Qwen 3.8 27B vs 3.6 on RTX 3090: Speed and VRAM Tested

August 14, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Qwen 3.8's 27B open weights landed this week, and if you own a 24GB card the only question that matters is what they do on hardware you already have. I ran both generations back to back on the same RTX 3090 in one headless session, same quant publisher, same recipe name, both files checksummed against upstream before either one loaded. The speed axis turned out to be the boring half. The memory axis put a measured number on something this site has been quoting secondhand from a pull request since May, and the number is a lot smaller than the one we've been repeating. Here's the run, what a single 3090 actually does with the new weights, and the four things this bench cannot tell you.

📚 **More on this topic:** DFlash vs MTP on RTX 3090 · Qwen 3.6 27B with MTP: 60 tok/s · Qwen 3.6 Complete Guide · Best Qwen Models Ranked · The 35B MoE on a 3090

The Qwen3.8-27B weights went up on Hugging Face on August 5, the community GGUFs followed on the 13th and 14th, and by the time I finished downloading one, my feed was already full of people quoting the model card at each other. Apache 2.0, 262K context, 64 layers, trained with multi-token prediction. All true, all copied from the same page, none of it a measurement.

So I benched it. One RTX 3090, one headless session, Qwen 3.8-27B against Qwen 3.6-27B at the same quant from the same publisher, run A-B-B-A so that drift and thermal state show up as a difference between the two halves of each pair rather than hiding inside the comparison. I went looking for other 3090 numbers on the 27B before I started and came up empty on August 14, which is less a boast than a statement about how new the file is.

Here's the deal: on generation speed, the new generation is a wash. The interesting result is 254 MiB wide and sits in the VRAM column, and it contradicts a figure this site has been carrying since May.

The run

Hardware. Miu, my workstation: single RTX 3090 24GB, Intel Core i7-8086K, 64GB DDR4-2667 across four Samsung dual-rank DIMMs in two channels, Xubuntu 24.04. Driver 580.159.03.

Display manager stopped for the duration, so the card starts each slot with 9 MiB in use and nothing competing for it.

Build. llama.cpp b10088, commit `67b9b0e`, CUDA.

Command. `llama-bench` with `-ngl 99 -fa 1 -p 512 -n 128 -d 0,4096,8192 -r 5`. Five repetitions per cell, three context depths, prompt processing and generation measured separately.

Models. Both Unsloth UD-Q4_K_XL, same publisher, same recipe name on the tin:

	Qwen 3.8-27B	Qwen 3.6-27B
File on disk	17,923,394,624 bytes (16.69 GiB)	17,612,564,704 bytes (16.40 GiB)
sha256 vs upstream	`bee238bb…` matched	`ff6941de…` matched
`general.architecture`	`qwen35`	`qwen35`
`block_count`	65	64

Both checksums were verified against the upstream repos before either model was loaded, which is the step that turns “I downloaded a file called UD-Q4_K_XL” into “I ran the file Unsloth published.” Caches were dropped between models so neither one got to reuse the other’s page cache.

Order. A1 (3.8), B1 (3.6), B2 (3.6), A2 (3.8). GPU start temperature ranged 57 to 62C across the four slots, clocks sat at 1725 to 1740 MHz, and package power landed between 417.05 and 418.90 W. Nothing thermally interesting happened.

Repeatability first, because it licenses everything below

A two-model comparison is only worth reading if the harness can tell the two models apart from itself. So here is the same model against itself, first slot versus last slot, roughly six minutes apart:

Test	3.8 A1	3.8 A2	3.6 B1	3.6 B2
tg128	41.49	41.49	41.78	41.76
tg128 @ d4096	41.02	41.00	41.21	41.19
tg128 @ d8192	40.49	40.47	40.68	40.67

Same-model repeats land 0.00 to 0.02 tok/s apart on every generation cell, against reported per-cell standard deviations of 0.02 to 0.05. Generation on this rig is a solid measurement.

Prompt processing is not, and I want that on the record before the table below rather than after it. Same-model repeats on `pp512` differ by 0.50 t/s for 3.8 and 2.87 t/s for 3.6, but the within-run standard deviation on those cells runs ±14.85 to ±37.22. Any prompt-processing difference smaller than about 30 t/s on this setup is noise wearing a number's clothing.

The numbers

Each figure is the mean of that model's two slots.

