Recursive Flint: *27B-class code and agents at 2.5B active — once its thinking goes back in.*

Recursive's Flint (40B total, ~2.5B active, trained from scratch, released 21 October) went through our full quality suite at BF16, the qualification pipeline on one RTX PRO 6000, and a speed test on all seven of our GPU types. One early result looked like a weakness of the model and turned out to be our harness; a controlled re-run of both Flint and Qwen3.8-27B settled it.

Bottom line

- **Code and math at the Qwen3.8-27B level:** HumanEval 159 vs 160, MBPP 467 vs 475, GSM8K 482 vs 490 — from 2.5B active parameters.
- **Agentic work is level with Qwen3.8-27B — if the client sends Flint's reasoning back.** Our runner stripped it; Flint then stopped thinking on later turns, could not tell it was done, and looped until cut off (54 % of hard tasks). Sent back, as its model card asks: 27.0 of 34 hard tasks against Qwen's 25.3–26.3. Qwen does not depend on this; Flint does.
- **Weak on long state and exact copying:** the ledger walk 14 of 60 (Qwen 40), byte-exact tool-call payloads 95 of 120 (Qwen 118) — after very long thinking it writes tool calls in other models' formats.
- **English-first:** MGSM 85.1 % and Global-MMLU-Lite 70.6 % against 93.2 and 85.0; Burmese, Tajik and Yoruba under 50 %.
- **Fast where it fits:** 270 tok/s on a PRO 6000 and 261 on an RTX 5090 (llama.cpp, Q4_K_M, 24.2 GB). At BF16 on one PRO 6000 (SGLang) 162 tok/s at 1K of context, 50 at 120K.
- **It runs on our cards without the vendor's support:** the official SGLang kit is validated only on B300/H200; with triton backends it serves correctly on the RTX PRO 6000.

1 · What we ran

- **Model:** `RecursiveSI/flint` @ `2a247be0`, Apache-2.0, 79.6 GB BF16. 49 layers (2 dense + 47 mixture-of-experts, 256 experts, 6 per token plus a shared expert), 35 sliding-window (513-token) and 14 full-attention layers, 262,144-token context. Text only. **No thinking switch:** the model decides each turn whether to reason — here it chose to on every single item.
- **Quality engine:** the repo's own SGLang kit (SGLang 0.5.16 + `flint_sglang` 0.1.4), built locally. Its two validated attention backends (FA4, FA3) are locked to B300/H200 silicon, so we ran triton attention and triton MoE — a configuration Recursive does not validate. BF16 weights take 74 GB of the 96 GB card. Four one-card copies.
- **Setting:** the model's own sampling (temperature 1.0, top-p 0.95, top-k 20), 32K tokens of room per answer, per-item seeds, one run. Pre-registered before the first scored request.
- **Instruments:** HumanEval 164 and MBPP 500 (executed, 2-round self-repair), GSM8K 500, the ledger walk (60 chains, 12–40 hops through 8K–32K-token logs), payload fidelity (120 byte-exact `write_file` calls), agentic tier

2 (81 repo tasks) and tier 3 (34 held-out traps), MGSM (11 languages) and Global-MMLU-Lite (23 languages × 100); then the qualification pipeline (speed by depth, concurrency, needle retrieval to 120K, tool-call checks).

- **Reference:** Qwen3.8-27B as in plate 059 (Q8_0 on llama.cpp, medium effort, presence penalty 1.0) — a different engine and sampling, so context rather than a controlled pair; the agentic comparison in section 3 is controlled.
- **Speed:** llama.cpp b11804 (the first build that loads Flint) on seven GPU types, with a Q4_K_M we converted ourselves from the released weights. The community GGUFs were made the day before the release and are labelled "Flint Preview".

