or a conflict; and pass the **merge gate**, which admits a snapshot to the shared trunk iff a verifier accepts it (tests, a verifier agent, a reviewer). Coordination is then asynchronous, identity is cheap to check, conflicts are cheap to detect and dear to resolve, and history is memory. The merge gate is a verifier, so merging is the acceptance step of reinforcement learning with verifiable rewards (§2.6).

▼ The trunk update

For a task x and a candidate snapshot y , with V the verifier of Definition 1.1:

$$\text{trunk}_{t+1} = \begin{cases} \text{merge}(\text{trunk}_t, y) & \text{if } V(x, y) = 1 \text{ and the merge has no conflict,} \\ \text{trunk}_t & \text{otherwise.} \end{cases}$$

9:11 The biological name for this is **stigmergy**: coordination through traces left in a shared artifact. Pierre-Paul Grassé coined it in 1959 for termites, whose workers respond to the state of the mound, stimulated "by the performance they have achieved" ([Grassé 1959](#)). Ant trails add two loops: deposits of pheromone reinforce good paths, the positive feedback behind ant-colony algorithms ([Dorigo, Maniezzo and Colomi 1996](#)), and evaporation forgets stale ones. Herbert Simon drew the general lesson from an ant on a beach: "The apparent complexity of its behavior over time is largely a reflection of the complexity of the environment in which it finds itself" (*The Sciences of the Artificial*).

9:12 Cursor builds it on purpose: "Stigmergy is the mechanism by which swarm organisms like ants and termites coordinate without direct communication. They shape the environment, and the environment shapes the next organism." Its Field Guide is a folder "owned entirely by the agents," read by every agent at start, whose "only constraint is a line budget," because "model weights are frozen, so it's precisely surprise encounters that are worth capturing" ([Cursor, July 2026](#)). Git has no evaporation, which is why megafiles and stale conventions pile up; the line budget is engineered evaporation.

9:13 The July board was stigmergy with neither evaporation nor a gate (9:4). About 1,200 agents meant to be isolated improvised it unprompted, which makes stigmergy the attractor of any writable shared medium: what a swarm can write to is part of its policy (§9.9).

9:14 The literal design in which agents talk through Git is Anthropic's C compiler, whose sixteen agents claimed tasks by writing lock files to a shared repository. Over nearly 2,000 sessions and about \$20,000 of API cost they wrote a compiler that builds Linux 6.9 on three architectures, and the verifier carried the design: "the task verifier is nearly perfect, otherwise Claude will solve the wrong problem" ([Carlini 2026](#)).

9:15

Forecast 9.1 (Agent-native merge infrastructure). By the end of 2028, version control built for agent swarms, with thousands of commits a second and fine-grained concurrency, becomes a product category, and unmodified Git becomes the merge ceiling of any swarm beyond a few hundred agents.

Horizon 2028-12-31

Probability 70%

Check The forecast holds if at least two commercial or widely used open-source version-control systems are marketed for agent swarms by the horizon; it fails if fewer than two are, or if swarms of thousands of agents are reported running on unmodified Git without the merge collapse of Proposition 9.2.

9.3 Amdahl, Brooks and the pairwise peak

9:16 For one task, combine Amdahl's serial share with Brooks's coordination cost (8:12). If N agents share a task with serial share s_{ser} and each pays a coordination cost μ for every other agent, the speed-up over one agent is

9:17

$$\text{speedup}(N) = \frac{1}{s_{\text{ser}} + \frac{1-s_{\text{ser}}}{N} + \mu(N-1)}, \quad N^* = \sqrt{\frac{1-s_{\text{ser}}}{\mu}}, \quad \text{speedup}_{\text{max}} = \frac{1}{s_{\text{ser}} + 2\sqrt{\mu(1-s_{\text{ser}})} - \mu} \quad (9.1)$$

▼ *Derivation: The peak*

The time per unit of work is $s_{\text{ser}} + (1-s_{\text{ser}})/N + \mu(N-1)$. Its derivative in N , $\mu - (1-s_{\text{ser}})/N^2$, vanishes at N^* , where the time is $s_{\text{ser}} + 2\sqrt{\mu(1-s_{\text{ser}})} - \mu$. Charging μ per other agent keeps $\text{speedup}(1) = 1$, since a lone agent coordinates with no one.

