

Qwen3.8 Flash Next on Apple silicon with NukeTorch: four MacBook Pro M5 Max and one Apple M4 Max

Klay Moon, NukeTorch · Report v1.1 · model Jundot/Qwen3.8-Flash-Next-oQ4e-mtp (106.3 GB as shipped)

Headline

- **One user: 177.7 tokens per second**, relay over four Apple M5 Max: one take in the chat app, reasoning on, a series of 5 short coding prompts (170.5–193.4 per prompt, 3,034 tokens; Table 2a). The rate depends on the prompt.
- **One user, benchmark prompts, default sampling: 133.0 tokens per second** per request (median of 3 runs of 8 requests each; runs 132.7–133.1; Table 2).
- **Many users: 859.0 tokens per second combined**, 32 users at once in the chat app, relay over four Apple M5 Max (27.2 per user; Table 3). Benchmark protocol, greedy: 553.1 generated tokens per second at 32 concurrent requests (Table 1).
- **Quality:** GSM8K 91.5% with the default sampling against 95.1% for exact sampling, the same 1,319 problems (Table 4).

1. Four MacBook Pro M5 Max (relay over four Macs)

Table 1. Throughput benchmark: 1,024-token prompt, 128 generated tokens, greedy

Concurrent requests	Prompt tokens/s	Generated tokens/s	Time to first token (ms, mean)
2	1,250.7	156.7	1,591
4	1,608.4	296.4	2,385
8	1,881.7	407.6	3,962
32	2,127.3	553.1	14,262

Metrics, all requests submitted at once: prompt tokens/s = all prompt tokens ÷ the last first-token time and generated tokens/s = all generated tokens ÷ (end – last first-token time). Prompts: 1,024 tokens of a fixed Python-code corpus with a different random prefix per request, no chat template, 128 tokens forced (end token ignored). One run per row. The table started after the Macs had rested 235 minutes, with no thermal or performance warning and under 0.4 GB of swap in use on any Mac.

Table 2. One user

Run	Generated tokens/s per request (median of 8 requests)	Time to first token (ms, median)
default sampling, run 1	133.1	890
default sampling, run 2	133.0	885
default sampling, run 3	132.7	889

Default sampling = NukeTorch's default sampler with the model's own generation settings (temperature 1, top-k 20, top-p 0.95), eight chat prompts of 1,016–1,023 tokens, one at a time, up to 256 new tokens (answers of 230–256 tokens); a sampled answer differs from run to run, so these are per-request rates under the default sampling, not a fixed amount of work; the default sampler's quality against exact sampling is in Table 4. These runs followed the throughput table above back to back; the Macs had rested 235 minutes before that table (no thermal or performance warning, under 0.4 GB of swap in use on any Mac).

Table 2a. One user in the chat app

Prompt	Generated tokens/s
Go FlipCase	176.1
Python valid_date	170.5
Python Fibonacci (memoized)	193.4
Python palindrome check	185.1
Python merge two sorted lists	188.6
Series: 3,034 tokens	177.7

One take, one user, reasoning on, in the chat app on the four Macs: the 5 prompts sent one after another, the next when the answer before it finished. Each prompt's rate = its generated tokens ÷ its own time from first to last token; the series rate = all generated tokens ÷ the answers' own times added up. One take, no repeats: these prompts are short coding requests, and the rate depends on the prompt; Table 2 is the repeated benchmark series on longer prompts.

Table 3. Many users in the chat app

Users at once	Prompt	Combined tokens/s	Per user: min / median / max	Tokens
32	merge-two-sorted-lists, cap 1,536 (none capped)	859.0	24.4 / 25.5 / 38.2	17,753

One filmed take. Each user's speed = its own tokens ÷ its own time from first to last token; combined = the sum of the users' speeds, a throughput of users running at the same time. This counts differently from Table 1's all-request rate, so the two are not compared.

