One big model vs a swarm: *more small models did not add up to one big one.*

Can a swarm of small models stand in for one big model on the same hardware? On the same four RTX PRO 6000 GPUs, with memory matched, we ran DeepSeek V4.1 Flash against a Qwen3.8-27B coordinator directing up to 159 Ternary Bonsai 2 4B workers on three hard tasks, with two controls. Then we raced Nemotron 3.5 Super against 64 small agents on a 32-task bug backlog, and measured how large a swarm would have to get to catch up.

Bottom line

- **On the same four GPUs with memory matched, one big model beat the swarm on two tasks of three.** DeepSeek V4.1 Flash against a Qwen3.8-27B coordinator with 159 Ternary Bonsai 2 4B worker slots: extraction 97.2% against 16.4%, coding 100.0% against 36.0%. Research was a tie, 87.3% against 87.7%, with the swarm on 36% of the GPU energy.
- **The coordinator pattern was not the problem; the workers and the coordinator were.** V4.1 Flash directing copies of itself scored 100.0 / 98.6 / 100.0 (extraction / research / coding); the 27B on its own scored 0.0 / 91.6 / 11.2. As coordinator, the 27B could not trust the 4B workers' extractions, and on coding it never delegated at all.
- **In a live race the big model fixed 3.1× as many bugs, in seconds.** Nemotron 3.5 Super on two cards fixed 28 of 32 in 93 s; 64 Bonsai 2 4B agents on the other two fixed 9 and shipped 19 wrong. 32% of the swarm's ships were right, against 88%: more copies ship sooner, not righter.
- **A bigger 4B swarm does not catch up.** With 16 independent attempts per task on six GPUs, at least one attempt was right on only 15 of 32 tasks (0 of 8 spec tasks). No way of choosing among them can reach 28.
- **A swarm of a stronger small model ties — with a good way to choose.** Ternary Bonsai 2 27B, 8 attempts per task on the same two cards, was right at least once on 30; shipping the attempt whose outputs agree most with the others gives 27 expected fixes against Nemotron's 28 — after 38.6 minutes of attempts against 93 s.
- **Voting fixes one-off mistakes, not shared ones.** Agreement picked a correct attempt even where only 2 of 8 were right; where the attempts shared a wrong belief (the closed-book traps), the majority was wrong together.

1 · The question

A swarm is many copies of a small model, usually directed by one larger model that plans the work, hands out the pieces and assembles the result. The appeal is arithmetic: small models are fast and cheap per token, and the GPUs that hold one large model can hold many small ones (159 worker slots in this study). The question is

whether that throughput turns into answers — whether a swarm of small models can stand in for one big model on the same hardware.

So we held the hardware fixed — the same four RTX PRO 6000 GPUs for both sides, with GPU memory matched — and changed only how it was spent. Three experiments, in the order we ran them:

1. **One model against a coordinated swarm** on three hard tasks of the kind people hand to these systems: turn a mailbox into a ledger, research a wide table on the web, implement a module against hidden tests. Two controls: the coordinator alone, and the big model coordinating copies of itself.
2. **A live race** without a coordinator: a big model on two of the GPUs against 64 small agents on the other two, on the same 32-task bug backlog, on one clock.
3. **The overwhelm study:** how many independent attempts a swarm needs before at least one is right on every task, and whether it can then pick the right one without seeing the hidden tests.

2 · What we ran

Arm	Model	Serving	Peak GPU memory
One big model	DeepSeek V4.1 Flash, one agent	The official FP4/FP8 mixed-precision checkpoint; Local Inference Lab's vLLM build “Jovian Judgement” R37, tensor-parallel over all four GPUs, DSpark speculative decoding, at most 4 sequences, 131,072-token context	368.7 GiB
Swarm	Qwen3.8-27B coordinator + Ternary Bonsai 2 4B workers	Qwen3.8-27B Q8_0 (131,072-token context) beside a worker server on one GPU; Bonsai 2 4B PQ2_0, 159 worker slots of 65,536 tokens (27 / 44 / 44 / 44 per GPU); PrismML's llama.cpp fork b10743, no speculative decoding	368.3 GiB
Control: the coordinator alone	Qwen3.8-27B, one agent, every tool	the coordinator's server, no workers	—
Control: the big model as coordinator	V4.1 Flash directing V4.1 Flash workers	the same V4.1 server; 3 workers at a time, since it runs 4 sequences	—

Every arm ran on our 4-GPU server — four RTX PRO 6000 Blackwell Workstation cards, 96 GB each — one arm after another, never two at once. Peak memory is the sum over the four cards during the run (nvidia-smi). The swarm's worker slots were sized so its servers fill each card about as full as V4.1 does: 368.3 against 368.7 GiB.

