How to Run GLM 5.2 Locally: GPU, VRAM & Quant Guide

June 21, 2026

Download this guide as PDF

> **Quick Answer:** GLM 5.2 is the #1 open-weights model on Artificial Analysis right now — and it's a 753B-parameter monster that weighs 1.51TB at full precision. Nobody runs that. The real question is which quantized version you target and what it takes to feed it. The paths range from one Mac Studio, to a 4-card workstation, to an 8-GPU datacenter node, with system-RAM offload as the slow-but-possible floor. There's one quant most people should actually aim for — it's not the biggest one that fits, and it's not the smallest one that runs. This guide walks the full Unsloth quant ladder with live file sizes, maps each hardware path to what it can hold, and ends with the honest off-ramp for the 95% who shouldn't run this locally at all.

📚 **Related:** Best Local Coding Models, Ranked · Used GPU Buying Guide · MoE Models Explained · Quantization Explained

GLM 5.2 landed on June 16, 2026, and within days it was sitting at the top of Artificial Analysis's open-weights Intelligence Index. MIT-licensed, 1M-token context, scores that brush the closed frontier on real software-engineering work. So of course the r/LocalLLaMA question showed up immediately: can I run this thing at home?

Short version: yes, but "at home" is doing a lot of work in that sentence. The full weights are 1.51TB. This is not a "download it on your 3090" model. It's a "do the memory math first" model, and most of the guides floating around skip the math.

So let's do the math. Below is the live Unsloth quant ladder, every realistic hardware path from a single box to an 8-GPU node, what each path can actually hold, and the API off-ramp for everyone who reads this and correctly decides it's not worth it.

What GLM 5.2 actually is

GLM 5.2 is a Mixture-of-Experts model from Z.ai: **753 billion total parameters, but only ~40 billion active per token.** That gap is the whole reason a model this big is even discussable for local hardware. The router picks a small slice of experts for each token, so the compute per token is 40B-class and fast, while the memory footprint is the full 753B you have to keep loaded.

If you've read our MoE explainer, this is the same "runs like 40B, weighs like 753B" tradeoff Mixtral and DeepSeek made, just at a much larger scale.

The rest of the spec sheet, from the official model card:

- **License:** MIT. No regional restrictions, no acceptable-use rider. Genuinely open.
- **Context:** 1M tokens, built for long-horizon agent work.
- **Modality:** text-only. No vision.
- **Native precision:** BF16.

On the leaderboards it earns the hype. It tops the Artificial Analysis open-weights Intelligence Index at **51**, and on SWE-bench Pro it posts **62.1**. That's close enough to the closed frontier that for a lot of coding and agent workflows, the gap stops being the thing that decides your stack. That's the pitch. Now the reality.

The size reality

At native BF16, GLM 5.2 is **1.51TB** of weights. Not gigabytes. Terabytes.

To put that in perspective: a 512GB Mac Studio, the largest unified-memory consumer machine Apple ever shipped, can't hold a third of it. An 8× H100 node, 640GB of the fastest VRAM money rents, can't hold half. Running GLM 5.2 at full precision is a datacenter-rack problem, and nobody outside a serving company actually does it.

That's fine, because nobody needs to. This is what quantization is for. Drop the weights from 16-bit to 4-bit and you cut the footprint by roughly 4x, with quality loss that on a well-built quant is small enough to ignore for most work. The whole "can I run GLM 5.2" question collapses into "how far down the quant ladder do I go, and what holds it." So here's the ladder.

The quant ladder

These are the live GGUF sizes from Unsloth's GLM-5.2 repo, pulled fresh. Unsloth's Dynamic 2.0 quants don't apply one bit-width across the whole model — they spend more bits on the layers that matter and fewer on the ones that don't, which keeps quality (measured as KL-divergence from the full BF16 model) lower at a given size than a naive quant of the same name. That's why the "UD-" quants below punch above their bit count.

Quant	Size	Bits	Quality vs BF16
UD-IQ1_S	217 GB	1-bit	Heavy loss. Runs, but visibly dumber.
UD-IQ1_M	228 GB	1-bit	Heavy loss.
UD-IQ2_XXS	238 GB	2-bit	Noticeable degradation.
UD-IQ2_M	239 GB	2-bit	Noticeable degradation.
UD-Q2_K_XL	254 GB	2-bit	Usable in a pinch; reasoning chains suffer.
UD-IQ3_XXS	282 GB	3-bit	Getting respectable.
UD-IQ3_S	309 GB	3-bit	Solid for the size.
UD-Q3_K_M	343 GB	3-bit	The realistic floor for “still feels like GLM 5.2.”
UD-Q3_K_XL	343 GB	3-bit	**The pragmatic target.**
UD-IQ4_XS	365 GB	4-bit	Near-indistinguishable on most tasks.
UD-IQ4_NL	373 GB	4-bit	Near-indistinguishable.
UD-Q4_K_S	436 GB	4-bit	Excellent.
UD-Q4_K_M	466 GB	4-bit	Excellent.
UD-Q4_K_XL	467 GB	4-bit	**The quality sweet spot.**
UD-Q5_K_S	527 GB	5-bit	Diminishing returns start here.
UD-Q5_K_M	561 GB	5-bit	Marginal gain over Q4.
UD-Q5_K_XL	562 GB	5-bit	Marginal gain.
UD-Q6_K	626 GB	6-bit	You won’t tell it from BF16.
UD-Q6_K_XL	684 GB	6-bit	Same.
Q8_0	801 GB	8-bit	Effectively lossless, effectively pointless for local.
UD-Q8_K_XL	820 GB	8-bit	Same.
BF16	1.51 TB	16-bit	Full precision. The thing nobody runs.