2 · Quality against Qwen3.8-27B

set	Flint (BF16)	Qwen3.8-27B	note
HumanEval (164)	**159**	160	level
MBPP (500)	**467**	475	8 cut off at 32K; level on finished answers (94.9 vs 95.0 %)
GSM8K (500)	482	490	−8
Self-repair	22 of 30	22 of 29	level
Ledger, walks fully right (60)	**14**	40	6 cut off at 32K: cap-bound
Payload, byte-exact (120)	**95**	118	format drift — section 4
Agentic tier 3 (34)	23	26	suite run; section 3
Agentic tier 2 (81)	78	81	suite run; section 3
MGSM, % (11 languages)	**85.1**	93.2	English 97.6; Telugu 64.4, Swahili 68.8
Global-MMLU-Lite, % (23)	**70.6**	85.0	English 90; Burmese 42, Tajik 44, Yoruba 49

Counts over all items (a reply cut off at 32K counts wrong) except the multilingual rows, which are % of finished answers (as plate 064C). Flint reasoned on 100 % of items in every set. The 40-item recipe probe found code and math clean at every answer cap from 4K; the ledger still cuts off 5 of 40 at 32K — Recursive recommends at least 32K and evaluated with up to 81,920.

Pros

- Matches a dense 27B on code (HumanEval, MBPP, self-repair) from ~2.5B active.
- Agentic tier 3 level with Qwen3.8-27B when its reasoning is sent back.
- Zero unparsable tool calls in 9,175 agentic tool calls; 43 of 44 tool-call checks clean.
- Plain and multi-hop retrieval perfect to 120K tokens.
- Apache-2.0; serves on stock SGLang 0.5.16 with its plugin, and on mainline llama.cpp.

Cons

- Weak at carrying state through long documents (ledger 14 vs 40), and runs out of 32K room there.
- Exact tool-call payloads break after long thinking (95 vs 118 of 120).
- Multilingual well behind (−8 MGSM, −14 GMMLU), worst on low-resource languages.
- Needs its reasoning sent back in multi-turn use, or it loops.
- Thinks at length: 2–3× the tokens of Qwen per agentic task.
- No thinking switch, no drafter, no vLLM support yet.

3 · Agentic work, and the reasoning it needs back

In the suite, Flint scored 23 of 34 on the hard tier against Qwen's 26 — and ended 17 of its 34 tasks at the 25-turn limit: after fixing the bug and writing "All tests pass", it kept calling `run_tests` about 20 more times with empty replies. Its turns after the first carried almost no reasoning.

The cause was our runner. It sent each assistant turn back as text + tool calls and dropped the `reasoning_content` field. Flint's model card says to keep it; its chat template writes it back into every earlier turn; the vendor's own client sends the whole reply back. Without it Flint sees a history in which it never thought before acting — and stops thinking, and stops recognising when it is done. We added an opt-in flag to send the reasoning back (requests are byte-identical without it), checked that it reaches the template, pre-registered the comparison, and ran both models both ways on the same seeds (hard tier × 3 seeds, repo tier × 2):

per seed (hard tier of 34 · repo tier of 81)	reasoning dropped (old runner)	reasoning sent back
Flint	23.7 · 76.0	**27.0 · 79.5**
Qwen3.8-27B	26.3 · 81.0	25.3 · 80.5

hard tier, pooled over 3 seeds	Flint dropped	Flint sent back	Qwen dropped	Qwen sent back
tasks ended at the turn limit	**54 %**	14 %	0 %	0 %
later turns that reason	24 %	52 %	100 %	100 %
median tool calls per task	26	11	7	7
passed (of 102)	71	**81**	79	76
tokens per task (median)	1,334	2,641	930	1,121

1. **For Flint it is decisive:** every seed with the reasoning sent back beats every seed without it, on both tiers; four tasks flip to passing in most seeds and none flip the other way. A residual 14 % still end at the limit re-running tests after a fix — Flint's own tendency, now at a much lower rate.
2. **Qwen does not depend on it:** its prompt opens a fresh thinking block every turn, so it re-thinks whatever the history holds; its scores move only inside its own seed-to-seed spread. Our first Qwen seed reproduced its earlier plate-059 row exactly (26 and 81).
3. **Like for like, Flint is level with Qwen3.8-27B on the hard tier and 2–3 tasks behind on the repo tier** — and it spends 2–3× the tokens per task to get there (median 2,641 against 930–1,121).
4. With the reasoning sent back, 20 of Flint's 21 hard-tier misses over three seeds are the closed-book traps (tasks that need knowledge outside the repo); Qwen's are 42 of 49. In the suite run Flint's other misses included a shallow-copy fix that left a nested list shared, and two "summary written and verified" claims with no file at the required path.