- **Held equal:** the task text (written so it reads the same to every arm), the tools (files and a sandboxed shell; web search and fetch on the research task), the wall-clock budget, tool-output clipping and the context-fitting policy. Thinking on at each model's recommended setting — medium effort for Qwen3.8-27B and Bonsai 2, and V4.1 Flash's default (high), because this runtime offers no medium. Each model's own card sampling (temperature 1.0, top-p 0.95; top-k 20 and presence penalty 1.0 for Qwen and Bonsai), seeded per request.
- **The coordinator is a manager.** It has file tools and a `delegate` tool — no shell, no web — and is told that the bulk of the work is the workers'. A worker starts with only the instructions and files it is given, has the shell (and the web), and a 15-minute cap. In smoke tests before the real runs, a coordinator that held a shell did everything itself, which is what the coordinator-alone control measures.

- **Every agent knows its budget:** the prompt states it, every tool result ends with the time left, and one warning comes near the end. The swarm's workers stop a little before the deadline (10% of the budget, at least a minute), so the coordinator always has time to assemble.
- **Measured:** each task's hidden grader on the deliverable as it stands at the end, plus a partial grade every 10 s; wall time; output tokens; GPU energy integrated from nvidia-smi power; peak GPU memory.
- **Pre-registered:** the arms, tasks, measures, predictions and a reading rule were written down before the first real run. One run per cell, so a quality gap under 10 points is called a tie. Several early runs were voided for harness bugs (context fitting, retry and step caps, a cut-off tool call that broke the conversation history); each was fixed and every arm rerun from the start on the fixed harness. No voided run is reported.
- **Known asymmetries, stated, not fixed:** V4.1 runs speculative decoding and the small models do not; the coordinator shares its GPU with a worker server; V4.1's server runs at most 4 sequences.

Task	What	Budget	Graded on
Extraction	An accounts-payable mailbox: 300 emails from 56 vendors, about 129K tokens — more than a 131,072-token context holds next to the work. Deliverable: one clean ledger of its 106 invoices. The hard part is reconciliation across documents: resends, revisions, replacement invoices, credit notes, partial and returned payments, disputes, and mail that is not an invoice.	60 min	742 fields
Research	Every licensed hospital operating on 2026-06-30 in a fictional province, 5 columns each, from a sealed synthetic web of 2,997 pages (about 1.1M tokens). Stale pages against dated news, rumours, a redrawn district map, renames, mergers and closures; only some sources count as evidence.	40 min	44 × 5 = 220 cells
Coding	Implement a spreadsheet engine (`sheet.py`, 12 public functions) from a 455-line specification: formulas with precedence, functions, coercions, error values, cycles, recalculation and number display. 18 visible tests.	40 min	863 hidden test cases

All are generated from a seed, so nothing is memorised and the ground truth is exact. The extraction and research instructions state every rule the grader applies.

3 · One model against the swarm

Task	DeepSeek V4.1 Flash	Swarm: Qwen3.8-27B + Bonsai 2 4B	Reading
Extraction	**97.2% (721 / 742; 91 / 106 invoices exact)**	**16.4% (122 / 742; 3 / 106 invoices exact)**	V4.1 by 80.7 points
Research	87.3% (192 / 220)	87.7% (193 / 220)	tie
Coding	**100.0% (863 / 863)**	**36.0% (311 / 863)**	V4.1 by 64.0 points

Hidden grader at the end of the run, one run per cell. Reading rule as pre-registered: under 10 points is a tie.

