Qwen3.5-4B on the Radeon RX 7900 XT: 11,000 tokens/s INT4 Prefill and 200 tokens/s Decode

Rotated GPTQ-INT4 Inference with INT4 WMMA Kernels and a 236-Launch Decode Path on RDNA3

Sachin Mahesh
sachinmahesh@berkeley.edu
github.com/SILVERTHRONE32

November 2, 2026

Abstract

On a single Radeon RX 7900 XT, Qwen3.5-4B runs at 11,012 tokens/s prefill (pp512) and 200.6 tokens/s decode (tg128) with INT4 weights and 4-bit activations. This is 2.01× the prefill and 1.38× the decode throughput of Q4_K_M built from the same checkpoint and measured on the same build, and it places the RX 7900 XT third of eight GPUs compared (Section 2), ahead of the RTX 5080 in prefill and level with it in decode. The default preset (8-bit activations) reaches 1.30× the prefill and 1.38× the decode throughput of Q4_K_M at the same perplexity degradation (+1.9% against +2.0%) and a lower KL divergence from BF16 (0.024 against 0.028). The path stores every linear weight as a block-Hadamard-rotated, GPTQ-quantized symmetric INT4 tensor and evaluates it on `v_wmma_i32_16x16x16_iu4`; it transfers without modification to a fine-tune of the same backbone (Ossicle-4B), reproducing both the throughput and the accuracy ordering. Weights occupy a single copy in VRAM (peak 3,358 against 3,621 MiB), prefill uses a chunked WMMA gated-delta-net kernel and a WMMA flash-attention kernel for head size 256, and the decode graph was reduced from 847 to 236 kernel launches per token.

1 Introduction

RDNA3 executes `v_wmma_i32_16x16x16_iu4` at twice the rate of its iu8 counterpart. The quantized matrix-multiplication path of llama.cpp (MMQ) widens 4-bit weights to 8 bits and uses the iu8 instruction, reaching 36–40 integer TOPS on this device in our measurements. Using the INT4 pipeline directly requires 4-bit activations, and for unmodified models the channel-wise activation outliers make that unusable. The present work combines three components:

(i) a block-Hadamard rotation applied to both operands, which makes weights and activations close to Gaussian within each 256-element block;
(ii) GPTQ weight quantization in the rotated basis (symmetric INT4, one fp16 scale per 32 columns);
(iii) INT4 WMMA kernels that evaluate 8-bit activations as two INT4 passes and 4-bit activations as one, together with fused prefill and decode paths around them.

We evaluate on the stock Qwen3.5-4B checkpoint (Section 5) and on Ossicle-4B, a fine-tune of the same backbone with an audio encoder, to establish that the result is not specific to one set of weights (Section 5.1). Section 7 analyzes the decode path in detail; Section 8 lists what did not work.

