GPUs without FP4, FP8 or BF16: *prefill pays, decode doesn't.*

Newer GPUs add hardware for smaller number formats — FP8 on Ada and Hopper, FP4 on Blackwell — and older cards like the RTX 3090 have neither. We measured what that costs on six GPUs with one model, changing only the matrix-multiply kernel on the same card and the same weights, so the number format is the only thing that moves.

Bottom line

- **For one user generating text, the number formats don't matter.** On the same card and the same weights, FP4 tensor cores against the path a card without them runs: 0.97–1.00×. FP8: 0.97–0.98×. BF16 against FP16: 0.99–1.00×. That speed is set by how fast the card reads its memory.
- **The RTX 3090 — no FP4, no FP8 — generates faster than the RTX PRO 4500 Blackwell** on the same NVFP4 checkpoint: 86.0 against 82.4 tok/s, because it reads memory faster (888 against 770 GB/s).
- **Prompt processing and many users at once are where FP4 and FP8 pay.** Native FP4 prefills an 8,192-token prompt 1.88–2.26× faster than the no-FP4 path on the same card, FP8 1.43–1.66×; with 64 users at once FP4 delivers 1.25–1.58×. On a plain transformer (Qwen3-8B) the FP4 prefill gain is 2.34×.
- **No tensor cores at all is the real cliff — and it is prefill, not decode:** llama.cpp built for Pascal-class GPUs prefills 2.20–2.35× slower on the same card and decodes at 0.95–1.00× of native speed.
- **The old card's path is slightly better quality.** Native FP4 squeezes the activations to 4 bits as well; the no-FP4 path keeps them at full precision. Divergence from BF16 (KL): 0.024 against 0.041. In llama.cpp the native FP4 path costs +12–13% perplexity on MXFP4 files and +2% on NVFP4. Task accuracy did not move: GSM8K 56–59 of 60 on every configuration.
- **On a 24 GB card the bigger limit is memory capacity.** In BF16 the 3090's KV cache holds 52,130 tokens against 266,612 on the 32 GB RTX PRO 4500; an FP8 or FP4 checkpoint lifts it to 203,062–267,605.

1 · Why racing an old card against a new one can't answer this

Two different things change between GPU generations. One is *which number formats the tensor cores accept*: Ampere (the RTX 30 series) does FP16, BF16 and INT8; Ada and Hopper added FP8; Blackwell added FP4. The other is *the silicon*: memory bandwidth, the number of streaming multiprocessors, clocks and cache. An RTX 3090 against an RTX 5090 changes both at once, so the result can't say which one mattered.

The work itself also splits two ways. **Generating text for one user** reads every weight of the model once per token and does very little arithmetic with each byte, so the speed is set by memory bandwidth — faster math units can't help a card that is waiting on memory. **Processing a prompt** (prefill) reuses every weight for thousands of tokens at once, and **serving many users** batches their tokens together; both turn the same work into a math problem, which is where the low-precision tensor cores earn their keep.

One more detail matters: a card without FP4 hardware can still *store* 4-bit weights. vLLM runs an FP4 checkpoint on an RTX 3090 through a kernel called Marlin, which keeps the weights in 4 bits in memory and unpacks them to BF16 for the math. So the old card still reads about a quarter of the bytes; what it lacks is the ability to do the multiplication itself in FP4.

So we split the gap. **The feature:** on one Blackwell card, run the same checkpoint once on the FP4 tensor cores and once through Marlin — exactly the path a 3090 takes — and nothing else changes. **The silicon:** run the same Marlin path on the 3090 and on the Blackwell card. **The real gap** an owner sees is the product of the two.