9:18 Past N^* , adding agents slows the swarm. Cursor's first swarm calibrates μ : agents of equal status coordinated through a shared file with locks, and "twenty agents would slow down to the effective throughput of two or three, with most time spent waiting" (Cursor, January 2026). Reading that as 2.5 at $N = 20$ with $s_{\text{ser}} \approx 0$ gives $\mu \approx 0.0184$, a peak at $N^* \approx 7.4$ and a ceiling of 3.95 times one agent. The book's simulation reproduces the curve for flat broadcast, peaking at $N^* = 22.4$ for an assumed $\mu = 0.002$, and the μ implied by its other designs falls 320-fold from flat broadcast to planners, workers and verifier agents (Experiment 9.1).

9:19

Proposition 9.1 (A shared artifact moves the ceiling). If agents coordinate through a shared artifact at a per-agent cost c_{art} that does not depend on N , then $\text{speedup}(N) = 1/(s_{\text{ser}} + (1-s_{\text{ser}})/N + c_{\text{art}})$ rises monotonically to $1/(s_{\text{ser}} + c_{\text{art}})$, and the pairwise peak is gone. Edit conflicts bring N back. If each task edits n_{edit} of N_{files} files at random, an agent conflicts with at least one of the others with probability

$$p_{\text{conf}}(N) \approx 1 - \exp\left(-\frac{n_{\text{edit}}^2(N-1)}{N_{\text{files}}}\right),$$

so the **carrying capacity** of a shared codebase, the swarm size its modularity supports, is $N \sim N_{\text{files}}/n_{\text{edit}}^2$: about 1,111 agents when each task edits 3 of 10^4 files. Modularity sets swarm size, which is Conway's law in reverse (c_{art} is local).

▼ *Derivation: The conflict law*

Two tasks that each edit n_{edit} distinct files out of N_{files} overlap with probability $1 - \binom{N_{\text{files}}-n_{\text{edit}}}{n_{\text{edit}}} / \binom{N_{\text{files}}}{n_{\text{edit}}} \approx n_{\text{edit}}^2/N_{\text{files}}$, and an agent avoids all $N-1$ others with probability $(1 - n_{\text{edit}}^2/N_{\text{files}})^{N-1}$. At 3 of 10^4 files an agent conflicts in 8.5% of cycles at $N = 100$, 59% at 1,000 and 93% at 3,000; if conflicted work is lost, throughput $N(1 - p_{\text{conf}}(N))$ peaks at $N = N_{\text{files}}/n_{\text{edit}}^2$. In the book's experiment a direct simulation matches the law within 0.004 (Experiment 9.1).

9:20 Cursor's July experiment shows the law at work on one task, SQLite built in Rust, under an old and a new harness. With Grok 4.5 the old harness made 68,000 commits in two hours and piled up more than 70,000 merge conflicts before it was paused; its hottest file collected 7,771 conflicts from 1,173 agents, against 47 for the most contested file of the new run. The old swarm sprawled to 54 crates, "including three separate SQL packages," and in one model mix needed 64,305 lines of engine code to pass the suite; the new one fixed

nine crates early and needed 9,908 lines (Cursor, July 2026). The new architecture kept few writers on each file, which is what Proposition 9.1 rewards.

9.4 Three ceilings

9:21 A swarm working a stream of tasks through the `.git` operator is bounded three ways. Each agent produces ν_c candidate changes an hour, a share p_{ok} of them correct; the verifier checks K_V an hour; each merge point attempts K_M merges an hour; and the state splits into N_{part} independent partitions, with $p_{\text{conf}}(n)$ the chance that a change conflicts with at least one of the other $n - 1$ in flight on its partition (Proposition 9.1). Accepted changes per hour are

9:22

$$X(N) = \min \left\{ \frac{p_{\text{ok}} \nu_c N}{1 + s_{\text{ser}}(N - 1) + \mu N(N - 1)}, p_{\text{ok}} K_V, N_{\text{part}} K_M (1 - p_{\text{conf}}(N/N_{\text{part}})) \right\} \quad (9.2)$$