2 Throughput relative to current GPUs

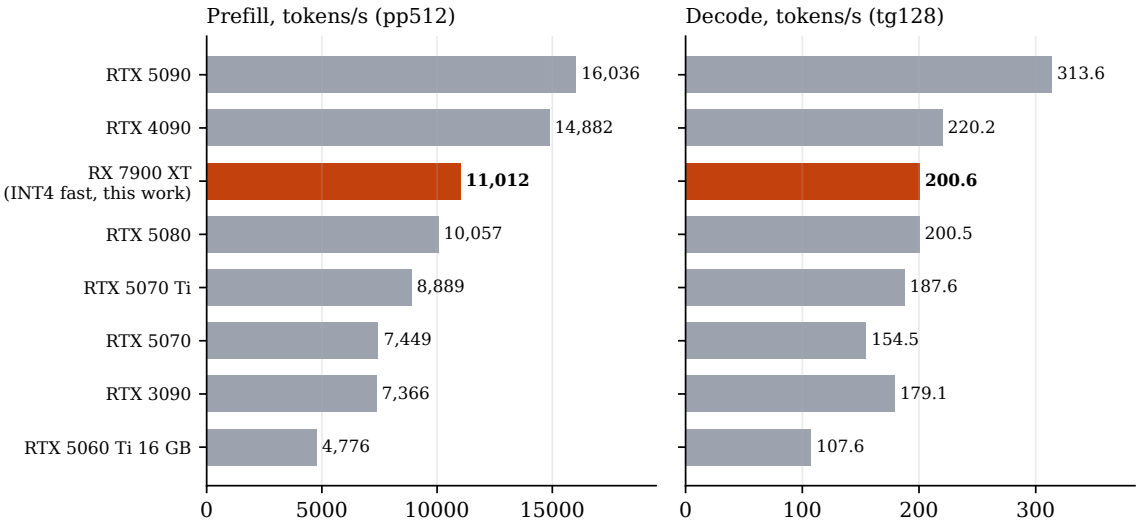


Figure 1: Qwen3.5-4B, 4-bit quantization: prefill and decode throughput by GPU. RX 7900 XT: this work, INT4 `fast`. Other GPUs: Local LLM Advisor leaderboard (llama.cpp, CUDA).

GPU	prefill (tokens/s)	decode (tokens/s)
RTX 5090	16,036	313.6
RTX 4090	14,882	220.2
RX 7900 XT (INT4 `fast`, this work)	**11,012**	**200.6**
RTX 5080	10,057	200.5
RTX 5070 Ti	8,889	187.6
RTX 5070	7,449	154.5
RTX 3090	7,366	179.1
RTX 5060 Ti 16 GB	4,776	107.6

Table 1: Qwen3.5-4B, 4-bit quantization, pp512 and tg128, sorted by prefill throughput.

The RX 7900 XT ranks third of the eight in both prefill and decode, behind the RTX 5090 and RTX 4090. Its prefill throughput is 1.09× that of the RTX 5080 and 1.50× that of the RTX 3090; its decode throughput equals the RTX 5080's (200.6 against 200.5 tokens/s) and is 1.12× the RTX 3090's. Against the RTX 4090 it reaches 0.74× the prefill and 0.91× the decode throughput.

3 Method

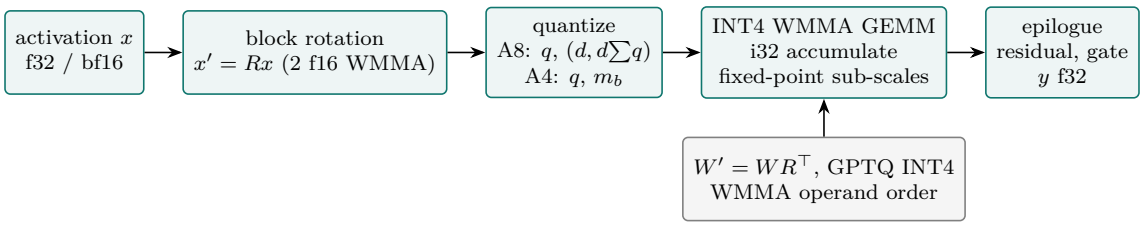


Figure 2: Dataflow of a rotated linear layer. Rotation, quantization and packing form a single fused kernel in prefill and are folded into the preceding normalization (or into the matvec itself) in decode.

3.1 Rotation

For an input dimension K (a multiple of 256) define, per 256-block, $R = \frac{1}{16} H_{256} D$ with H_{256} the Sylvester–Hadamard matrix and $D = \mathrm{diag}(s_j)$, $s_j = -1$ if $((j \cdot 2654435761) \bmod 2^{32}) \gg 31$, else $+1$ (a hash, so converter and backend agree without storing the signs). R is orthogonal, hence $W'x' = Wx$ for $W' = WR^\top$ (applied blockwise along K) and $x' = Rx$. Weights are rotated offline; activations are rotated in the backend as $H_{16} \otimes H_{16}$ on a 16×16 view of each block, i.e. two f16 WMMAs with the sign diagonal folded into the first operand.

3.2 Weight quantization

GPTQ is run in the rotated basis with Hessians accumulated by forward hooks on each of the 200 linear tensors in fp32 (128 sequences $\times$ 512 tokens of WikiText-2 train), Cholesky-inverse error feedback (damping 0.01 of the mean diagonal, column blocks of 128), codes in $[-8, 7]$ and one fp16 scale per 32 columns chosen by an MSE search over 26 candidates $\{0.55, \ldots, 1.05\} \cdot \max|w|/7$ on the error-compensated weights. The result is stored as ordinary Q4_0 blocks; the token embedding/output head is Q6_K. The mean relative weight error is 0.107 for both Qwen3.5-4B and Ossicle-4B. The loader recognizes a rotated file from the GGUF key `ossicle.int4_rotation_block`. GPTQ takes about four minutes and 15 GB of host memory.