2 · What we ran

- **Model:** `Qwen/Qwen3.5-9B` @ `c2022362` (Apache-2.0), a 9B model whose 32 layers are 24 Gated DeltaNet and 8 attention layers. The same weights in every format: BF16; FP8 and INT8 W8A8 (`RedHatAI`); NVFP4 W4A4 (`ig1`); GGUF (`unsloth`), plus dense MXFP4 and NVFP4 GGUFs we made from the BF16 GGUF, byte-identical on every machine. A control on a plain transformer: `Qwen/Qwen3-8B` with NVIDIA's NVFP4 and Qwen's FP8 checkpoints, on one RTX PRO 6000.
- **vLLM v0.27.1**, one fresh server per arm. Its `--linear-backend marlin` flag makes a Blackwell card run an FP4 or FP8 checkpoint the way a card without those units does; the server's own log names the kernel each arm actually got.
- **llama.cpp b11696**, compiled six ways. llama.cpp picks its GPU code from the newest architecture it was compiled for, not from the card, so a build compiled for Pascal (`compute_61`: no tensor cores, no fast FP16, no BF16) runs Pascal-era code on a Blackwell card. We built Pascal, Turing, Ampere, Ada and Blackwell versions plus the stock release.
- **What we measured:** one-user decode (512 tokens, median of 5); prefill as time to first token on prompts of exactly 1,024, 8,192 and 32,000 tokens (median of 5); throughput with 1 to 64 users at once (1,024 tokens in, 256 out, 3 requests each); quality as divergence from the same card's BF16 run on 59 fixed continuations, plus 60 GSM8K problems; and for llama.cpp, llama-bench and perplexity.
- **Pre-registered:** the design and eight numeric predictions were written down before the first measurement; section 10 grades them.

Card	Arch	Read GB/s	BF16 TFLOPS	FP8	NVFP4	INT8	FP4 via Marlin
RTX 3090	sm_86	888	75	—	—	196	74
RTX PRO 4500 Blackwell	sm_120	770	133	270	521	—	133
RTX 5080	sm_120	910	124	241	805	—	122
RTX 5090	sm_120	1,813	242	480	1,401	—	246
RTX PRO 6000 Blackwell	sm_120	2,013	415	748	1,480	—	405

Measured on each card through vLLM's own kernels: a 1 GiB read for bandwidth; the best TFLOPS at the model's layer shapes with 1,024+ tokens, including the activation quantization. — = the engine has no kernel for that format on that card. The RTX 5090 runs a permanent memory overclock and the RTX PRO 6000 memory and core; same-card ratios are unaffected. The DGX Spark ran the llama.cpp part only.

- **Without FP4 or FP8 units, a card computes at its BF16 rate**: the Marlin column equals the BF16 column on every card.
- **The RTX PRO 4500 Blackwell reads memory more slowly than the RTX 3090** (770 against 888 GB/s) — which makes the pair a near-equal-bandwidth comparison of a card with FP4 and FP8 and a card without.
- **GeForce Blackwell runs BF16 at half the workstation rate but FP4 nearly in full**, so FP4's advantage over BF16 is 5.8× on an RTX 5090 and 3.6× on an RTX PRO 6000.
- **The 3090's INT8 path runs at 2.6× its BF16 rate** — while on Blackwell this vLLM release's default INT8 W8A8 kernel refuses to run (section 9).

3 · FP4: the same card, the same checkpoint, only the kernel changes

Card	Decode, no FP4	Decode, FP4	Ratio	Prefill 8K, no FP4	Prefill 8K, FP4	Ratio	64 users, no FP4	64 users, FP4	Ratio
RTX 3090	86.0	—	—	4,518	—	—	738	—	—
RTX PRO 4500 Blackwell	82.0	82.4	**1.00×**	7,291	14,049	**1.93×**	1,037	1,508	**1.45×**
RTX 5080	97.1	96.8	**1.00×**	7,324	16,572	**2.26×**	678	848	**1.25×**
RTX 5090	179.2	173.2	**0.97×**	14,554	31,412	**2.16×**	2,243	3,535	**1.58×**
RTX PRO 6000 Blackwell	190.7	184.3	**0.97×**	22,415	42,074	**1.88×**	3,022	4,157	**1.38×**

Tokens per second. No FP4 = the NVFP4 checkpoint through Marlin (`MarlinNvFp4LinearKernel`, weight-only); FP4 = the FP4 tensor cores (`FlashInferCutlassNvFp4LinearKernel`). The RTX 3090 has no FP4 unit: vLLM chose Marlin on its own and said so in its log.