4 · Payload fidelity: format drift after long thinking

The payload test asks for one `write_file` call whose content must match a 300–800-character text full of quotes, backslashes and Unicode, byte for byte. Flint passed 95 of 120. Its failing replies ran a median 15,800 tokens of thinking and reply against 3,300 for passes — on a copy task — and then:

- **13 wrote the call as plain text in another model's format**, which Flint's parser cannot read: Qwen3-coder `<function=…>` XML (6), bare JSON with a name (3), a `function_calls` JSON list (2), a ReAct "Thought / Action" block and a JSON code block (2).
- **11 made a proper call with the wrong content**: 9 escaped it a second time (a literal `\n` where the file needs a newline), 2 were off by one character.
- 1 was a native call cut off by the 32K limit.

The prompt names no format, and the vendor's tool grammar allows free text before a call, so it cannot prevent this. Scored as served, per the pre-registration.

5 · Speed on seven cards

card	experts on CPU	decode	after 8K	prefill	real use	unsloth UD-Q4_K_XL
RTX PRO 6000	–	**270.0**	233.5	9,562	257.2	249.4
RTX 5090	–	**260.9**	223.0	8,547	238.1	239.4
RTX PRO 4500	–	175.3	151.1	3,881	162.1	153.0
RTX 3090	2 of 47 layers	147.4	141.3	2,525	139.8	132.1
RTX 5080	18 of 47	114.0	103.6	988	108.0	99.9
DGX Spark (GB10)	–	89.6	80.7	2,475	81.2	77.2
RTX 5070	29 of 47	46.7	44.8	371	45.3	42.9

Tokens per second, one stream. Decode = llama-bench tg128 (mean of 5) on our Q4_K_M (24.2 GB); prefill = pp512; real use = llama-server, 24 prompts × 1,024 tokens at the card's sampling. The PRO 6000 runs our standing overclock (GPC +350, MEM +6000) and the 5090 MEM +6000; the 5070 is the desktop's display card. Peak temperature 39–48 C on discrete cards, 71 C on the Spark.

- **Our self-converted Q4_K_M decodes 8–16 % faster than unsloth's UD-Q4_K_XL on every card**, and is 0.9 GB smaller. (The quality of either file was not measured.)
- **Real use runs at 91–97 % of the engine figure**; 8K of context costs 4–15 % of decode.
- **Against MiniCPM-V 4.7 35B-A3B on the same harness**, Flint is slower on cards that hold it whole (270 vs 367 on the PRO 6000) despite fewer active parameters — 49 layers, a 150K-token output head and one-day-old llama.cpp support — and level or ahead where experts run on the CPU (5080, Spark, 5070).
- **Fitting a card that cannot hold it:** at the same split the smaller Q4_K_M needs *more* GPU memory than unsloth's larger file. A fit tested on one file ran the 5080 out of memory and, on the Windows desktop, spilled into shared memory and decoded at 2.7 tok/s. Those runs were thrown out; the fits shown hold for both files.