3.3 INT4 WMMA execution

Weights are repacked into WMMA operand order with fixed-point integer sub-scales. The repacked layout has exactly the byte size of the Q4_0 tensor, so each weight is converted *in place* at first use (through a temporary copy of one tensor at a time, before any graph is captured) and only one copy resides in VRAM.

A8 (`quality`). An INT8 code $q \in [-128, 127]$ is split as $q = 16h + \ell$ with $h \in [-8, 7]$ signed and $\ell \in [0, 15]$ unsigned. The h pass is accumulated, shifted by 4 bits, and the ℓ pass is accumulated into the same accumulator before the integer sub-scale (fixed-point shift 12) is applied; the cross-unit accumulation is carried in f32 to avoid 24-bit integer overflow. Two INT4 passes cost one INT8 pass, but on the faster pipe: 134 INT4-TOPS effective (74% of the INT4 peak) against 36–40 TOPS for MMQ.

A4 (`fast`). A single pass; the scale of each 256-block is the multiplier $m_b = \lceil 16\, c\, \mathrm{amax}_b / \mathrm{amax}_{\mathrm{token}} \rceil \in \{1, \ldots, 16\}$ of 1/16 of a per-token reference, with clip $c = 0.85$ (KLD 0.113 $\rightarrow$ 0.092 relative to $c = 1$ in the clip sweep; $c = 0.6$ gives 0.146), and weight sub-scales with shift 10. The weight scale group (32/64/128) is detected at load time. A standalone kernel reaches 125 TOPS at scale-per-32 (149 at scale-per-128) against 182 TOPS for a pure-WMMA loop.

preset	activations	selection	decode
`quality`	INT8, two INT4 passes	default	INT8
`fast`	INT4, one pass, clip 0.85	`GGML_INT4_PREFILL=fast`	INT8

Table 2: Presets. Decode (batch ≤ 8) always uses 8-bit activations, so decode throughput does not depend on the preset.

4 Experimental setup

Hardware and software. One Radeon RX 7900 XT (gfx1100, wave32, 20 GB, 254 W cap), ROCm 10.1.0 from the pip wheels; llama.cpp fork `llama.cpp-ossicle`, branch `ossicle`, commit `bb04ca964` (branch `decode/matvec-prologue-attn-fold`, on top of `ad3d82c29`; Section 7). All rows of every table were measured on the same build in one session.

Baseline. Q4_K_M quantized from the same BF16 GGUF with an importance matrix from WikiText-2 train (250 chunks of 1024 tokens).

Accuracy. Mean $D_{\mathrm{KL}}(P_{\mathrm{BF16}} \,\|\, P)$ over positions ("KLD"), perplexity ratio against the model's own BF16 GGUF, and same-top-1 rate, on 24 chunks of WikiText-2 test (context 512, about 12,000 tokens); standard errors are those reported by `llama-perplexity`.

Throughput. `llama-bench`; prefill is the median of 8 runs after discarding the first (cold clocks), decode is `tg128` from an empty context, median of 5. With a 128-token context (`-d 128`) every decode figure is 1.4–2.2 tokens/s higher. Decode drifts by ±1.5 tokens/s with device temperature, so only same-session ratios are interpreted.

5 Results on Qwen3.5-4B

Qwen3.5-4B	pp512	pp2048 tokens/s	tg128	ΔPPL %	KLD	top-1 %	peak VRAM MiB
Q4_K_M	5,477	5,387	145.2	+2.00 ± 0.35	0.0278 ± 0.0011	91.9 ± 0.3	3,621
rotated INT4, `quality`	7,140	7,013	200.7	+1.92 ± 0.40	0.0241 ± 0.0018	92.7 ± 0.3	3,358
rotated INT4, `fast`	11,012	10,624	200.6	+8.67 ± 0.74	0.0969 ± 0.0033	86.1 ± 0.4	3,358*

Table 3: Stock Qwen3.5-4B, one RX 7900 XT. ΔPPL and KLD are against the BF16 GGUF (PPL 8.837). *Measured for `quality`; `fast` uses the same weights.