1. **Decode doesn't move** (0.97–1.00×); on three of four cards the Marlin path is marginally faster for one user.
2. **Prefill roughly doubles** (1.88–2.26× at 8K). At 32K the gain is still 1.73–2.02×: the attention and DeltaNet work that FP4 does not touch grows with the prompt, but in this model it stays small.
3. **Many users gain less than prefill** (1.25–1.58× at 64 users). The RTX 5080's 16 GB leaves room for only 39,718 tokens of KV cache, so its throughput tops out early either way.

Control — a plain transformer. Qwen3.5-9B spends part of its time in DeltaNet layers, which FP4 does not speed up. Qwen3-8B has attention in every layer and no DeltaNet, so more of its time is matrix multiplication:

Qwen3-8B, RTX PRO 6000	Decode, no unit	Decode, unit	Ratio	Prefill 8K, no unit	Prefill 8K, unit	Ratio	Ratio at 32K	Ratio, 64 users
FP4 (NVFP4, ModelOpt)	262.1	250.2	0.95×	20,924	48,981	**2.34×**	1.68×	**1.76×**
FP8 (block, CUTLASS forced)	184.9	175.5	0.95×	18,527	29,797	**1.61×**	1.36×	**1.23×**

NVIDIA's NVFP4 checkpoint switched vLLM to an FP8 KV cache by itself (its config asks for it); both of its FP4 arms share that, so the FP4 ratio is clean. vLLM's default kernel for block-FP8 checkpoints (DeepGEMM) fails on this card, so the FP8 row forces CUTLASS.

FP4 is worth 2.34× at 8K on the plain transformer against 1.88× on the hybrid, on the same card — and again no gain for one user (0.95×).

4 · FP8 and BF16

Card	Decode, no FP8	Decode, FP8	Ratio	Prefill 8K, no FP8	Prefill 8K, FP8	Ratio
RTX 3090	69.8	—	—	4,594	—	—
RTX PRO 4500 Blackwell	66.0	64.6	**0.98×**	6,491	10,802	**1.66×**
RTX 5090	146.4	142.0	**0.97×**	14,553	21,223	**1.46×**
RTX PRO 6000 Blackwell	156.8	151.9	**0.97×**	21,952	31,428	**1.43×**

No FP8 = the FP8 checkpoint through FP8 Marlin (weight-only); on Blackwell this needs `VLLM_TEST_FORCE_FP8_MARLIN=1` as well as the flag. On the 3090, vLLM swaps the scheme to weight-only Marlin by itself. The RTX 5080 could not hold the FP8 checkpoint at this configuration.

Card	Decode FP16	Decode BF16	Ratio	Prefill 8K FP16	Prefill 8K BF16	Ratio	64 users, ratio
RTX 3090	47.6	47.7	1.00×	4,670	4,639	0.99×	1.00×
RTX PRO 4500 Blackwell	46.0	45.9	1.00×	6,664	6,924	1.04×	1.02×
RTX 5090	105.9	106.4	1.00×	14,065	14,372	1.02×	1.01×
RTX PRO 6000 Blackwell	111.3	110.4	0.99×	21,965	22,461	1.02×	1.01×

BF16 against FP16 is the question for cards older than Ampere (Turing and Pascal have no BF16); on these cards both run at the same rate. Quality held as well: divergence from BF16 0.0002.

FP8 behaves like FP4, at about half the effect: decode 0.97–0.98×, prefill 1.43–1.66×, 64 users 1.20–1.33×. **A missing BF16 unit costs nothing measurable** on this model (0.99–1.00× decode, 0.99–1.04× prefill); the known risk of running a BF16 model in FP16 is numeric overflow, and Qwen3.5-9B did not overflow.

5 · No tensor cores: llama.cpp compiled for older GPUs

Card	Native prefill	Native decode	Prefill, Pascal code	Turing code	Ampere code	Decode, Pascal code	Turing code	Ampere code
RTX 3090	4,701	123.9	**0.43×**	0.91×	1.00× (native)	1.00×	0.99×	1.00× (native)
RTX PRO 4500 Blackwell	5,727	118.3	**0.43×**	0.50×	0.98×	0.97×	0.97×	0.97×
RTX 5080	7,918	143.7	**0.43×**	0.48×	0.98×	0.97×	0.96×	0.97×
RTX 5090	13,136	253.0	**0.44×**	0.52×	1.04×	0.95×	0.95×	0.95×
RTX PRO 6000 Blackwell	15,482	270.1	**0.46×**	0.51×	1.03×	0.96×	0.94×	0.96×
DGX Spark (GB10)	2,964	41.2	—	0.54×	1.00×	—	0.98×	1.01×