Table 3b. Many users, benchmark protocol

Sampling	Concurrent requests	Generated tokens/s, all requests	Generated tokens/s per request (median)	Time to first token (ms, mean)
default sampling	8	488.2	57.7	4,020
default sampling	32	655.7	17.6	14,333
greedy	8	407.6	44.8	3,962
greedy	32	553.1	15.4	14,262

Every request is the Table 1 benchmark request (1,024-token prompt, 128 tokens forced), all submitted at once; all-request rate = all generated tokens ÷ (end – last first-token time). Sampled rows use the model's generation settings of Table 2; the per-request column is a per-request rate under the default sampling. Sampled rows measured on build 8548c049; greedy rows are Table 1's runs.

Table 4. Quality: GSM8K

Sampling	Correct	Accuracy
default	1,207/1,319	91.5%
exact	1,255/1,319	95.1%

The same 1,319 GSM8K problems (two runs, both on build 8548c049); 'exact' samples every token exactly from the model's distribution. Our bar: at most 5% loss against the reference; the default is 3.8% below it. Setup: reasoning on, the model's sampling settings, answers limited to 1,650 tokens; 110 default and 65 exact answers reached the limit (13 and 17 of them still correct). With 1,319 questions the binomial 95% interval of each accuracy is about ± 1.5 points for the default and ± 1.2 for exact sampling. On the same problems, 63 were solved only by exact sampling and 15 only by the default (paired sign test $p < 0.001$): a small but real loss; more of the default's answers run to the limit (110 against 65). The default arm answered 1 question at a time, the exact arm 8 at a time.

Table 5. MLX on four Macs

Engine, as installed	Four-Mac result
MLX / mlx-lm 0.32.2	no maker path for this model

mlx-lm 0.32.2 ships no implementation of this model's architecture (`qwen4_exp`), so it cannot load it on any number of Macs.

Table 5b. Memory and power on four M5 Max

Workload	Peak memory, all four Macs (GB)	Per Mac (GB)	Power, all four Macs (W)	Energy per generated token (J)	Generated tokens / active serving (s)
one user, default sampling	79.1	21.0 / 23.2 / 23.3 / 11.7	341.7	3.60	28,672 / 302
32 users	89.7	23.8 / 26.5 / 26.6 / 12.8	374.0	2.24	57,344 / 344
idle (no model loaded; a separate earlier run)			17.7		

Peak memory = the serving process's peak footprint on each Mac (GB = 10^9 bytes). Power is whole-Mac input power from each Mac's battery controller; the meter gives no GPU, CPU or memory breakdown. Power = mean of the battery-controller intervals that lie entirely inside the active serving period (4–5 per Mac, about 239 of 302 s for one user and 241–301 of 344 s for 32 users); the first and last partial intervals, which mix in idle or loading, are not used. The meter refreshes about every 60 s, so the edges of the active period are assumed to run at the same power. Active serving includes prompt prefill, so energy per generated token counts prefill too. Idle: a separate earlier run with no model loaded; the power run's own idle reading was 43.9 W, raised by the charging battery below. At the start of the power run the battery of Mac 4 was still finishing its charge, which can add to the measured power.

2. One Apple M4 Max (128 GB)

Table 6. Throughput benchmark on one M4 Max: 1,024-token prompt, 128 generated tokens, greedy

Concurrent requests	Prompt tokens/s	Generated tokens/s	Time to first token (ms, mean)
2	326.1	66.1	5,908
4	344.1	65.0	10,789
8	346.9	109.0	20,898

Same protocol and formulas as Table 1, on one Apple M4 Max (128 GB, macOS 26.7.1 (25G241), High Power, mains power, battery 100%; charged); every row started after a quiet-machine check. The whole model runs in one process on one Mac. No comparison is drawn between this Mac and the M5 Max machines.