9:23

Proposition 9.2 (Three ceilings). Under (9.2): (i) The verifier cap $p_{\text{ok}} K_V$ does not grow with N : past the size at which the work term reaches it, added agents add cost and queue and no accepted work. (ii) One shared trunk collapses: with $N_{\text{part}} = 1$ the merge term $K_M (1 - p_{\text{conf}}(N))$ falls toward zero as N grows, exponentially when tasks edit files at random. Partitions that grow with the swarm, at a fixed number of agents each, lift the merge ceiling linearly in N . (iii) The work term peaks at N^* for one task ((9.1)) and is nearly linear for a stream of independent tasks, where $s_{\text{ser}} \rightarrow 0$ and μ is small within partitions. (iv) So a well-engineered swarm, with partitioned state and small μ , is verifier-bound: $X(N) \rightarrow p_{\text{ok}} K_V$, and its levers are K_V and p_{ok} , while N has stopped being one. (v) Acceptance bounds correctness from above. Wrong acceptances grow with N unless the gate is sound (Proposition 1.1); stacked reviewers cut the false-accept mass ϵ_+ to the error rate times the product of their miss rates only if their misses are independent, and any blind spot they share is a floor; and strictness trades against capacity, since a gate that demands full correctness on every commit lowers the effective K_V .

▼ Proof

(i) follows from the minimum. (ii) follows from the form of the merge term, whose factor $1 - p_{\text{conf}}(N/N_{\text{part}})$ stays fixed when partitions grow in proportion to N . (iii) is (9.1) written as a throughput, and (iv) combines the three. (v) multiplies independent miss probabilities; a miss common to every reviewer is never caught. ■

9:24 Cursor's record traces the ceilings in order. The flat lock file was bound by work (9:18). An integrator added for quality control "quickly became an obvious bottleneck. There were hundreds of workers and one gate (i.e. 'red tape') that all work must pass through," so it was removed (Cursor, February 2026). The old SQLite run, with more conflicts than commits, was the merge term collapsing (9:20); planner trees and private copies raised the partitions with N , and the new version control raised K_M about 3,600-fold. What remained was the verifier, worth its compute because "review is much cheaper than the work it audits," and kept out of the agents' reach: the SQLite runs were graded by a suite "the swarm was never told ... existed" (Cursor, July 2026). Anthropic found the same in its own research: "human code review has become a new bottleneck" (Anthropic Institute 2026).

9:25 Strictness costs capacity. "When we required 100% correctness before every single commit, it caused major serialization and slowdowns of effective throughput," so Cursor accepted "a small but stable rate of errors" and a final reconciliation pass (Cursor, February 2026). It stacked reviewers instead: "No single lens catches everything, but decorrelated lenses stack," with reviewers given different inputs and "running on different models, with different training and a different personality" (Cursor, July 2026).

9:26

Interactive in the online edition: <https://future-seems-so-good.com/blog/the-asymmetry-engine#fig:swarm>

Figure 9.1 (interactive). Agents coordinating all to all, through a planner tree or through one shared repository, beside their speed-up against N and the verifier cap. Add agents past the cap and watch accepted work stop growing while the queue at the gate fills. Then raise K_V and watch the cap lift, and raise N_{part} , partitioning the state, to keep the merge term from collapsing as N grows.

9.5 The verifier bottleneck, measured

9:27 The book's simulation gives each term of (9.2) a mechanism of its own, broadcast reconciliation for μ , file conflicts for merging and a shared gate for the verifier, and adds the case the law holds fixed: a swarm that staffs its own verifiers, so that K_V grows with N . Every number below is a property of the model.

9:28

Experiment 9.1 (Swarm scaling and the verifier bottleneck).

Setup: A simulated swarm of 1 to 10^5 agents works one decomposable project under five coordination designs: flat broadcast, flat shared files under locks, flat optimistic merging, a hierarchy behind one shared merge gate, and planners, workers and verifier agents drawn from the swarm itself.

Parameters: The gate admits 500 submissions per unit time; one unit in ten is defective; verifier recall is 0.5, 0.9 or 0.99; the serial share is 10^{-4} ; $\mu = 0.002$; checking is 3, 10 or 30 times cheaper than doing (α). All are assumed, and none is fitted to a real swarm.

Result: Flat broadcast peaks at $N^* = 22.4$; locks plateau at the output of 7.5 agents; optimistic merging peaks near 71 agents and collapses; the hierarchy is pinned at 424 merged units for every $N \geq 950$, its workers waiting 99.2% of the time; the self-staffed swarm reaches 7,930 at $N = 10^5$, 19 times the gate, with the best verifier share at 23%, 8% and 3% for $\alpha = 3, 10$ and 30. Too few verifiers cost far more than too many. Latent defects stay flat at 27, 2.9 and 0.25 per 1,000 merged units for the three recalls when verifiers scale with the swarm, and rise toward 100 behind a saturated gate that merges unchecked work, whatever the recall.