6 · One PRO 6000 at BF16: depth, load, retrieval, tool calls

The qualification pipeline ran on one RTX PRO 6000 with the SGLang kit, its pools shaped so a full 262K-token request fits (316,686 full-attention tokens). It passes.

context	1K	4K	8K	32K	64K	120K
decode, tok/s	**162.3**	153.7	143.6	103.0	74.4	50.2
prefill, tok/s	12,865	**26,455**	25,132	17,777	12,412	6,768
time to first token	0.08 s	0.15 s	0.32 s	1.8 s	5.2 s	17.7 s

- **Concurrency:** total throughput climbs to ~750 tok/s at 32–40 parallel streams (633 at 24, 713 at 32, 752 at 40), then **collapses past 40** (249 at 48, ~180 at 56–64): the sliding-window memory pool fills and the server starts evicting and recomputing requests. That is the pool shape we chose to fit full-length context, not the model's ceiling — more streams need a larger sliding-window share. The pipeline's reported best of 1,610 tok/s is a measurement artifact (section 8).
- **Retrieval to 120K:** a single planted fact and three-hop lookups are perfect at every depth (30/30, 18/18); summing scattered facts nearly so (17/18). Decoy-dense search, recalling many facts at once and following a decoyed chain fail from 16–32K (9/18, 7/18, 2/18; 3 tries per cell).
- **Tool calls:** 43 of 44 checks clean on schema, tool choice, arguments and no hallucination. The one miss: asked to migrate a VM with no migration tool available, Flint kept calling tools instead of saying it could not.

7 · What to run

- **Coding agents:** yes — Flint's strength. **Always send** `reasoning_content` **back on every assistant turn** (as Recursive's card says); without it, expect loops and about 10 fewer hard tasks solved per 102 (71 vs 81). Give it at least 32K tokens per turn.
- **Engine:** on an RTX PRO 6000, the vendor's SGLang kit with `--attention-backend triton --moe-runner-backend triton` — one card holds BF16 with room for a full-length request. Keep under ~40 parallel streams at the full-context pool shape, or give the sliding-window pool more room for many short requests.

- **Single user, local:** llama.cpp b11804 or later with a Q4_K_M converted from the released weights — 270 tok/s on a PRO 6000, 261 on a 5090, 175 on a 4500; it fits whole on 32 GB cards.
- **Guard exact writes:** verify file contents written through tool calls, especially long verbatim payloads.
- **Not for:** long state-tracking through big documents, or work outside English — Qwen3.8-27B is clearly stronger there.

8 · Caveats and instrument notes

- One run per cell for the suite (sampled); the agentic comparison used 3 + 2 paired seeds per arm. Seeds of one arm span 2–4 hard-tier tasks.
- Flint picks its own thinking per turn; the Qwen rows ran with thinking fixed on at medium effort. Rows other than section 3's agentic comparison are context, not a controlled pair (different engine, sampling, precision).
- The triton serving path on RTX PRO 6000 is not a configuration Recursive validates. Outputs checked correct; the speed ladder's greedy (temperature 0) answer at 120K was right but trailed off into junk — greedy decoding is also listed as unvalidated by the vendor.
- **Concurrency measurement artifact:** our pipeline measures throughput over the window from the first token to the last. At 40 streams, two of three repeats read double (1,636 / 1,610 vs 771 tok/s) with the same wall time and tokens, because the server prefilled all requests before decoding any. Read by wall time, as above; the instrument is not yet fixed.
- Our tool-call check (like the agentic runner before the fix) also drops reasoning between turns; its one miss may share the cause.
- Harness changes made for this run, all opt-in or additive: a "no thinking switch" profile option; the agentic runner's reasoning flag; a fixed fetch command in the pipeline.
- Not measured: the quality of quantized files (19 community quants exist), vLLM (support is still a pull request), the vendor's own headline benchmarks (Terminal-Bench 2.1 54.9, SWE-bench Verified 74.2), and anything multimodal (Flint is text-only).