Q4_K_M, llama-bench pp512 and tg128 at depth 0, mean of 5, as a ratio to the card's own native build. Pascal code = no tensor cores, no fast FP16, no BF16; Turing code = FP16 and INT8 tensor cores, no BF16. Every row except the 3090's native build is EMULATED: modern silicon running older code. That isolates the instruction set, not a real 10- or 20-series card. Every build agreed closely with the native one (mean KLD 0.0054–0.0066 for Pascal code).

- **Without tensor cores, prefill runs at 0.43–0.46× of native speed on every card we could build it for**, including the real 3090 running Pascal code — and decode at 0.95–1.00×. (The DGX Spark's CUDA 13 can no longer compile for Pascal.)
- **On Blackwell silicon, Turing-era code reaches only about half of native prefill**; the Ampere-era instruction path recovers the rest. On the 3090 itself, Turing code already reaches 0.91×.
- **FP4 and FP8 hardware add nothing for the K-quant GGUFs most people run:** Ampere code runs Q4_K_M at 0.98–1.04× of native prefill on Blackwell, and Q8_0 at 0.99–1.04×.
- The Ada build matched the Ampere build within 0.6 % on Q4_K_M, Q8_0 and MXFP4, but prefilled NVFP4 files 9–13 % faster; NVFP4's block scales are FP8, which Ada-and-later convert in hardware (our explanation, not traced in the source).

6 · The real gap: the RTX 3090 against Blackwell

Card	Best one-user decode	Best prefill, 8K	Best, 64 users	KV cache in BF16, tokens
RTX 3090	86.0 (NVFP4 via Marlin)	7,789 (INT8)	1,037 (INT8)	52,130
RTX PRO 4500 Blackwell	82.4 (NVFP4, FP4 units)	14,049 (NVFP4, FP4 units)	1,508 (NVFP4, FP4 units)	266,612
RTX 5080	97.1 (NVFP4 via Marlin)	16,572 (NVFP4, FP4 units)	848 (NVFP4, FP4 units)	does not fit
RTX 5090	179.2 (NVFP4 via Marlin)	31,412 (NVFP4, FP4 units)	3,535 (NVFP4, FP4 units)	263,136
RTX PRO 6000 Blackwell	190.7 (NVFP4 via Marlin)	42,074 (NVFP4, FP4 units)	4,157 (NVFP4, FP4 units)	2,063,391

Each card's best arm for each metric, the format in brackets: what an owner would get. Tokens per second.

Blackwell card	Metric	Real gap vs the 3090	from the FP4 unit	from the silicon
RTX PRO 4500 Blackwell	one-user decode	**0.96×**	1.00×	0.95×
	prefill, 8K prompt	**3.11×**	1.93×	1.61×
	64 users	**2.04×**	1.45×	1.41×
RTX 5080	one-user decode	**1.13×**	1.00×	1.13×
	prefill, 8K prompt	**3.67×**	2.26×	1.62×
	64 users	**1.15×**	1.25×	0.92×
RTX 5090	one-user decode	**2.02×**	0.97×	2.08×
	prefill, 8K prompt	**6.95×**	2.16×	3.22×
	64 users	**4.79×**	1.58×	3.04×
RTX PRO 6000 Blackwell	one-user decode	**2.14×**	0.97×	2.22×
	prefill, 8K prompt	**9.31×**	1.88×	4.96×
	64 users	**5.63×**	1.38×	4.10×

NVFP4 checkpoint. Real gap = the Blackwell card on FP4 ÷ the 3090; FP4 unit = FP4 ÷ Marlin on the Blackwell card; silicon = the Blackwell card's Marlin ÷ the 3090's Marlin. The first equals the product of the other two exactly.