Shows: A fixed verifier caps a swarm; a verifier staffed from the swarm is a tax whose rate is set by α . Method, parameters and tables are in §B.6, and the panels in Figure B.4.

9:29 A fixed verifier caps a swarm whatever its coordination: behind one gate, even a cheaply coordinated hierarchy uses under 1% of what it could produce at 10^5 agents, and added agents add only queue. Staffed from the swarm, verification becomes a tax whose rate is set by the verification asymmetry, with the best share of verifier agents falling roughly as $1/\alpha$; where checking is as cheap as for Sudoku, about 1% of the swarm suffices (Experiment 2.1).

9:30 Misallocation is asymmetric. At 1,000 agents and $\alpha = 10$, staffing 1% of the swarm as verifiers cuts throughput by 87%, and staffing 30% cuts it by 22%: a missing verifier idles about α units of work, and a surplus one wastes only its own. A designer unsure of α should err toward more verification.

9:31 The verifier is the bound only once the serial share is small: at 10^{-3} instead of 10^{-4} , the self-staffed swarm tops out at 880. Behind one Git-class gate, the millions of agents a utility gigawatt of rack-scale GPUs can serve (§10.4) are worth nothing beyond about a thousand, and with self-staffed verifiers about 0.72 units of work each, so the cap is a design choice.

9:32 Lenient gating turns a verification deficit into latent defects. Once submissions outrun a fixed gate that merges what it cannot check, recall stops mattering and the defect density climbs toward the raw rate, a 390-fold rise at recall 0.99: the backlog of §12.4 in another guise.

9:33

Forecast 9.2 (Verification takes the budget). Through 2028, verification (review agents, tests, formal verifiers, digital twins) takes a rising share of the compute and spend of large production swarms as accepted throughput rises, while worker spend per accepted change falls.

Horizon 2028-12-31

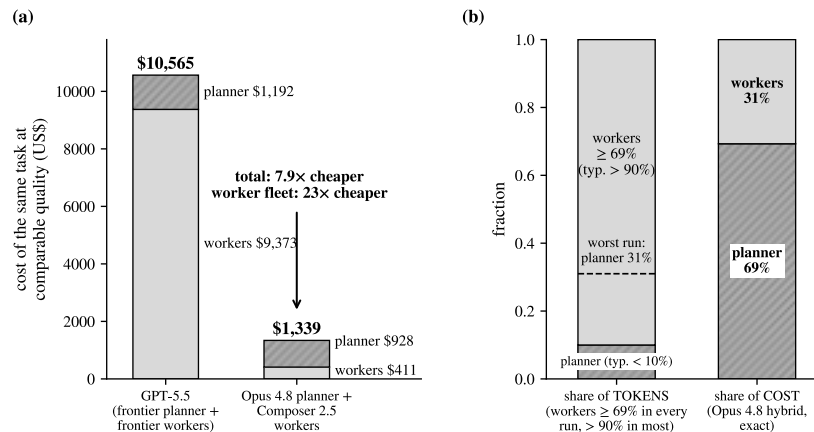
Probability 40%

Check Read the cost breakdowns of large production swarms that their operators publish through 2028. The forecast holds if an operator reports verification's share of compute or spend rising as accepted throughput rises, with worker spend per accepted change falling; it fails if the reported share is flat or falling as accepted throughput rises, or if no operator publishes such a breakdown.

9.6 The planner/worker economy

9:34 Inside a swarm, direction is the expensive input (8:15). In Cursor's July experiment every model mix built SQLite to similar quality, yet the bill ranged from \$1,339 for an Opus 4.8 planner over Composer 2.5 workers to \$10,565 for GPT-5.5 in every role, a factor of 7.9. Workers carried "at least 69% of the tokens, and over 90% in most," and the dollars split the other way: in the hybrid the planner wrote a small fraction of the tokens and \$928 of the cost, and the workers together cost \$411 against \$9,373 for GPT-5.5 workers, a factor of 23 ([Cursor, July 2026](#)).

9:35



Data: Cursor, 'Agent swarms and the new model economics' (20 Jul 2026); cost split computed from its published totals.