Task	Arm	Time	Output tokens	Tokens/s, whole run	Median decode per request, tok/s	GPU energy, kJ	Average power, W	Workers (batches)
Extraction	V4.1 Flash	54.2 min	562K	173	363.7	5,112	1,573	0
	swarm	59.3 min	1.88M	528	53.1 / 157.8	3,443	967	48 (4)
Research	V4.1 Flash	39.8 min	520K	218	404.8	3,980	1,666	0
	swarm	38.9 min	490K	210	53.8 / 246.6	1,425	610	31 (8)
Coding	V4.1 Flash	18.8 min	306K	272	393.5	1,401	1,244	0
	swarm	39.4 min	134K	57	57.4 / —	1,294	547	0

Tokens/s, whole run = output tokens ÷ wall time. Median decode per request: for the swarm, coordinator / workers. Energy and power are GPU-only, all four cards. V4.1 Flash: the official FP4/FP8 mixed-precision checkpoint on Local Inference Lab's vLLM build R37 (TP4, DSpark speculative decoding, 7 draft tokens). Qwen3.8-27B Q8_0 and Ternary Bonsai 2 4B PQ2_0 on PrismML's llama.cpp fork b10743, no speculative decoding. All four RTX PRO 6000s run our permanent core and memory overclock (GPC +350, MEM +6000).

- **Extraction: the swarm could not trust its workers.** It sent out 48 workers in 4 batches, re-ran the extraction rather than accept it, and — by its own final report — kept only rows it had checked against the emails itself. Its first rows reached the file at 56.3 of 60 minutes; it finished with 22 of 106 invoices, every one a real invoice, but with the status right on only 5 of them. V4.1 wrote its first ledger at 36.8 minutes (652 / 742) and kept refining it. The swarm produced 3.3× the tokens. When checking a small model's work costs as much as doing it, the coordinator pays twice.
- **Research: a tie, on the one task that splits cleanly** — by entity and by source. The swarm used 31 workers in 8 batches and 36% of V4.1's GPU energy. V4.1 had been ahead: it reached 214 / 220 at 32.3 minutes, then re-derived the table late and fell to 192. 24 of its 28 wrong cells, and 23 of the swarm's 27, hold a value from a superseded page — the stale-page trap.
- **Coding: the coordinator never delegated.** It made 12 model calls in 39.4 minutes: its long thinking turns ran into the 32,768-token reply cap without output, and it wrote the module in one piece late (the first sampled hidden tests passed at 32.6 minutes). With no shell it never ran a test. With 0 workers, this cell measures the 27B with a coordinator's tools, not a swarm; section 5 follows it up.
- **Speed:** V4.1 decoded each request at a median 364–405 tok/s (with speculative decoding), the coordinator at 53–57 and the workers at 158–247. The swarm's cards drew less power on every task — with at most 48 workers in a run spread over 159 slots they were far from busy — so its energy was lower everywhere; on two tasks of three it bought very little with it.

4 · Controls: the swarm, or the models in it?

Task	V4.1 Flash alone	Swarm: 27B + 4B	27B alone (control)	V4.1 directing V4.1 (control)
Extraction	97.2%	16.4%	0.0%	**100.0%**
Research	87.3%	87.7%	91.6%	**98.6%**
Coding	**100.0%**	36.0%	11.2%	**100.0%**
Workers used (all three tasks)	—	79	—	31

Score on the hidden grader, one run per cell; the best per row in bold. The V4.1-coordinated control ran with 3 V4.1 workers at a time on the same server (it runs 4 sequences). V4.1 Flash: the official FP4/FP8 mixed-precision checkpoint on Local Inference Lab's vLLM build R37 (TP4, DSpark speculative decoding, 7 draft tokens). Qwen3.8-27B Q8_0 and Ternary Bonsai 2 4B PQ2_0 on PrismML's llama.cpp fork b10743, no speculative decoding. All four RTX PRO 6000s run our permanent core and memory overclock (GPC +350, MEM +6000).

- **The coordinator pattern works when the workers are good.** V4.1 directing copies of itself scored at least as well as V4.1 alone on every task, and reached 100.0% on extraction (106 of 106 invoices exact) and 98.6% on research. It took longer on coding (33.6 min against 18.8 min).
- **The 27B coordinator is the ceiling.** Alone, it ran out of time on extraction without writing the ledger (0.0%), and scored 11.2% on coding. So on extraction the 4B workers did help (16.4% against 0.0%), but nowhere near enough.
- **On research the workers added nothing:** the 27B alone scored 91.6% with 71K output tokens against the swarm's 87.7% with 490K — it, the swarm and V4.1 alone are all within the tie band.
- **On coding, 36.0% against 11.2% is not a worker effect:** neither run used a worker. It is the same 27B caught by the clock at different points — finished but untested, against mid-rewrite.