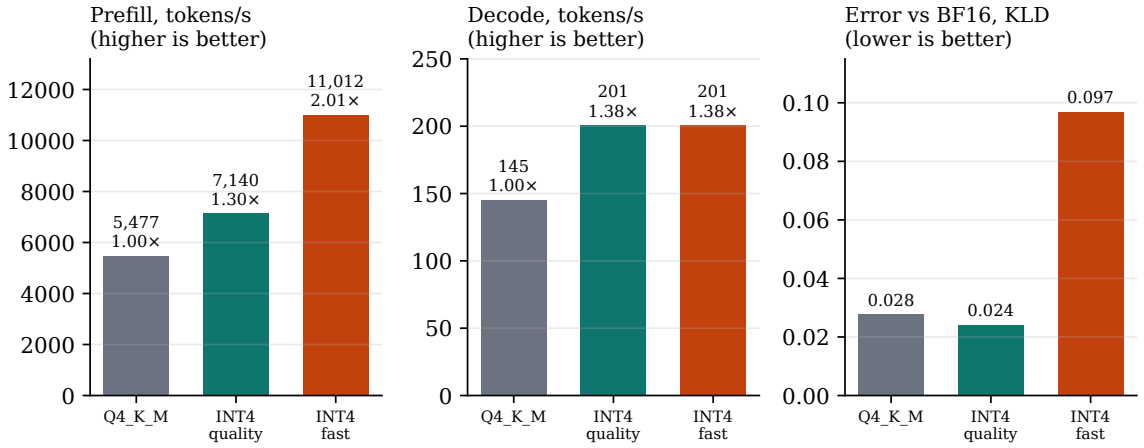


Figure 3: Stock Qwen3.5-4B; ratios are against Q4_K_M. Prefill: pp512. Decode: tg128.

`quality` matches Q4_K_M in perplexity degradation (difference −0.08, well inside the standard errors) with a nominally lower KLD (0.0241 against 0.0278, a gap of about two standard errors unpaired) and a 0.8 pp higher top-1 rate, at 1.30× prefill throughput at both pp512 and pp2048, 1.38× decode throughput and 7% lower peak VRAM. We do not claim an accuracy advantage over Q4_K_M; the result is equal accuracy at higher throughput. `fast` reaches 2.01× (pp512) and 1.97× (pp2048) prefill throughput; its KLD is 3.5× that of Q4_K_M and its perplexity degradation 8.7%. Moving the activations from 8 to 4 bits accounts for the entire difference between the presets (KLD 0.024 → 0.097), consistent with the per-tensor-kind simulation in the technical report, in which the FFN projections and the GDN output projection dominate the 4-bit-activation cost.

5.1 A second model: Ossicle-4B

Ossicle-4B shares the backbone and architecture (24 gated-delta-net and 8 full-attention layers, hidden size 2560) and adds an audio encoder; its language weights are a fine-tune. The identical pipeline (Section 9) was applied to both checkpoints; the only change was extending the GPTQ script to read the original checkpoint's tensor names.

model	format	pp512	pp2048	tg128	ΔPPL (%)	KLD	top-1 (%)
Qwen3.5-4B	Q4_K_M	5,477	5,387	145.2	+2.00 ± 0.35	0.0278 ± 0.0011	91.9
	INT4 `quality`	7,140	7,013	200.7	+1.92 ± 0.40	0.0241 ± 0.0018	92.7
	INT4 `fast`	11,012	10,624	200.6	+8.67 ± 0.74	0.0969 ± 0.0033	86.1
Ossicle-4B	Q4_K_M	5,461	5,356	145.0	+3.51 ± 0.42	0.0311 ± 0.0015	91.2
	INT4 `quality`	7,094	6,992	200.6	+2.02 ± 0.37	0.0250 ± 0.0014	92.3
	INT4 `fast`	10,934	10,606	200.6	+9.11 ± 0.79	0.1006 ± 0.0036	85.0

Table 4: Both models, one pipeline, one build; accuracy against each model's own BF16 original (PPL 8.837 and 9.475).