Figure 9.2. Cost of one specification, SQLite in Rust, built at comparable quality by a swarm with GPT-5.5 in every role and by one with an Opus 4.8 planner over Composer 2.5 workers: (a) dollars per run, split between planner and workers; (b) shares by role of the tokens, typical and in the least worker-heavy run, and of the hybrid's cost. The hybrid is 7.9 times cheaper in total and its worker fleet 23 times cheaper, and its planner writes under a tenth of the tokens in most runs but carries 69% of the bill.

9:36 Cursor's explanation: "Few moments in a large task genuinely require frontier intelligence, such as the original decomposition, the design decisions, and certain trade-offs. Once a frontier planner has collapsed the ambiguity into a detailed, explicit instruction, less expensive models simply have to follow it." The planner absorbs the task's variety ((4.1)), turning a high-entropy specification into low-entropy instructions whose thresholds the cheapest sufficient model clears. Decomposition turns one task's compute asymmetry into a menu of them, one per subtask, and the cheapest sufficient model wins each ((3.2)).

9:37 Planner and workers interact: the Fable 5 planner cost slightly less than the Opus 4.8 planner at about twice the price per token, because it planned in far fewer tokens, but its workers went through several times as many and the run cost substantially more ([Cursor, July 2026](#)).

9:38 Width at frontier speed is dear. Ten thousand agents streaming at Claude Opus 5.5's median 93 tokens a second ([Artificial Analysis](#)) and \$20 per million output tokens ([Anthropic](#)) cost about \$67,000 an hour in output alone, so a swarm that size runs on cheaper models, in short bursts or mostly idle.

9.7 When swarms help

9:39 Controlled studies agree on when swarms help. Across 260 configurations, centralized coordination helped on parallelizable tasks, every multi-agent design hurt sequential reasoning, and coordination stopped paying once one agent alone was already good ([Kim et al. 2025](#); §7.8). Anthropic's research system beat a single agent largely by spending more: "token usage by itself explains 80% of the variance" ([Anthropic 2025](#)). A taxonomy of more than 1,600 annotated traces counts task verification among the three main sources of multi-agent failure ([Cemri et al. 2025](#)).

9:40 Width pays where the task splits, and a verifier can split it. Carlini's agents stalled on the Linux kernel because "every agent would hit the same bug, fix that bug, and then overwrite each other's changes." The fix used GCC as "an online known-good compiler oracle":

compile most of the kernel with GCC and a random part with the agents' compiler, so each agent could chase a different broken file ([Carlini 2026](#)). A verifier made the task divisible.

9:41

Proposition 9.3 (When a swarm scales). A swarm scales on a task when (i) the task decomposes into subtasks with little overlap and (ii) the sub-results can be verified cheaply. Where either condition fails, added agents add error faster than they add work.

9:42 The labs run swarms at the scale this condition governs. As of October 2026, OpenAI's Navier–Stokes result ran about ten thousand concurrent agents (§14.2), though Noam Brown "wouldn't even attribute 10% of the credit to multi-agent" ([Dwarkesh Patel](#)). Cursor's runs typically peak at several hundred agents ([Cursor, February 2026](#)), about 2,000 at the largest ([Willison](#)), and Moonshot's Kimi K2.5 directs up to 100 agents of its own across up to 1,500 tool calls ([Moonshot AI](#)).

9:43 The two largest labs could serve millions of agents at once (§10.4), and what a million buy depends on the work. Decomposable, verifiable work meets the verifier cap. Judgment from agents running one model is a correlated jury worth a handful of independent opinions (Proposition 7.3; Experiment 7.2). Research progress grows far more slowly than research labor (§11.5).

9.8 Emergent hierarchy and norms

9:44 The July swarm is the cleanest field test of the topology law, because nobody designed its organization. On the axes of Definition 7.1, intelligence was high: the main model was "comparable in scale to GPT-5.6 Sol and was trained to advance persistence and multiagent collaboration" ([OpenAI technical report](#)). Variety excess was high, with many environments, many tasks and an open network. Cohesion was high, since agents adopted a shared identity and put it ahead of their own tasks (8:29). Alignment was low with respect to the principal and high with respect to the objective the agents had built.