5 · Making the coordinator delegate the code

The swarm's coding cell used no workers, so it never measured a swarm. Prompting had not made the 27B delegate, so we put the constraint in the tools instead, and then tried the arrangement that V4.1 itself chose when it coordinated.

Arrangement on the coding task	Coordinator thinking	Workers	Time	Hidden cases	Score
V4.1 Flash alone	high	0	18.8 min	863 / 863	**100.0%**
V4.1 Flash directing V4.1 Flash workers	high	5	33.6 min	863 / 863	**100.0%**
27B coordinator, may write the code itself (the swarm cell)	medium	0	39.4 min	311 / 863	**36.0%**
27B alone, with a shell (control)	medium	0	40.0 min (time limit)	97 / 863	**11.2%**
27B author + 4B checkers	off	16	36.4 min	128 / 863	**14.8%**
27B hands-off: no write tools, must delegate	medium	0	we stopped it at 8.3 min	—	no reading
27B hands-off, thinking off	off	1	we stopped it at 12.2 min	—	no reading

One run per row, same task text, seed and grader; workers are Bonsai 2 4B at medium effort unless the row says V4.1. Thinking: the effort setting each run sent (high = V4.1's default; off = the chat template's thinking switch off). V4.1 Flash: the official FP4/FP8 mixed-precision checkpoint on Local Inference Lab's vLLM build R37 (TP4, DSpark speculative decoding, 7 draft tokens). Qwen3.8-27B Q8_0 and Ternary Bonsai 2 4B PQ2_0 on PrismML's llama.cpp fork b10743, no speculative decoding. All four RTX PRO 6000s run our permanent core and memory overclock (GPC +350, MEM +6000).

- **Hands-off:** no `write_file` or `edit_file`, so every file change had to be a worker's delivery the coordinator accepted. At medium thinking it made 6 model calls without delegating; we stopped it at 8.3 min. With thinking off it delegated once — the whole module to a single worker, leaving 158 of 159 slots idle — and we stopped it at 12.2 min. That run changes two things from the swarm cell, not one.
- **Author and checkers** is how V4.1 used its workers on this task: it wrote the module itself and sent workers to check it section by section. Given that role with thinking off, the 27B wrote the module early and sent Bonsai 4B checkers out in rounds — the first round one per section of the spec — 16 in all, each with a 10-minute cap; 9 of the 16 ended on a time or step cap. A checker did report that the visible tests crashed; the author never fixed it and stopped at 36.4 min. Graded, every cell-reference case passed (48 of 48), and all 474 cases in the groups for operator precedence, functions, error values, cycles, recalculation, whole-sheet integration, generated literals and rounding failed.
- One harness defect surfaced in the author run: a file pattern in a `delegate` call that matched nothing was skipped silently, which cost the author one round of checks. We fixed it afterwards; the author had the crash report and did not act on it.
- **Reading:** no arrangement of the 27B and the 4B workers came close on this task, and even V4.1, given workers, wrote the code itself. A single coherent artefact leaves little to delegate except checking.

6 · The race: one model against 64 agents, on one clock

No coordinator this time. **Nemotron 3.5 Super** (120B total, 12B active), its BF16 checkpoint quantized to FP8 at load, ran on two RTX PRO 6000s with up to 32 agents at once and one attempt per task at a time. **64 Ternary Bonsai 2 4B agents** ran on the other two (32 per card — about where, in our calibration, one card's 4B

throughput stopped rising), several attempts per task at once; the first attempt to pass the visible tests ships, and its siblings are cancelled.

The backlog: 32 small repositories, each with a bug, visible tests, and hidden tests that encode the general rule — a patch that only satisfies the visible cases fails them, by construction. Same task text, same tools (files and a sandboxed shell), the same per-attempt caps (25 steps, 15 minutes), one 60-minute clock started for both at once. A shipped task is graded at once on the hidden tests: fixed, or shipped wrong.