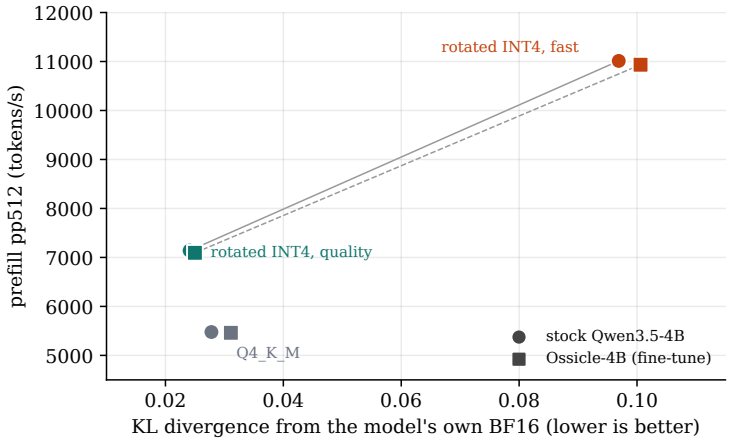


Figure 4: Prefill throughput (pp512) against KLD. Circles: Qwen3.5-4B; squares: Ossicle-4B.

Throughput agrees within 0.7% between the models for every format, as expected for identical tensor shapes. The accuracy ordering is the same: `quality` below Q4_K_M in KLD (ratio 0.87 for Qwen3.5-4B, 0.80 for Ossicle-4B) and `fast` at 3.5× and 3.2× Q4_K_M. Both models are 4B members of one architecture family; other families and sizes were not measured.

5.2 Speculative decoding

Qwen3.5 checkpoints carry a one-layer MTP head, used as a draft model (`--spec-type draft-mtp`) with a 0.50 GB head file (Q8_0 layer, Q4_K embedding/output). With two drafts under `llama-server` and greedy sampling, the output is identical to plain decoding. Ossicle-4B: 190.7 → 229 tokens/s (71% acceptance, build `ae36245f8`); stock Qwen3.5-4B: 168.5 → 218.8 tokens/s (64%, build `72ec3b93e`). These were measured on earlier builds than the tables above and were not repeated on the final one.

6 Prefill path

The GEMM runs at about 74% of the INT4 peak at A8; tile shapes (MI,NJ,waves) $(2, 2, 4)$, $(2, 2, 16)$, $(1, 2, 8)$ differ by under 2%. Beyond the GEMM, a 512-token `fast` pass is dominated by the operations below, each of which was replaced or fused (Ossicle-4B, pp512 on the builds of the time).

- **Chunked WY gated-delta-net on WMMA.** A prep kernel (parallel over chunk×head) forms $A = \mathrm{tril}(\beta_j e^{g_j - g_i} k_j \cdot k_i)$ and $P = \mathrm{tril}(e^{g_t - g_j} q_t \cdot k_j)$ with f16 WMMA and solves $(I + A)X = [\beta v \,|\, \beta e^{g} k]$ in 16-row panels (WMMA for off-diagonal updates, f32 for the diagonal blocks); a scan kernel, sequential over chunks and parallel over heads×16 state columns, keeps the f32 state in WMMA accumulators. 0.46 → 0.14 ms per layer per 512 tokens; relative RMS error against an f64 recurrence 1.6–$2.5 \cdot 10^{-4}$ per layer. A scalar-code version of the same algebra was 9–42% slower than the recurrent kernel; the gain required ISA-level work (branch-free preloads, no `vmcnt(0)` per load).

- **WMMA flash attention, head size 256.** The upstream WMMA path is excluded for RDNA3 at this head size (errors of order 10^{-2}); the new kernel (64 rows = tokens×GQA heads per block, 8 waves with the head dimension split over wave pairs, Q in LDS, K and $V^\top$ tiles through one LDS buffer with conflict-free transposed V writes, mask-driven tile skipping, heaviest causal blocks first) takes a pp2048 pass from 25.5 to 12.0 ms; all 2,920 `FLASH_ATTN_EXT` tests pass.
- **Fusion.** Shared activation packing across sibling matmuls (qkv+z, q/k/v, gate/up), SwiGLU and SiLU·mul inside the pack kernel, the residual add in the GEMM epilogue, GDN α/β (convert, hipBLASLt, convert → one skinny bf16 WMMA kernel with sigmoid / softplus·A epilogue), paired q/k L2 norms, a tiled LDS-transpose conv concat (1.8 → 0.6 ms), and removal of three memsets per GEMM.