Test	Qwen 3.8-27B	Qwen 3.6-27B	Delta
pp512	1440.31 t/s	1432.12 t/s	not resolvable
tg128	41.49 tok/s	41.77 tok/s	−0.67%
pp512 @ d4096	1372.78 t/s	1367.86 t/s	not resolvable
tg128 @ d4096	41.01 tok/s	41.20 tok/s	−0.46%
pp512 @ d8192	1312.78 t/s	1308.09 t/s	not resolvable
tg128 @ d8192	40.48 tok/s	40.67 tok/s	−0.48%

Every `pp512` gap in that table is smaller than the run-to-run spread of the runs that produced it, so there is no prefill delta to report in either direction. The generation rows are measurements.

Read the two halves differently, because they carry different weight.

The prompt-processing column is a wash and I can't honestly call it anything else. The largest gap is 8.18 t/s at depth 0, which sits well inside the ±25 to ±37 spread of the runs that produced it. If 3.8 prefills faster on this card, this bench can't see it.

The generation column is real. A 0.28 tok/s gap at depth 0 is more than ten times the same-model repeat spread and roughly six times the reported standard deviation, and the sign is consistent across all three depths and all four slots. Qwen 3.8-27B generates a bit under one percent slower than 3.6-27B here, every time, and the margin narrows slightly as context grows.

One percent is not a thing you will feel. At 41 tok/s you are looking at about a quarter of a second across a 1,000-token reply. What makes it worth measuring is the reason it is that small, which is the next section.

The “nothing changed for your rig” argument has been made twice on this site already, and I’m not going to make it a third time. Inkling 975B vs Your 3090 covers what a frontier release does and doesn’t do to hardware you own, and the DFlash bench across both Qwens found the same near-parity a generation earlier, 2.59x against 2.56x on speculative-decoding speedup. If you want the stay-versus-move framing for the Qwen family specifically, that decision lives here and this page is not going to fork it.

The 254 MiB

Peak VRAM, measured on the card during each slot:

	Peak VRAM	Repeats
Qwen 3.8-27B	17,942 MiB	A1 and A2 identical
Qwen 3.6-27B	17,688 MiB	B1 and B2 identical
Difference	**+254 MiB**	**+1.44%**

Four slots, two values, no variance at all. That is the cleanest number in the whole run.

Now the part that matters. Qwen 3.8’s Unsloth GGUF carries 65 blocks where 3.6’s carries 64, and the extra one is the multi-token-prediction head: `nextn.eh_proj` plus the `enorm`, `hnorm`, and `shared_head_norm` tensors that go with it. Both models were trained with MTP, and Qwen’s own 3.6 card says so too. The difference is that the stock Unsloth 3.8 quant ships the block inside the file, and the stock Unsloth 3.6 quant does not, which is exactly why the community had to bake separate MTP GGUFs for 3.6 back in May.

This site has been quoting a number for what that block costs, and quoting it from someone else. Our DFlash vs MTP head-to-head puts “~2.49 GiB MTP layer (per PR)” in its comparison table, sourced from the PR #22673 thread rather than from anything I measured. Our MTP throughput piece turns that into advice: “MTP costs roughly 2.5 GiB extra VRAM beyond the base model,” with a warning that a 16GB card might not have room for it.

Here is the honest reconciliation, because the two figures are not measuring the same thing and I’d rather say so than take a cheap win. The PR-derived 2.49 GiB is the layer plus its KV cache during an active speculative run. My 254 MiB is peak allocation with the block resident and speculation switched off, so no MTP KV cache is ever allocated. Those are different quantities and both can be true.

What does not survive is the weights half of the claim, the one our own page states as “2.5 GiB extra VRAM beyond the base model.” The entire generation-over-generation difference on this card, MTP block included, is 254 MiB. The block’s weights cannot be costing gigabytes, because the whole delta between the two models is a quarter of one. If you have been sizing a 16GB card around a 2.5 GiB penalty for having MTP in the file, that penalty is smaller than you were told, and both of those pages need the correction.

A bigger file that costs nothing to generate from

3.8 is the larger model on every static measure. It carries 27.32B parameters against 26.90B, a 1.56% increase. Its file is 310,829,920 bytes bigger, which is 296.4 MiB. The tensors llama-bench loads come to 16.68 GiB against 16.39 GiB, the same 0.29 GiB gap from the other direction.

It generates 0.67% slower.