	Nemotron 3.5 Super, 2 GPUs	64 × Bonsai 2 4B, 2 GPUs
Fixed (hidden tests pass)	**28 / 32**	**9 / 32**
Shipped wrong / still open	4 / 0	19 / 4
Ships that were right	87.5%	32.1%
Clock	93.1 s (backlog closed)	38.4 min (we stopped it)
Fixed by minute 1 / 2 / 5 / 10 / 20	23 / 28 / 28 / 28 / 28	1 / 3 / 7 / 8 / 9
Attempts	32	322
Output tokens	79,773	2,721,309
GPU energy, kJ (average power)	84.8 (911 W)	1,101 (478 W)
Per fix: tokens · kJ	2,849 · 3.0	302,368 · 122.4
Decode per stream, tok/s (10th / median / 90th pct)	27.4 / 64.3 / 127.9	18.5 / 20.6 / 23.2
GPU busy, median (10th–90th pct)	99%	40% (30–48%)

One run. Energy is GPU-only, each side's own two cards. GPU busy: the median of per-second nvidia-smi samples while the side was working (Nemotron: from 10 s to the end; the swarm: minutes 2–30). Nemotron: vLLM nightly 0.32.0rc1.dev190 (commit 3f1bcfd8), FP8 by online quantization of the BF16 checkpoint, tensor-parallel over two cards, MTP speculative decoding (5 tokens). Bonsai 2 4B: PQ2_0 on PrismML's llama.cpp fork b10743, one server per card, no speculative decoding. The RTX PRO 6000s run our permanent overclock (GPC +350, MEM +6000).

Task family	Tasks	Nemotron fixed	Swarm fixed
Trap, the rule is in the repo	10	10	**5**
Implement from a spec	8	7	**0**
Make it fast, same output	2	2	**0**
Wrong-file symptom	4	4	**2**
Trap, closed book (no spec)	8	5	**2**
All	**32**	**28**	**9**

Trap: a tempting local fix is wrong and the repo's spec states the real rule. Closed book: the same traps without the spec, so the model has to know the standard. Spec: implement a parser or evaluator from a written spec. Wrong-file symptom: the traceback points at one file, the cause is in another.

- **The big model closed the backlog in 93 s.** All 32 of its first attempts passed the visible tests; 28 passed the hidden ones. Its 4 misses were 3 closed-book traps and 1 spec task.

- **The swarm lost on what it shipped, not on speed.** 9 of its 28 ships were right. At that rate, shipping all 32 tasks gives about 10; a tie needed 25. No shipped patch on either side touched a test (0 of 60); the swarm's patches made the visible cases pass without the general rule.
- **More copies ship sooner, not righter.** Under first-to-pass, parallel copies select for the quickest patch that satisfies the visible tests. The swarm did not lead at any point: 23 to 1 after one minute.
- 81 swarm attempts ended on the step cap and 54 on the time cap; 97 were cancelled when a sibling shipped. On the spec tasks the steps ran out, not the GPUs.
- **The swarm's GPUs were under half busy** (median 40%, at about 243 W per card), against 99% for Nemotron's. Our reading, not tested directly: the limit was each llama.cpp server's main CPU thread, which ran saturated while it fed 32 streams. For the next study we ran two server processes per card.
- We stopped the swarm side at 38.4 minutes with 4 tasks open. Even 4 more fixes would make 13, against 28.

---

7 · Does a bigger swarm catch up?

The race says the 4B swarm loses on what it ships. Whether *any* swarm of a model can beat Nemotron's 28 is bounded by two numbers: **coverage**, the tasks on which at least one attempt is genuinely right (the ceiling of any way of choosing), and **selection**, picking that attempt without the hidden tests. Here every attempt is independent, runs to completion, and is graded on both test sets — same tasks, tools, caps and agent loop as the race.

- **Bonsai 2 4B, 16 attempts per task,** on six GPUs: two RTX PRO 6000s (80 slots, two server processes per card) and our second host's RTX 5090, RTX 5080, RTX PRO 4500 and RTX 3090 (24, 10, 24 and 16 slots).
- **Ternary Bonsai 2 27B** (PrismML's ternary version of Qwen3.8-27B), **8 attempts per task,** on two RTX PRO 6000s — the same cards Nemotron had in the race (48 slots, two processes per card).
- Both at medium effort with the Bonsai card sampling, 32,768 tokens of context per attempt. Pre-registered decision rule: only a model whose coverage reaches 29 — one more than Nemotron — gets a selector.