	`quality`		`fast`	
step (cumulative)	pp512	pp2048	pp512	pp2048
single-copy weights (baseline of this round)	5,836	5,644	8,212	7,773
chunked WMMA gated-delta-net	6,369	6,106	9,208	8,721
shared packing, GLU in pack kernel, tiled concat	6,744	6,489	10,090	9,501
WMMA attention, GEMM residual epilogue, skinny gate kernel	6,945	6,885	10,457	10,333
gate epilogue, paired L2 norm, 128-thread head norms	7,082	6,991	10,895	10,570

Table 5: Prefill throughput (tokens/s) by optimization step, Ossicle-4B. KLD (0.0250) and MMAU did not change at any step; the fusions are bit-identical to the unfused path.

7 Decode path

A decode token reads 2.53 GB of weights (2.0 GB of body weights, 0.52 GB Q6_K head), i.e. 3.37 ms at 750 GB/s, an upper bound of about 297 tokens/s. The first complete INT4 decode path (not the Q4_K_M baseline) ran at 138.3 tokens/s on an earlier build, at 64% device occupancy, with 847 launches per token and ≈3.1 µs of idle time after each, while the matvecs already streamed at 90% of the bandwidth bound; the work was therefore launch reduction. On the final build, Q4_K_M decodes at 145.2 tokens/s. The matvec (`k_mv`) operates on the repacked layout: one block is 16 weight rows × 10 waves splitting the 256-blocks of K (10 divides the 10 super-blocks of $K = 2560$).

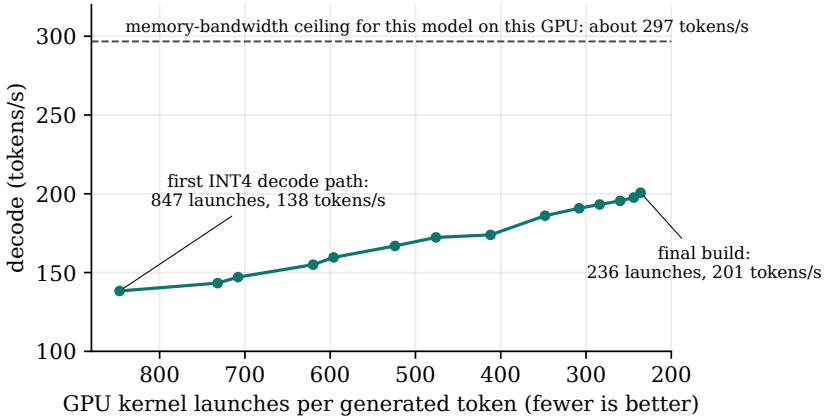


Figure 5: Decode throughput against launches per token (Ossicle-4B and Qwen3.5-4B share the graph; readings from different sessions, empty context). The leftmost point is the first complete INT4 decode path (847 launches, 138.3 tokens/s, earlier build), not the Q4_K_M baseline (145.2 tokens/s on the final build). Dashed: the bandwidth bound.

fusion (cumulative)	launches/token	tokens/s
first complete INT4 decode path (earlier build)	847	138.3
rotation shared by sibling matvecs; residual add in the matvec epilogue	732	143.3
RMSNorm fused with rotation + INT8 quantization; SiLU(z)·norm folded into the `ssm_out` matvec rotation; 10-wave blocks	620	155.0
GDN state read in place from the cache row; conv cluster (gather, concat, cpy, `ssm_conv`, SiLU) as one kernel	524	166.9
GDN α+β in one launch; q/k L2 norm inside the GDN decode kernel; rotation inside the matvec for single-consumer inputs	412	173.9
DPP reductions in the rotate/quantize prologues; attention gate and per-head norm folded into the matvec rotations	348	186.1
q/k RMSNorm + multi-section RoPE + K/V write in one launch per attention layer	308	190.8
conv cluster in the epilogue of the qkv matvec	284	193.2
GDN z-projection shares the qkv launch	260	195.5
attention q, k, v in one launch; register-resident norm kernel; 8-column gate matvec as two launches of 4	244	199.4
matvec prologue and in-kernel attention combine (below)	236	200.7

Table 6: Decode fusion steps (tg128, empty context); readings from different sessions; last row: stock Qwen3.5-4B, `quality`.