Table 7. One user on one M4 Max, default sampling

Run	Generated tokens/s per request (median of 8 requests)	Time to first token (ms, median)
default sampling	83.4	3,363

The same eight chat prompts and settings as Table 2; per-request rate under the default sampling. Its quality on this Mac, measured on an earlier build with the same sampling settings: GSM8K 186/200 with the default sampling against 187/200 with exact sampling, within our 5% bar.

Published speeds for comparison

Source	Generated tokens/s, one user
NukeTorch, four Apple M5 Max (48 GB each), 4-bit, relay	177.7
Strata v0.1.42, one NVIDIA H100 NVL, ~4-bit (Unsloth GGUF)	151
mlx-serve v26.8.11, one Apple M4 Max, 4-bit, its fastest setting	78
mlx-serve v26.8.11, one Apple M4 Max, 4-bit	60

At 4-bit on one NVIDIA H100 NVL, the published one-request rate is 151 tokens per second; NukeTorch's one-user rate is 1.18× that. With four requests at once, that H100 setup reached about 177 tokens per second in total. The fastest published 4-bit figure we found is 78 tokens per second on one Mac; NukeTorch's one-user rate is 2.28× that. These are different machines, packs and prompts, so this is a reference, not a same-conditions comparison; the published figures were not run here. Sources: Microsoft Tech Community, Qwen3.8-Flash-Next on two H100s in an Azure Spot VM (<https://techcommunity.microsoft.com/blog/appsonazureblog/qwen3-8-flash-next-on-two-h100s-in-an-azure-spot-vm/4563893>); mlx-serve release notes (<https://github.com/ddalcu/mlx-serve/releases/tag/v26.8.11>).

Method

Setting	Value
Model	Jundot/Qwen3.8-Flash-Next-oQ4e-mtp at revision c4ddfb54a967, 106.3 GB as shipped; base model Qwen/Qwen3.8-Flash-Next. With thanks to Jundot for the quantized checkpoint (made with oMLX's quantizer) and to the Qwen team for the model
Machines	Section 1: four MacBook Pro, Apple M5 Max, joined by a Thunderbolt bridge, relay over four Macs. Section 2: one Apple M4 Max, 128 GB
Build	NukeTorch, clean trees: 8548c049 (Table 1), 8548c049 (Table 2), 8548c049 (Table 3b, sampled), 0913b001 (Table 5b, memory and power, run commit 9783fdf2), 0913b001 (Table 5b, idle, run commit c0be0c5c), 8548c049 (Table 4, default-sampling arm), 8548c049 (Table 4, exact), 79819113 (Tables 6 and 7), ec273c70 (Table 7, quality)
Rate	Table 1's formulas; per-request generated tokens/s = tokens ÷ (last – first token)

How to reproduce

1. **Machines:** Section 1: four MacBook Pro 16-inch, Apple M5 Max (18-core CPU, 40-core GPU, 48 GB), joined by a Thunderbolt bridge, on mains power. Section 2: one Apple M4 Max, 128 GB, as recorded under Table 6.
2. **Cooling (four-Mac runs):** No external fans; each Mac's own fans under automatic control (declared by NukeTorch). Our other published benchmarks used external fans; the four-Mac runs here did not.
3. **Model:** Jundot/Qwen3.8-Flash-Next-oQ4e-mtp at revision c4ddfb54a967, 106.3 GB as shipped, unchanged; on four Macs NukeTorch relays it across them (pipeline parallel).
4. **Engine:** NukeTorch, builds as listed under Method (available to licensees).
5. **Workloads:** Table 1 and Table 3b: the 1,024-token benchmark prompt with 128 generated tokens, every request submitted at once; Table 2: the eight chat prompts, one at a time, with the model's own sampling settings; Table 2a: the chat app's five-prompt series; Table 3: 32 users in the chat app, one coding prompt; Table 4: the 1,319 GSM8K problems.
6. **Measure:** the formulas under Tables 1 and 2.

Prepared by NukeTorch Inc. Report v1.1. Author-run measurements; no independent reproduction yet.