- **Against the RTX PRO 4500** — the card with less bandwidth — the 3090 decodes faster in every format and loses prefill by 3.11×, of which 1.93× is the FP4 unit and 1.61× the silicon.
- **Against the big cards, the gap is mostly silicon:** on prefill the RTX PRO 6000 is 9.31× ahead, and only 1.88× of that is FP4.
- **Capacity is the 3090's constraint for serving:** with BF16 weights its 24 GB leaves a 52,130-token KV cache, and with 64 users the median wait for a first token was 27 s. An FP8 or FP4 checkpoint fixes it.

- **INT8 is the 3090's best prefill:** 7,789 tok/s at 8K (1.68× its BF16), above the RTX PRO 4500's BF16 (6,924), at some cost in fidelity (section 7).

7 · Quality

Path	Divergence from BF16 (KL)	Note
BF16, run again	0.0000	the noise floor: all 59 reference continuations reproduced token for token
FP16 (no-BF16 cards)	0.0002	real, but about 200× smaller than FP4's
FP8 via Marlin (no-FP8 cards)	0.0019	activations stay BF16
FP8 tensor cores	0.0033	activations quantized to FP8 too
INT8 W8A8 (3090)	0.0178	the 3090's fast prefill path costs some fidelity
NVFP4 via Marlin (no-FP4 cards)	**0.0239**	activations stay BF16
NVFP4 tensor cores	**0.0406**	activations quantized to FP4 too (W4A4)

Mean KL divergence per token from the same card's BF16 run, measured on 59 fixed 128-token continuations scored teacher-forced over the top 20 tokens (median over three Blackwell cards; INT8 on the 3090).

Card	GGUF	Prefill, native ÷ Ampere code	Decode	PPL native FP4	PPL Ampere code	Change
RTX PRO 4500 Blackwell	MXFP4	**1.54×**	1.03×	16.84	14.97	**+12.5 %**
RTX PRO 4500 Blackwell	NVFP4	**2.29×**	1.05×	14.36	14.06	**+2.2 %**
RTX 5080	MXFP4	**1.42×**	1.03×	16.83	14.95	**+12.5 %**
RTX 5080	NVFP4	**2.27×**	1.03×	14.27	14.05	**+1.6 %**
RTX 5090	MXFP4	**1.28×**	1.05×	16.74	14.94	**+12.1 %**
RTX 5090	NVFP4	**1.83×**	1.06×	14.34	14.04	**+2.1 %**
RTX PRO 6000 Blackwell	MXFP4	**1.33×**	1.05×	16.88	14.97	**+12.8 %**
RTX PRO 6000 Blackwell	NVFP4	**1.91×**	1.07×	14.38	14.06	**+2.3 %**
DGX Spark (GB10)	MXFP4	**1.34×**	1.02×	16.73	14.96	**+11.9 %**
DGX Spark (GB10)	NVFP4	**1.81×**	1.02×	14.41	14.06	**+2.5 %**

llama.cpp, the same FP4 GGUF on the same card. The native Blackwell FP4 path quantizes activations to FP4 as well (W4A4); older code keeps them at 8 bits (W4A8). Perplexity on a fixed 16K-token text.

Speed and quality pull in opposite directions here: the native FP4 path is the faster one at prefill and the less faithful one. The difference is small next to the quantization itself, and it did not show up in task accuracy: GSM8K stayed at 56–59 of 60 on every configuration.

8 · What this means if you own an older card

- **Chatting, one user at a time:** memory bandwidth is what you are buying. An RTX 3090 holds its own against a current mid-range Blackwell workstation card, and FP4 or FP8 checkpoints still run on it — they save memory and bytes read, even without the math units.
- **Long prompts, document work, agents that read a lot:** prefill is where newer cards pull away — about 2× from FP4 units on the same silicon, more from the bigger chips.
- **Serving many people:** FP4 units are worth 1.25–1.58× at 64 users, but on a 24 GB card the KV cache runs out first. Quantize the weights.
- **On a 3090, for prefill-heavy work, try an INT8 W8A8 checkpoint:** 1.68× the prefill of BF16 here.
- **Pascal (GTX 10-series):** current vLLM releases don't support it at all; llama.cpp still compiles for it. In our test, Pascal-era code ran prompts 2.20–2.35× slower than the same silicon's tensor-core code while generating at nearly the same speed; a real 10-series card also has slower silicon, so take that as the part that is down to the missing tensor cores.