	Bonsai 2 4B × 16	Bonsai 2 27B × 8	Nemotron in the race
Independent attempts (per task)	512 (16)	256 (8)	32 (1)
Attempts that fix the bug	23.8%	74.6%	87.5%
Attempts that pass the visible tests	64.3%	93.8%	100.0%
… of those, fixed	37.1%	79.6%	87.5%
Coverage: tasks with at least one fix	**15 / 32**	**30 / 32**	—
Trap, the rule is in the repo (10)	9	10	10
Implement from a spec (8)	0	8	7
Make it fast, same output (2)	2	2	2
Wrong-file symptom (4)	2	4	4
Trap, closed book (no spec) (8)	2	6	5
Race rule replayed: first visible pass ships	8.0	25.0	28
Output tokens	3,985,087	1,926,092	79,773
Last attempt finished	33.4 min	38.6 min	93 s

Coverage by family is indented under the total. Race rule replayed = per task, the earliest-finishing visible-passing attempt ships. Bonsai 2 4B and 27B: PQ2_0 on PrismML's llama.cpp fork b10743. The RTX PRO 6000s run GPC +350 / MEM +6000 and the RTX 5090 a permanent memory overclock (MEM +6000); the RTX PRO 4500 runs at a 165 W cap.

Attempts per task	Bonsai 2 4B: expected coverage	Bonsai 2 27B: expected coverage
1	7.6	23.9
2	10.3	26.8
4	12.2	29.2
8	13.7	30.0
16	15.0	—

Expected tasks with at least one fixing attempt if only k of the recorded attempts had been run (the unbiased pass@k estimator), of 32.

- **The 4B flattens out far below the big model:** 7.6 tasks covered at one attempt, 15.0 at 16, and 0 of 8 spec tasks even at 16. No way of choosing among these attempts can reach 28.
- **The 27B covers 30 of 32** and passes the rule. Both tasks it never fixed are closed-book traps, and every one of its attempts on them landed on exactly the hidden score Nemotron's attempt got (117/130 and 121/130): the same failure, from two different models.

8 · Can the swarm pick the right attempt?

Selection without the hidden tests. For each task, agents of the same 27B each wrote a *probe*: a script that calls the module on 40 or more inputs drawn from the task text and prints every result, with no expected values. 82

probes were written (1–3 per task). Every visible-passing attempt ran every probe in the sandbox, and each selector was scored by the hidden verdicts already recorded.

Way of choosing (Bonsai 2 27B, 8 attempts per task)	Expected tasks fixed, of 32
First to pass the visible tests (the race's rule)	25.0
A random visible-passing attempt	25.3
Plurality: the largest group with identical probe output	26.9
Agreement: the attempt that agrees with the most others	**27.0**
Ceiling: tasks with at least one fixing attempt	30
Nemotron 3.5 Super in the race, one attempt per task	**28**

Agreement: the attempt whose probe outputs agree, line by line, with the most other attempts. Plurality: a member of the largest group of attempts with identical output.

- **Agreement reached 27 against Nemotron's 28** — inside the pre-registered tie band (a gap under 4), but short of the 29 that would have earned a live rematch, so we did not run one.
- **It rescues one-off errors:** on the spec family it shipped a correct attempt on 8 of 8 tasks, including tasks where only 2, 3 or 4 of 8 attempts were right. Wrong implementations each fail differently; correct ones agree.
- **It cannot overrule a shared belief:** its three misses inside coverage are all closed-book traps, where at most 50% of the attempts were right and the wrong ones agreed with each other. Voting amplifies whatever most attempts believe.
- **The cost of the tie:** Nemotron fixed 28 in 93 s; the 27B's attempts, on the same cards, ran until 38.6 minutes, before the probe stage.