7.1 Profile of the final build (236 launches)

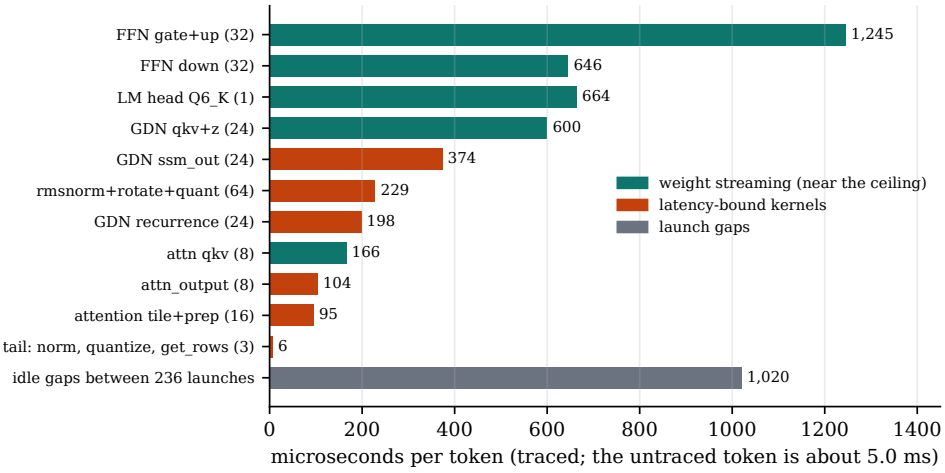


Figure 6: Kernel time per token on the final build (236 launches; stock Qwen3.5-4B, `rocprofv3`, last 100 tokens of one trace; tracing inflates a token by about 7%). Launch counts in brackets.

kernel	calls	µs/call	µs/token	share of busy	GB/s
FFN gate+up (26.5 MB)	32	38.9	1,245	28.8%	683
LM head, Q6_K (521 MB)	1	664	664	15.3%	785
FFN down (13.3 MB)	32	20.2	646	14.9%	658
GDN qkv+z (17.7 MB)	24	25.0	600	13.9%	708
GDN `ssm_out`, in-kernel rotation (5.9 MB)	24	15.6	374	8.6%	379
RMSNorm + rotation + quantization	64	3.6	229	5.3%	–
GDN recurrence	24	8.3	199	4.6%	–
attention q+k+v (14.7 MB)	8	20.8	166	3.8%	710
attention `attn_output`, in-kernel rotation (5.9 MB)	8	13.0	104	2.4%	455
attention tile + prep	16	–	95	2.2%	–

Table 7: Decode kernel profile of the final build (236 launches; one trace, stock Qwen3.5-4B, last 100 tokens): 5,347 µs per traced token, of which 4,327 µs is kernel time and 1,020 µs (19%) idle time between launches. Shares are of kernel time.

Kernels that stream large weights run at 660–790 GB/s. The remainder is launch gaps and two latency-bound projections ($K = 4096$: `ssm_out`, `attn_output`) whose input is the preceding kernel's output. An ablation of `ssm_out`, run before the two changes below (15.8 µs), separates the components: constants in place of weight loads 8.2 µs; no rotation 12.7 µs; neither 3.1 µs (launch, dot product, epilogue). The weight stream alone costs 9.6 µs (615 GB/s against a 7.9 µs bound) and the rotation prologue adds about 2.8 µs that the stream does not hide. Two changes address this and the attention tail; the profile above already includes them.

Matvec prologue. Each of the 160 blocks rotates and quantizes the full activation, one 256-block per wave. The quantizer performed eight dependent max/sum reduction pairs with a divergent lane-0 store between them; the eight reductions are independent and now execute in lock-step on DPP, with a single scale store, and the per-row scale and residual loads are issued at kernel start. Per-call times fall by 0.4–0.6 µs for `ssm_out`, `ffn_down` and `attn_output`, and by 0.15 µs for the 64 normalization kernels (197.2 → 199.4 tokens/s, interleaved A/B).

