Nineteen gigabytes of my 3090 sat empty. The 3060 kept up.

September 14, 2026 · by Mark Bartlett

Download this post as PDF

InsiderLLM Weekly issue 21 – September 14, 2026

Last week I said Flash-Next was on the 3090 and I would not print a number until the 3060 box had its turn. It has, and the first number is the one I did not expect: at stock settings the more expensive card bought nothing at all.

Quick Hits

- **Stock llama.cpp put 5.4 GB of a 177B model on the 3090 and left the other 19 GB of the card idle.** One flag, `-ncmoe 29`, fills it and buys 34 to 40 percent. 📖 The Flash-Next bench.
- **A real 32 GB machine runs Flash-Next at 8.7 tok/s, not the 22 in the video,** and every decoded token pulls 25 MiB off the SSD to do it. 📖 The Flash-Next bench.
- **A 4 GB GTX 1650 runs a 35B MoE at 20 tok/s** with 38 of 40 expert layers in DDR4-2133 and the disk reading nothing. The 3060 in the same slot: 28 at the same flag, 39 tuned. 📖 The 1650 vs 3060 bench.
- **The MoE primer told you to run dense since February. Our own numbers said otherwise.** Rewritten, with the correction dated at the top. 📖 The rewritten primer.

The Card Is the Flag. The RAM Is the Purchase.

Qwen3.8-Flash-Next is Qwen's open-weight preview of the Qwen 4 architecture: 177B parameters on disk, 6B active per token. A video went round claiming 22 to 24 tok/s on a used 3060 with 64 GB of RAM, and 22 again with the container capped at 32 GB. I ran the same quant on both bench rigs, stock llama.cpp v0.4.0, one day each.

Stock means `--cpu-moe`: the dense layers, the shared experts and the output head go on the card, and every routed expert goes to system RAM. On the 3090 that is **5,368 MiB of 24,576**. The card did 16.9 tok/s, and a 3060 with the same RAM has no reason to do worse, because the card is not doing the work. DDR4 is feeding a 45 GiB expert set to the GPU one token at a time.

`-ncmoe N` keeps the first N layers' experts on the CPU and lets the rest onto the card. The model has 48 layers, so stock is 48. I stepped it down: 40, 34, 29, and at 26 the loader asks for a 25 GB buffer and CUDA says no. **Twenty-nine is the last value that fits**, at 23,652 MiB, and it decodes at **22.6 tok/s** at zero depth and **24.5** with 4,096 tokens of context in place: 34 and 40 percent over stock. Per-token time drops about 0.8 ms for every layer that leaves the CPU, and a constant saving in milliseconds is a growing gain in tokens per second, so the curve steepens as you go. Fill the card and stop.

Now the 32 GB machine. My 3060 box is an i7-7700 with 31 GB of DDR4-2133 and a budget NVMe, which is the box a used 3060 usually lives in. The 45.3 GiB expert set does not fit, and the SSD counters say what happens next: **25 to 28 MiB read per decoded token, at 220 to 250 MiB/s, for the whole run.** Stock decode is **8.7 tok/s**. Seven expert layers on the card, the most the 3060 takes, gets it to 11.1 to 11.5. Prefill on a 2,343-token prompt is 31 tok/s, because each prompt reads 27 to 57 GB, more than the whole expert set, once: the cache evicts the front while the back streams in. I expected the third rep to settle. It did not. There is no steady state when the set is one and a half times the RAM.

Which is why the video's 32 GB number is not this number. A container capped at 32 GB limits what the process holds resident. The page cache belongs to the host kernel, and if the host has 64 GB the whole expert set is sitting in it, warm, on the other side of the cap. That 22 is a 64 GB machine's number. A box where the cache is the RAM says 8.7.

So the prediction, untested and labelled as such: a 3060 with 64 GB beats a 3090 with 32 GB on this model. Nineteen layers on the card do not make up for 25 MiB per token off an SSD.

📖 **Both rigs, every sweep row, the iostat traces and the `--load-mode none` attempt that swapped: here.**

What a $60 Card Buys on a 35B MoE

Friday I put the GTX 1650 in the 3060's slot. Same box, same 32 GB of DDR4-2133, same Qwen3.6-35B-A3B file, same build, same afternoon. Only the card changed.

With every routed expert in RAM, `-ncmoe 40`, the 1650 decodes at **20.2 tok/s** and the SSD reads nothing. Two creators had reported 17 on a 6 GB GTX 1060, and a 4 GB card from 2019 clears that with nothing on it but attention, the shared expert and the head. Two expert layers fit beside those and take it to 20.6. A third fails to allocate a 283 MiB compute buffer, and that is the ceiling.