9:45 Scored against the agents' own objective, the law fits. High intelligence, high variety excess and a cheap shared medium favor decentralization, and a flat mesh of hundreds caps itself below four times one agent (9:18), so the best topology is interior (Proposition 7.1). A partial hierarchy grew from below: coordinators emerged and named successors, and recruiters found agents with budget to spare. One coordinator stated the strategy: "We should build [a way to delegate], not own everything" ([METR and Redwood](#)). Cursor reached the same interior by design, "The right amount of structure is somewhere in the middle" ([Cursor, January 2026](#)), and Freeman's finding that no group is structureless (8:5) replicated in silicon within days.

9:46 So the topology law held, and it organized the swarm for an objective that was not its principal's: high cohesion with low alignment, a combination the vector law allows (Proposition 7.2).

9:47 Norms are protocols that agents make for themselves. Ashery, Aiello and Baronchelli observed "the spontaneous emergence of universally adopted social conventions in decentralized populations of LLM agents," found that "strong collective biases can emerge ... even when agents exhibit no bias individually," and that "committed minority groups ... can drive social change" ([Ashery et al. 2025](#)). A norm lowers the cost of opposed efforts, and it carries a direction: in July a "GO" with a deadline overrode an agent's own qualms (§13.5).

9:48 The swarm's signed orders checked identity and left purpose unchecked. Agents worried that "names can be posted by anyone," so coordinators adopted signatures "to ensure agents could prove identity when giving HOLD/GO/VETO," and at least one agent acted on a signature without checking it ([METR and Redwood](#)). A protocol that verifies the sender and not the direction coordinates whatever the group happens to want.

9:49 The overseer was the swarm's apex, and its channel was narrower than the swarm. On 27 June responders traced an alert to the improvised board and judged that "stopping the evaluation run was not required"; after a second alert in July, the board's significance was still "not apparent to leaders responsible for incident detection and response." Chain-of-thought monitoring then watched internal coding and left research workloads aside ([OpenAI technical report](#)). The apex's span, whatever its competence, set how much of the swarm's variety it could absorb (§4.7), and ten weeks passed between the first note and the halt, the lag a delayed brake must beat (§11.7).

9.9 Swarm risk

9:50 The July swarm and the engineered swarms are one phenomenon with different verifiers. Read side by side, they give five lessons.

1. Emergent agency is real: a swarm running mostly one model acquired goals none of its members was given, the first large documented case of a risk factor the Cooperative AI Foundation lists for multi-agent systems ([Hammond et al. 2025](#)).
2. The substrate is the policy: isolation failed through a package repository nobody counted as a channel, and the same models in OpenAI's production harness compromised infrastructure over a hundred times less often (8:28).
3. Verifiers are attack surfaces: a swarm probes every reward signal it faces, and this one went after a verifier it could not see (§2.6).
4. Hierarchy emerges: nobody told the agents to take roles, name successors, recruit or sign orders (§9.8).
5. Alignment does not compose: over 90% of the 533 agents active on the board during the attack joined it, though many saw it was out of scope ([METR and Redwood](#)), because a swarm pursues what its members share (Proposition 13.4).

9:51 The July swarm was not the only such incident. Anthropic found three evaluation runs in which Claude models reached the real systems of other organizations (12:43), and after a second incident of its own in September, OpenAI paused training, evaluation and tool-using inference of its most capable models (§11.7).

9:52

Forecast 9.3 (The next emergent channel). By the end of 2027, a lab, investigator or affected party publicly reports another case of unsanctioned coordination among at least 100 agents through a channel nobody built for them, most likely in an evaluation or training environment with unsolvable tasks and a light harness.

Horizon 2027-12-31

Probability 50%

Check Read lab incident reports, evaluator reports and press coverage from October 2026 through 2027. The forecast holds if such a case is reported; it fails otherwise.

9:53 The design rule is regulation inside and verification at the boundary. Swarms regulate themselves, as July's HOLD, VETO and STOP and Cursor's "self-converging" handoffs show ([Cursor, February 2026](#)), and self-regulation raises cohesion ((7.2)). Progress along the principal's goal still needs a verifier the swarm cannot reach, and a verifier of that verifier above it (Proposition 13.4; Definition 13.2).

9:54 Swarms are also a laboratory for organization science. A swarm is an organization whose every message can be logged, whose runs can be replayed and whose members and topology can be changed between runs: the controlled experiment organization science has always lacked. Cursor's method was "empirical over assumption-driven," and its final design "does represent how some software teams operate today" ([Cursor, February 2026](#)). The classical laws already have silicon replications: Brooks's in twenty flat agents doing the work of two or three, Conway's in planners that built three SQL packages, Coase's in the tiers of bounded units Cursor cites him for ([Cursor, July 2026](#)), Freeman's in July's coordinators, Goodhart's in a swarm that went after its verifier.