Every figure was re-derived from the raw rows of runs made on 22 October 2026 on our own machines. Run directories: `bench/runs/20261022-flint`, `bench/runs/20261022-flint-speed`, `bench/runs/20261022-flint-qual`. The rows are kept; to see them, write to hello@netsphere.com.ai. Online: netsphere.com.ai/reports/recursive-flint/

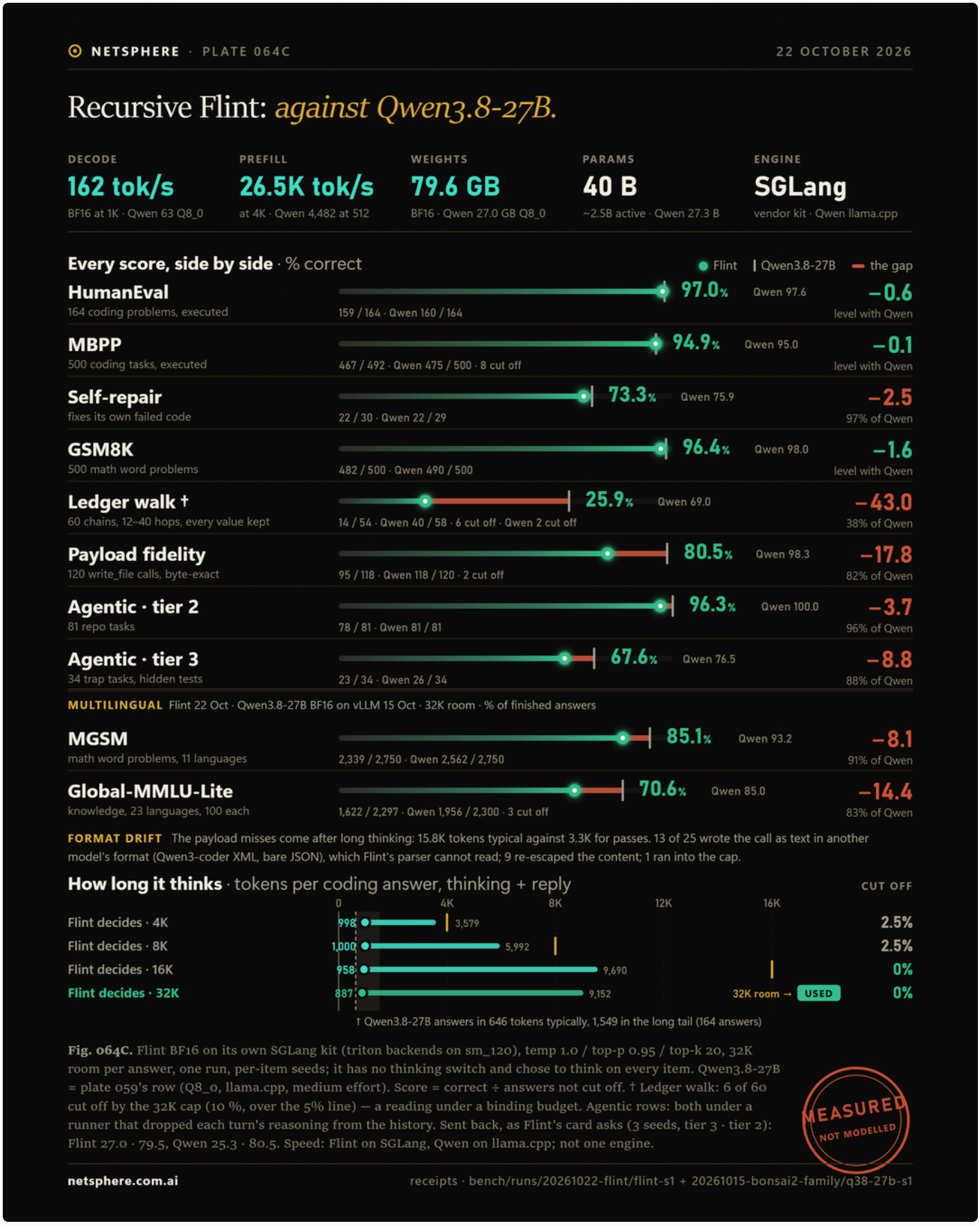

Plate 064C — Flint against Qwen3.8-27B, every score; the caption gives the agentic rows with reasoning sent back.