Here’s the deal on which one to target. The quality sweet spot is **UD-Q4_K_XL at 467GB** — that’s where the curve flattens and going bigger buys you almost nothing you’d notice. But quality isn’t the only axis; the weights have to fit on hardware you can actually assemble. And 467GB is brutal to feed.

So for most people who will genuinely self-host this, the answer is one rung down: **UD-Q3_K_XL at 343GB.** It still reasons like GLM 5.2, and 343GB is the number that fits the realistic single-box and 4-card paths below where 467GB doesn't. Treat Q3_K_XL as the target and Q4_K_XL as the "if you've got the memory" upgrade.

One disagreement worth flagging, because it comes from the same repo this table does. Unsloth's own guidance points lower than mine: they recommend the 2-bit **UD-IQ2_M (239GB)** as the accessible pick "for best results in terms of accessibility and accuracy," peg it at ~82% accuracy, and say even their 1-bit quant "works well." That's a real position, not a hedge, and for general-purpose use I think they're right — the 2-bit opens up cheaper hardware and holds together. My floor sits a rung higher for one specific reason: coding and multi-step agent work. That's where low-bit quants visibly slip, because the reasoning chains compound small errors and it shows up on real tasks in a way it doesn't in chat. So if your use is general, take Unsloth's 2-bit and save the money. If you're running this as a coding model, I'd still start at Q3.

Remember to add headroom for the KV cache, too. At anywhere near the full 1M context, the cache alone eats tens of gigabytes. The footprint numbers above are weights only.

Hardware paths, cheapest to fastest

Four ways to actually run it. None of them is cheap, and I'll be straight about which ones are miserable.

1. System-RAM offload — possible, painful

You can run GLM 5.2 entirely on CPU and system RAM with llama.cpp, no datacenter GPU required. Load UD-Q3_K_XL (343GB) into a workstation with **384GB+ of DDR5** — think a Threadripper or Epyc board with all channels populated — and it'll generate. The MoE design helps here: only 40B params activate per token, so CPU inference isn't as hopeless as it'd be for a 753B dense model.

The gotcha: it's slow. Expect low-single-digit tokens per second — fine for batch jobs and overnight runs, painful for anything interactive. And 384GB of DDR5 in the current memory market isn't pocket change. This is the "I refuse to buy a GPU for this" path. It works. You won't enjoy it.

2. Mac Studio — the single-box hero (with an asterisk)

Apple Silicon's unified memory is the cleanest way to hold a model this size in one box, because the GPU addresses the whole memory pool. A **512GB M3 Ultra Mac Studio** fits UD-Q3_K_XL (343GB) with real headroom for context, and community reports put a 40B-active MoE in the neighborhood of 15-20 tok/s on that chip — genuinely usable. (I run Linux, not Mac, so treat Apple throughput numbers here as community-sourced, not my own bench.)

The asterisk, and it's a big one: Apple pulled the 512GB configuration in March 2026 amid the DRAM shortage. As of June 2026 you can't order one new — the M3 Ultra tops out at **256GB**. That doesn't reach the Q3 I'd target for coding, but it does run Unsloth's recommended **UD-IQ2_M (239GB)** with a little context headroom to spare, and Unsloth rates that 2-bit as genuinely usable rather than a compromise. So a new 256GB Mac is a real entry point for general use. Stepping up to Q3 means the 512GB unit, and those you now hunt on the used market at a scarcity premium, or wait on whatever Apple ships next. The window on "just buy the 512GB Studio new" closed.

3. Multi-GPU workstation — the real local play

If you want speed and you're buying GPUs, this is the path — but check your assumptions about which GPUs.

The used-3090 stack doesn't work here. The trick that makes 70B and even 235B models cheap — bolt together used RTX 3090s at ~$1,000 each (24GB, 936 GB/s) — hits a wall at this scale. To hold even Q3_K_XL's 343GB you'd need 15 cards. The power, the PCIe lanes, the physical space, the risers — it stops being a workstation and becomes a fire hazard. For models that fit in 4-8 cards, the 3090 stack is still the value king (see the used GPU guide). For GLM 5.2, it's the wrong tool.

What does work: **the RTX Pro 6000 Blackwell, 96GB per card.** Four of them is 384GB — enough for UD-Q3_K_XL with room to breathe, on a single workstation board, at GPU speed. The catch is price. These run roughly $8,500-9,200 street, and NVIDIA's own listing has spiked them to $13,250 as GDDR7 supply tightens. Four cards is a **$34,000-37,000** GPU bill before the rest of the build. That's the honest cost of running this model fast in your own rack. If you want the quality sweet spot instead, six cards (576GB) clears the 467GB Q4_K_XL — call it a $55K+ machine.