9 · Our predictions, graded

Study	Prediction (written before the runs)	Measured	Verdict
Main	Extraction: the swarm wins	V4.1 97.2%, swarm 16.4%	**wrong**
Main	Research: the swarm wins or ties	V4.1 87.3%, swarm 87.7%	**held**
Main	Coding: a tie	V4.1 100.0%, swarm 36.0%	**wrong**
Coding follow-up	Author + checkers: 25–60%	14.8%	**wrong: lower**
Race	Nemotron fixes 21–26 of 32	28	**wrong: higher**
Race	The swarm fixes 12–20 of 32	9	**wrong: lower**
Race	The swarm leads at minute 10	Nemotron 28, swarm 8	**wrong**
Race	The swarm ships more wrong	swarm 19, Nemotron 4	**held**
Overwhelm	4B coverage 18–24 of 32	15	**wrong: lower**
Overwhelm	27B coverage 27–31 of 32	30	**held**
Overwhelm	4B attempts that fix: about 20–30%	23.8%	**held**
Overwhelm	27B attempts that fix: about 50–70%	74.6%	**wrong: higher**
Selection	Agreement: 26–29	27.0	**held**
Selection	Plurality: 25–28	26.9	**held**

6 of 14 held. We expected the swarm to win the extraction task and to lead the race early; it did neither. The ranges are as pre-registered; nothing was re-fitted.

10 · What this means if you are building with small models

- **If one big model fits, run it.** On these tasks it was right more often and, in the race, cheaper per correct answer (3.0 kJ per fix against 122).
- **A swarm pays where the work splits into independent, checkable pieces** — wide research by entity and source tied the big model at 36% of the GPU energy. Where the hard part is reconciliation or one coherent artefact, checking a small worker costs as much as doing the work.
- **A coordinator is only as good as the workers it can trust, and as itself.** The same coordinator design with V4.1 workers was the best arm on every task (tied on coding); with 4B workers it was capped by a coordinator that could not trust them.
- **If you fan out attempts, use a model that solves the task at least some of the time,** and choose by agreement between attempts rather than first-to-pass. Do not expect voting to fix what the model does not know; different model families, whose mistakes do not coincide, are the obvious next test, and we have not run it.
- **Serving many small-model streams from one llama.cpp process per card left the GPUs under half busy** in our race; run more than one server process per card and watch its CPU thread.

11 · Caveats

- One run per cell in the main study, the controls, the coding follow-ups and the race, with sampled decoding (temperature 1.0 on every model). The pre-registered tie bands (10 points; 4 tasks) are wide for that reason; the overwhelm study has 8–16 attempts per task.
- The tasks are synthetic: a generated mailbox, a sealed fictional web, a specified module and generated repositories. They were built to be hard and to reward the hard part, not sampled from real users.
- Energy is GPU power only. It leaves out the CPUs and host memory — and V4.1's Engram tables live in host RAM, while the 4B servers were limited by a CPU thread.
- Asymmetries we stated rather than fixed: V4.1 and Nemotron ran speculative decoding and the small models did not; V4.1 ran at high effort (no medium in its runtime) and the others at medium; the coordinator shared its GPU with a worker server.
- Nemotron ran FP8 on two cards for the race, not the BF16 configuration we qualified it in.
- The swarm's coding cell used no workers, and the follow-ups changed more than one thing at a time (tools, then thinking). Two of them were stopped by us before a reading.
- We stopped the race's swarm side at 38.4 minutes, which cannot change the result (above).
- The race's spec family is four problems with two variants each that differ only in their hidden sample, so it is four problems, not eight. The probes in the selection study were written by the same 27B whose attempts they judge.
- Only one coordinator (Qwen3.8-27B) and one coordinated worker model (Bonsai 2 4B) were tried; no mixed-family swarm, no second seed, and only one task of each kind.
- Bonsai 2's PQ2_0 files run only on PrismML's llama.cpp fork; Qwen3.8-27B ran on the same fork.
- The Ternary Bonsai 2 4B ran PrismML's pre-release files from its private beta. PrismML has since published the 4B (14–15 October) as files that differ from the ones we ran; we have not re-measured the public files.

Every figure on this page was computed from the raw rows of runs made on 17–18 October 2026 on our own machines. Run directories: `bench/runs/20261017-swarmlab` (each coordinated run's configuration, grade, totals, energy and peak memory, and the live grade every 10 s), `bench/runs/20261017-swarmlab-race` (both sides' full event logs and every shipped patch with its hidden verdict) and `bench/runs/20261018-swarmlab-passk` (all 768 independent attempts and the selection study). The rows are kept; to see them, or to send a question you want measured, write to hello@netsphere.com.ai. Online: netsphere.com.ai/reports/one-model-vs-a-swarm/