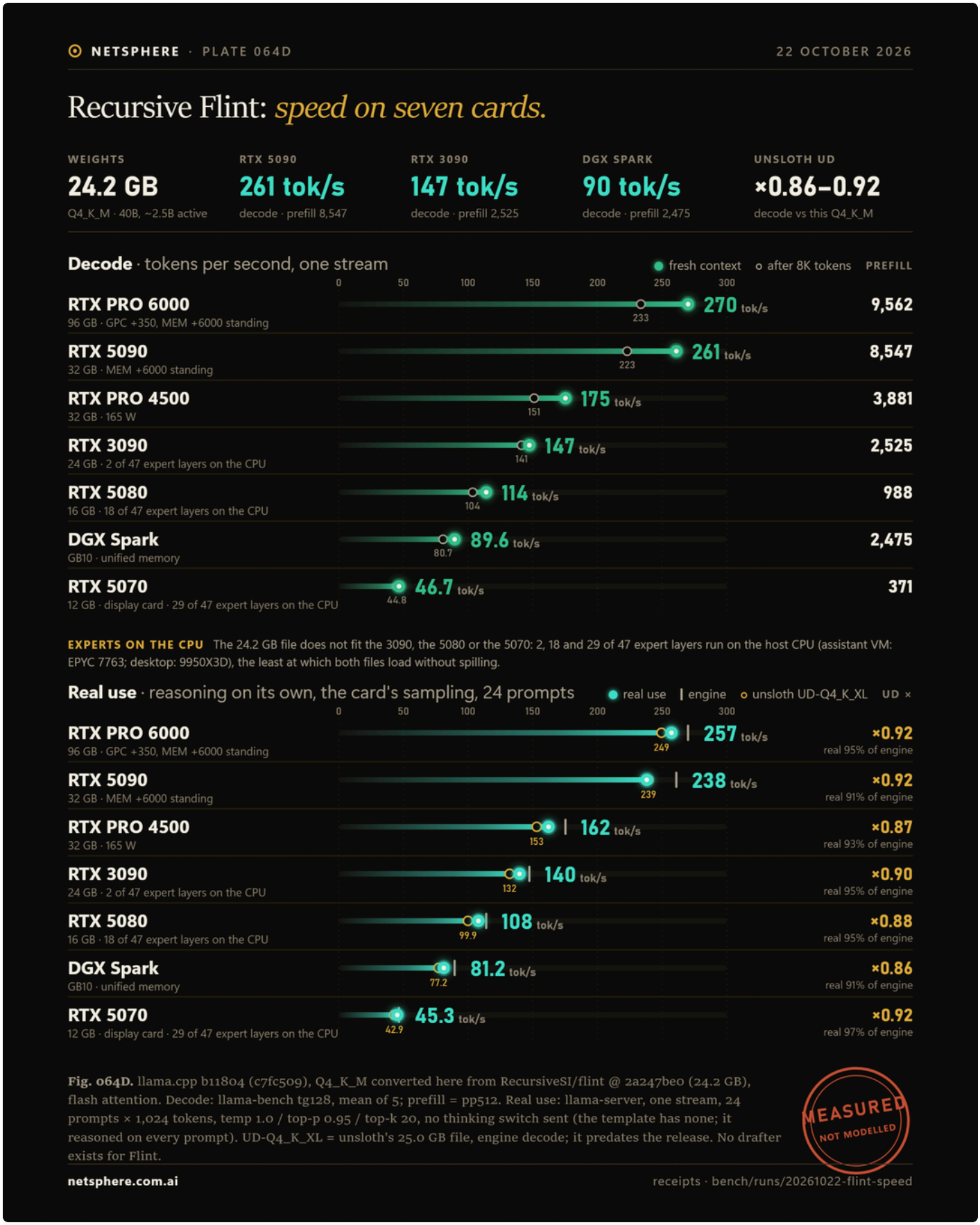

Plate 064D — speed on seven cards.