The intuition worth killing here is the one that reads a file-size number and expects generation speed to track it. Decode speed is set by the bytes the GPU reads on every single token, and with speculation off the MTP block sits outside that set. llama-bench does not run speculative decoding. The block gets loaded into VRAM, takes its share of those 254 MiB, and is then never touched by the decode loop. A model that is 296 MiB bigger on disk costs essentially nothing per token, because those particular bytes are not read per token.

That leaves a sub-one-percent residual on generation that this bench is not going to resolve into a cause. It’s consistent in sign and comfortably above the noise floor, so something is producing it, and 0.42B parameters of extra tensors sitting in the same memory space is a reasonable suspect. I’m not going to dress a suspect up as a finding.

There’s a rough consistency check sitting in the arithmetic, worth about as much as the word “rough” suggests. 0.42B parameters at Q4-class bit rates comes to somewhere around 225 MiB, against a 254 MiB measured VRAM delta and a 296 MiB file delta. Same order of magnitude, and that is the most it establishes. The two files also differ by quant enum, and 3.8’s is the smaller one, so some of that agreement is coincidence rather than accounting. It tells me nothing is wildly off. It does not tell me the delta is fully explained.

Four things this run does not tell you

The quant enums differ, despite identical names. llama-bench reports 3.8 as `Q4_K - Small` and 3.6 as `Q4_K - Medium`. Their `general.file_type` values are 14 and 15. Same publisher, same `UD-Q4_K_XL` label, different underlying quant enum. Unsloth's dynamic quants pick bit rates per tensor, so the top-line enum is a summary rather than a recipe, but it does mean these two files are not bit-for-bit comparable builds. Note the direction: 3.8 uses the smaller enum and is still 296 MiB bigger, which makes the extra block harder to argue away rather than easier.

`block_count` 65 is not 65 transformer layers. Both models are 64 layers deep, and Qwen's card for 3.8 states "Number of Layers: 64" outright, same as 3.6's. Block 64 in the 3.8 file is the MTP head, not another decoder layer. The raw count invites a "3.8 is deeper" reading that is simply wrong.

3.8's GGUF is a Dynamic V3.0 preview. Unsloth's card says so: "This GGUF uses Unsloth Dynamic V3.0 (preview)." Every other Unsloth row in our benchmark corpus is from the 2.0 generation. A preview quantization pipeline against a mature one is a confound I can name but not subtract, and it is the most likely home for that sub-one-percent generation residual.

Nothing here tells you what MTP does when you turn it on. llama-bench does not run speculation, so the block was loaded and idle for all four slots. Whether `--spec-type draft-mtp` works against this file on b10088, what acceptance rate it hits, and what it costs in KV cache once it is actually drafting are all untested. That's the next bench, not this one.

What this measures, and what it doesn't

Throughput on one rig, at one quant, on one build, with speculation off. That is the entire claim.

I have not evaluated Qwen 3.8-27B's output quality, run it against a single benchmark suite, or formed a view on whether it is a better model than 3.6. Nothing on this page supports a sentence about coding ability, reasoning, tool calls, or long-context behaviour. Anyone telling you today how much smarter 3.8-27B is has not had the weights long enough to know, and neither have I.

The bottom line

If you're running Qwen 3.6-27B on a 24GB card today, Qwen 3.8-27B at the same quant will load, will fit with the same headroom you're used to, and will generate at the same speed you're used to, within a percent that you will never notice. It asks for 254 MiB more VRAM than its predecessor and hands you an MTP block in the base file for it, which is a trade nobody running a 3090 needs to think about and anyone running a 16GB card should.

The reason to move is whatever the model does that this bench doesn't measure. Speed on your card is not the reason, and now there's a number behind that instead of a model card.

Related guides

- DFlash vs MTP on RTX 3090: I Tested Both Locally — where the 2.49 GiB figure this page corrects currently lives
- Wicked Fast Qwen 3.6 27B: 60 tok/s with MTP — what MTP does once it's actually drafting
- Qwen 3.6 Complete Guide — the 27B dense and 35B MoE, and which fits your card
- Best Qwen Models Ranked — the whole family, every size
- Best Way to Run Qwen 3.6 35B MoE Locally — the same llama-bench protocol, applied to the MoE
- Kimi K3 & Qwen 3.8: Open Weights You Can't Run — the page that spent a month waiting for these weights
- The benchmark corpus — every measured row on this site, with hardware and flags