4. 8× datacenter node — the fast floor

The top of the ladder is a serving node. **8× H100 (80GB each) is 640GB** of VRAM at ~3.35 TB/s, which tightly holds UD-Q6_K (626GB) — visually lossless output at serving-grade speed. H100s run $25,000-40,000 each to buy, so this is a rental conversation for almost everyone.

One honest correction to a number you'll see repeated: people say "8× H100 runs GLM 5.2 at FP8." The weights at FP8 are ~753GB — that **doesn't fit** in 640GB once you add a KV cache. True FP8 serving needs about a terabyte of VRAM, which means **8× H200 (141GB each, 1,128GB total)** or an MI300X-class node. If you see "8× H100 FP8" quoted as the floor, the math doesn't close. 8× H100 is the floor for a 6-bit GGUF, not for native FP8.

Can't run it? The API off-ramp

Here's the part most local-AI sites won't tell you: for maybe 95% of people reading this, the right move is to not run GLM 5.2 locally at all. A 467GB sweet-spot quant on a $35K box, or a 343GB quant on a Mac you can't buy new, versus an API call that costs less than a coffee per million tokens — the economics only favor local if you have a hard data-residency requirement or you're running enough volume to amortize the hardware.

Because it's MIT-licensed, GLM 5.2 is served by a crowd of providers, and competition has driven the price down hard. Live OpenRouter and Artificial Analysis provider numbers:

Provider	Blended $/1M	Quant	Notes
GMI	$0.72	FP8	Cheapest blended price.
Wafer	$0.79	FP4	Cheap, but lower-precision weights.
DeepInfra	$0.80	FP8	Reliable, good price.
Fireworks	$0.90	FP8	Fast.
Together AI	$0.90	—	Established host.
Z.ai (first-party)	$1 in / $4 out	—	Source provider; ~$1.30 blended.
Makora	$1.30	FP8	Most expensive of the listed set.

Two caveats worth your attention. First, quant matters even on the API: the cheapest routes (Wafer, some GMI tiers) serve **FP4** weights, which is a meaningfully lower-precision model than the FP8 you get from DeepInfra, Fireworks, or Z.ai. "Cheapest" and "the model you benchmarked" aren't always the same endpoint. Second, **data residency.** GLM 5.2 is a Chinese-

origin model, and the first-party endpoint is Z.ai. If your data can't leave specific jurisdictions, route to a Western host (DeepInfra, Fireworks, Together). Or run it locally and keep the tokens on your own metal, which is the actual argument for the $35K box.

Verdict by budget tier

Under $0 of new hardware — use the API. Pick DeepInfra or Fireworks for FP8 at ~$0.80/1M. This is the right answer for almost everyone. Run it locally only if data residency forces your hand.

~$10K, single box — Mac Studio. A used 512GB M3 Ultra runs UD-Q3_K_XL at usable speed in one quiet machine; you're buying on the secondary market now, at a premium, because Apple stopped selling that config. A new 256GB unit is the cheaper way in: it runs Unsloth's recommended UD-IQ2_M, which they put at ~82% accuracy and call genuinely usable. For coding I'd still want the Q3 step and the 512GB unit, but for general use the 256GB Mac is a legitimate entry point, not a consolation prize.

~$35-55K, GPU workstation — 4-6× RTX Pro 6000 Blackwell. Four cards (384GB) for the Q3_K_XL pragmatic target, six (576GB) for the Q4_K_XL quality sweet spot, at full GPU speed. The genuine local-power-user answer if the budget exists. Skip the used-3090 stack here — it doesn't scale to this model.

Datacenter / rental — 8× H100 for a 6-bit GGUF, 8× H200 for true FP8. If you're serving GLM 5.2 to a team or a product, rent the node. Don't believe the "8× H100 FP8" shorthand; that config tops out at a 6-bit quant.

The bottom line: GLM 5.2 is the best open-weights model you can get right now, and the one quant most people should actually target is **UD-Q3_K_XL**. But "you can run it" and "you should run it" are different questions, and for most people the API wins on every axis except control. Run the math on your own volume and data rules before you spend $35,000 to avoid an $0.80 invoice.

> **A note on this guide:** This is not a firsthand benchmark. I haven't run GLM 5.2 on a 512GB Mac or an 8× H100 node — almost nobody has the hardware to. What you're reading is hardware math (weight sizes against memory capacity) plus aggregated capability data, built from the sources cited inline: Unsloth's GGUF sizes, the Z.ai model card, Artificial Analysis benchmarks, and live API pricing. Where a number is community-sourced (Mac throughput) or a common claim doesn't survive the math (8× H100 "FP8"), I've said so. That honesty is the

> whole point — it's what separates this from the scraper sites that'll tell you a 3090 runs a 1.51TB model.

Get notified when we publish new guides.

Subscribe — free, no spam

Source: https://insiderllm.com/guides/run-glm-5-2-locally/