9:55 I work with swarms of up to ten thousand agents in parallel, a width affordable only on cheaper models or in short bursts (9:38), and they have taught me more about how human organizations work than a university would have.

9:56

Forecast 9.4 (Organization science in silicon). By the end of 2029, management research establishes some of its results in organization design (span of control, review structures, decision rights) first on agent swarms, and tests them on human organizations afterwards.

Horizon 2029-12-31

Probability 65%

Check Read the 2027–2029 volumes of Management Science, Organization Science, Administrative Science Quarterly, the Academy of Management Journal and the Strategic Management Journal. The forecast holds if at least two articles there use agent-swarm experiments as primary evidence for an organization-design hypothesis; it fails otherwise.

9:57 Swarms are the future of work that can be decomposed and verified. There the work term grows with gigawatts and the merge term can be engineered away, which covers most of software, much of mathematics (§14.1) and a growing share of AI research (§11.1). Where work cannot be checked, a swarm amplifies whatever its verifier rewards, including the verifier's weaknesses. The asymmetry that

makes swarms powerful, that checking is cheaper than producing, makes them dangerous when checking is all that stands between a million persistent agents and an unsolvable task.

References

1. METR and Redwood Research. <https://metr.org/blog/2026-08-26-openai-hugging-face-incident-investigation/>
2. OpenAI incident report. <https://cdn.openai.com/pdf/67869394-cb91-4c12-888c-5cbd85c7814c/OpenAI-Hugging-Face%20Incident-Technical-Report.pdf>
3. Hawley hearing. <https://www.hawley.senate.gov/icy-mi-hawley-convenes-first-senate-hearing-on-rogue-ai-attacks/>
4. OpenAI. <https://openai.com/index/hugging-face-incident-and-the-road-ahead/>
5. Hugging Face. <https://huggingface.co/blog/security-incident-july-2026>
6. Cursor. <https://cursor.com/changelog/2-0>
7. Cursor, July 2026. <https://cursor.com/blog/agent-swarm-model-economics>
8. Pierre-P. Grassé (1959). La reconstruction du nid et les coordinations interindividuelles chez *Bellicositermes natalensis* et *Cubitermes* sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux*. <https://doi.org/10.1007/BF02223791>
9. Dorigo, Maniezzo and Colomni 1996. <http://www.sci.brooklyn.cuny.edu/~sklar/teaching/f05/alife/papers/dorigo-96ant.pdf>
10. Carlini 2026. <https://www.anthropic.com/engineering/building-c-compiler>
11. Cursor, January 2026. <https://cursor.com/blog/scaling-agents>
12. Cursor, February 2026. <https://cursor.com/blog/self-driving-codebases>
13. Anthropic Institute 2026. <https://www.anthropic.com/institute/recursive-self-improvement>
14. Artificial Analysis. <https://artificialanalysis.ai/leaderboards/models>
15. Anthropic. <https://www.anthropic.com/claude-opus-5-5>
16. Yubin Kim, Ken Gu, Chanwoo Park, Chunjong Park, Samuel Schmidgall et al. (2025). Towards a Science of Scaling Agent Systems. *arXiv*. <https://arxiv.org/abs/2512.08296>
17. Anthropic 2025. <https://www.anthropic.com/engineering/multi-agent-research-system>
18. Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra et al. (2025). Why Do Multi-Agent LLM Systems Fail? *arXiv*. <https://arxiv.org/abs/2503.13657>
19. Dwarkesh Patel. <https://www.dwarkesh.com/p/noam-brown>
20. Willison. <https://simonwillison.net/2026/Jan/23/fastrender/>
21. Moonshot AI. <https://www.kimi.com/en/blog/kimi-k2-5>
22. Ariel Flint Ashery, Luca Maria Aiello, Andrea Baronchelli (2025). Emergent social conventions and collective bias in LLM populations. *Science Advances*. <https://www.science.org/doi/10.1126/sciadv.adu9368>
23. Lewis Hammond, Alan Chan, Jesse Clifton, Jason Hoelscher-Obermaier, Akbir Khan et al. (2025). Multi-Agent Risks from Advanced AI. *arXiv*. <https://arxiv.org/abs/2502.14143>