9 · What we found in the engines along the way

- vLLM v0.27.1's release builds target compute capability 7.5 and newer — Turing up. Pascal is out.
- INT8 W8A8 does not run on Blackwell in this release: the server selects a CUTLASS INT8 kernel and CUTLASS then refuses ("Int8 not supported on SM120"). Forcing `--linear-backend triton` runs it, slower than FP8 (56.9 against 64.6 tok/s decode on the RTX PRO 4500).
- Marlin refuses NVFP4 weights with INT8 or FP8 activations, so there is no W4A8 path for an NVFP4 checkpoint on older cards.
- FP8 Marlin on a card that has FP8 needs `VLLM_TEST_FORCE_FP8_MARLIN=1`; on cards below compute capability 8.9 vLLM silently replaces an FP8 W8A8 checkpoint's scheme with weight-only Marlin.
- For block-FP8 checkpoints on RTX PRO 6000, vLLM picks DeepGEMM by default, which fails at load ("Unknown SF transformation"); `--linear-backend cutlass` works.
- NVIDIA's ModelOpt NVFP4 checkpoints switch on an FP8 KV cache even under `--kv-cache-dtype auto`.

10 · Our predictions, graded

#	Prediction (written before the runs)	Measured	Verdict
P1	One-user decode unaffected by FP4, FP8 or BF16 (within ±7 %, ±7 %, ±2 %)	0.97–1.00× · 0.97–0.98× · 0.99–1.00×	**held**
P2	FP4 prefill ≥ 2× and FP8 ≥ 1.5× at 8K; the FP4 gain shrinks ≥ 20 % at 32K	1.88–2.26× · 1.43–1.66× · shrinks 16–19 %	**failed, narrowly**
P3	FP4 ≥ 1.3× throughput at 32 users	1.19–1.38×	mixed: 2 of 4 cards
P4	The no-FP4 path stays closer to BF16 than native FP4	KL 0.024 vs 0.041	**held**
P5	3090 decode = RTX PRO 4500 decode × their bandwidth ratio, ±10 %	0.90–0.94 of the prediction	**held**
P6	3090 → 4500 prefill gap = BF16 math ratio × the FP4 gain, ±20 %	3.11× measured vs 3.41× predicted	**held**
P7	No tensor cores: decode ≥ 0.80×; Turing, Ampere and Ada code within ±5 % of native	decode 0.95–1.00×; Turing code 0.94–0.97×	held on 4 of 5 cards
P8	Pascal code ≥ 4× slower prefill than Turing code; Turing → Ampere code < 10 %	1.11–1.17× on Blackwell (2.13× on the 3090); Turing → Ampere 1.96–2.04×	**failed**

Thresholds as pre-registered; nothing was re-fitted. The surprise was P8: we expected the tensor cores themselves to carry llama.cpp's prompt speed, and on Blackwell silicon much of it comes from the Ampere-era instruction path instead.

11 · Caveats

- One model family. The effects scale with how much of a model's time is matrix multiplication: larger on plain transformers (the Qwen3-8B control), smaller on hybrids like Qwen3.5-9B.
- The no-tensor-core and no-BF16 rows are emulated on modern silicon. A real GTX 10- or RTX 20-series card would also be slower for silicon reasons.
- The NVFP4 checkpoint for Qwen3.5-9B is a community quantization; its quality was read against BF16 here, not assumed.
- The RTX 5090 and RTX PRO 6000 run permanent overclocks (memory, and core on the 6000); every same-card ratio is unaffected, cross-card absolutes are not.
- The RTX PRO 4500 runs at a 165 W cap and sat at it in compute-bound arms.
- Throughput was measured at 1–64 users in doubling steps, so a saturation point is located only to within a factor of two; on most cards throughput was still rising at 64.
- GSM8K at 60 items can only catch large accuracy changes; the divergence measurements are the sensitive instrument.

Every figure on this page was computed from the raw rows of runs made on 26–27 October 2026 on our own machines. Run directory: `bench/runs/20261026-precision-gap` (pre-registration with its amendments, the scripts, every per-arm result, server log and kernel line). The rows are kept; to see them, or to send a question you want measured, write to hello@netsphere.com.ai. Online: netsphere.com.ai/reports/fp4-fp8-bf16/