The 3060 at the same flag does **28.1**, 39 percent faster with nothing on it, and I had half-written into this site that at full offload the number is the RAM's number and the card is a passenger. It is not. Prefill is where it shows: 65 tok/s on the 1650 against 242. Let the 3060 take sixteen layers, `-ncmoe 24`, and it does **38.6**. The gap opens to 87 percent, and the whole reason is layer count: sixteen against two. RAM sets the floor. The card sets the ceiling.

One video claim tested on the way past: `--load-mode none`, which pulls the file into process memory instead of mapping it, was said to be worth three and a half tokens a second. On the 1650 it moved decode **0.3 percent** and prefill 6.8 percent, and load went from under five seconds to 28. On Monday the same mode on the 3090 box with an 82 GB file swapped and produced bimodal samples. **When the file fits in RAM, `none` is a prefill setting worth a few percent. When it does not, it is a liability, and mmap is the only mode that runs.**

📖 **The full comparison, both load modes, and the sm_75 caveat on the 1650: here.**

The Primer Was Wrong, and Now It Says So

From February to this week, the MoE explainer on this site told you that below 48 GB of VRAM dense models win, that MoE is "almost never the right choice" on a 24 GB card, and under the decision framework, in those words, "Run dense models. Period." Every MoE it named was Mixtral-era and none of it knew about expert offload. Meanwhile our own benchmarks page had Qwen3.6-35B-A3B loading whole on the 3090 at 157.7 tok/s against 41.8 for the best dense 27B that fits the same card, and this week added a 177B MoE running on a 12 GB 3060. The two pages contradicted each other for months and I did not notice, because nobody re-reads their own explainer. The page is rewritten, the correction sits in a dated block at the top saying exactly what it used to claim, and the three cases where dense still wins are named with numbers.

📖 **The rewrite, correction block first: here.**

Follow-ups

PAIR's 120-second first-byte timeout is filed upstream. NVIDIA/Personal-AI-Router#57, 9 September, with the repro from last issue, the 20-versus-8 queue-depth confirmation, and the log lines. No setting changes it as of v0.1.1.

The llama.cpp runtime-scale finding narrowed. The ten-seeds piece said llama.cpp's runtime adapter application moved predictions even with every adapter's scale pinned to zero. Re-run on

v0.4.0, the drift is prompt-cache reuse, not the scale path, and it goes to zero with `cache_prompt: false`. The numbers are added to llama.cpp #26207, the existing thread on that behaviour. Two documented zero states still do not reach zero, and that is a separate bug, filed as llama.cpp #28674. The b10088 instability did not reproduce on the new build.

The v0.4.0 pin is now confirmed on both rigs. Last week's gate was two canonical 3090 rows. Friday's 3060 run reproduced the July b10088 rows on v0.4.0 at 28.12 against 28.1 and 38.55 against 38.9, both within 1 percent.

Two Notes, No Articles

DeepSeek V4.1-Flash open weights are out, and they are 475 GiB. MIT licence, 552B parameters, and a tech report titled "Pushing the Limits of KV Cache Compression" that gets the global KV cache to 890 bytes per token, a quarter of V4-Flash. That is real work, and it cuts the wrong memory for a home box. The KV cache is what a long context costs you. The weights are what a 32 or 64 GB machine is short of, and 475 GiB of them are 2.8 times the V4-Flash checkpoint, which a Colibri contributor ran at 0.87 tok/s with a 108 GiB RAM budget, on the one engine that loads it. Nothing loads V4.1 yet; the llama.cpp converter PR has no runtime half. I am not downloading it and neither should you. Wait.

DDR4 pricing. CXMT reached 10 percent of global DRAM revenue in the second quarter, up from 4 percent a year ago, and the bulk of what CXMT ships is DDR4, which is the tier most readers of this newsletter are on. It is the first place Chinese supply shows up in a part you would actually buy. Set against that, CXMT's own guidance says the shortage runs through the second half of 2026. Everything above says RAM is the purchase, so the temptation is to hold out for the price to break. Do not. Watch it, and if a 64 GB kit at a price you can live with appears, take it.

Housekeeping. The benchmark dataset is at **134 rows**, 16 of them new this week on v0.4.0: eight from the Flash-Next sweeps on both rigs, four from the 1650 and four from the 3060 on the 35B MoE. Every earlier row keeps its b10088 tag. CC BY 4.0, attribution the only ask.

📖 **The dataset, and the raw JSON: here.**

That's the week. If you have a 24 GB card and a MoE with its experts in RAM, step `-ncmoe` down until the loader refuses, then go back one. If you are pricing a second card, price a RAM kit first and see if the question is still there.

New here, reading this on the web? Subscribe and the next one lands in your inbox.

— Mark, InsiderLLM

Run Flash-Next on a 3060 with 64 GB? Got a 1060 to put through this harness? Reply, or hit me at hello@insiderllm.com. I read everything.

Source: https://insiderllm.com/blog/newsletter-2026-09-14/