In-kernel attention combine. The split-KV decode attention launched `flash_attn_combine_results` (2.2 µs plus a 3.1 µs gap) in each of the 8 attention layers. The last block to finish each (tile, head-group) combines the group's partial results: release fence, atomic increment of a counter in a zero-initialized device array that the last block resets (no per-launch memset or allocation), acquire fence, then one warp per head row with lane l reading the metadata of parallel block l and eight contiguous outputs per lane. A first version with dependent scalar coherent loads lengthened the tile kernel by 17 µs and reduced throughput; the warp-per-row version adds 1.9 µs (7.0 → 8.9 µs) and removes the launch. It applies to a single query token, at most 32 parallel blocks and $d_v \bmod 128 = 0$, on HIP only (`GGML_FA_FOLD=0` disables it). Throughput by context depth (tokens): 198.6/199.3 at 0, 200.8/201.5 at 128, 199.2/199.7 at 512, 195.1/195.2 at 2048 and 188.4/188.0 at 6000 (fold off/on), i.e. a gain of 0.5–0.7 tokens/s at short context that vanishes by 2k tokens.

Both changes together: 197.5 → 199.7 (empty context) and 199.4 → 201.6 (128-token context) for Ossicle-4B. Batch-8 and prefill-path KLD/PPL are bit-identical to the preceding commit; at batch 1 PPL changes by +0.0010 and KLD by −0.0002 (rounding-level, from different FMA contraction in the epilogue). Prefill throughput is unchanged. The remaining overhead is ≈1 ms of launch gaps (236 launches), the $K = 4096$ projections, and the GDN recurrence (8.2 µs against about 5.3 µs for its state traffic).

8 Negative and neutral results

experiment	outcome (tokens/s unless stated)
Scalar-code chunked WY gated-delta-net	correct, 9–42% slower than the recurrent kernel
NVFP4/MXFP4 re-gridded to uniform INT4	+0.07 KLD, no gain over BF16 → INT4
Sparsity skipping in the GEMM	exact; 0–6% of real tiles skippable
Scale groups of 64/128	+5% prefill for +0.005–0.015 KLD
Next-block weight prefetch in the in-kernel-rotation matvec	169.6–188.6 against 194.7–196.5 (VGPRs 112 → 128)
Rotation input requested before the weights	195.3 against 197.7
First weight loads issued before constant setup	194.4 against 199.6
Wave count ≠ 10 (rotation matvec: K=4096 at 5, 8, 16; K=9216 at 9, 12, 18)	188–191 against 195.6; gate/up kernel at 1, 2, 5 waves: no gain
Rotation constants as lane masks (∼95 fewer VALU operations)	neutral (199.65 against 199.8): the prologue is latency-bound
Single-block attention at short context (no combine)	slower at every depth (196.8 against 199.4 at depth 0)
Fused FFN, one launch with a grid barrier	191.7 against 196.9
Output norm and quantization fused before the head	193.3 against 198.7
Weight prefetch into the Infinity Cache from a side stream (8/24/40 MB)	181/136/105 against 186

Table 8: Experiments not retained. Every variant that raised the rotation matvec from 112 to 128 VGPRs lost 1–5%.

9 Reproduction on stock Qwen3.5-4B

Requirements: an RDNA3 GPU (tested: RX 7900 XT, ROCm 10.1.0), 15 GB of host memory for GPTQ, about 20 GB of disk, WikiText-2 raw files.

```
# fork with the INT4 path (branch "decode/matvec-prologue-attn-fold" = "ossicle" plus the decode commits),
    built for gfx1100
cmake -S . -B build -DGGML_HIP=ON -DAMDGPU_TARGETS=gfx1100 -DCMAKE_BUILD_TYPE=Release && cmake --build build
     -j

# BF16 GGUF from the unmodified checkpoint
huggingface-cli download Qwen/Qwen3.5-4B --local-dir qwen35-4b
python convert_hf_to_gguf.py qwen35-4b --outtype bf16 --no-mtp --outfile qwen35-4b-bf16.gguf

# GPTQ in the rotated basis (about 4 min, 15 GB RAM; calibration: WikiText-2 train)
python scripts/gptq_rotate.py --src qwen35-4b --out gptq_g32.safetensors --group 32 --calib 128

# rotated GGUF (per-tensor check against the BF16 weights; Q6_K output head)
python scripts/rotate_quantize.py qwen35-4b-bf16.gguf qwen35-4b-rotq4-gptq.gguf --blocks gptq_g32.
    safetensors --embd q6_k

# run
llama-server -m qwen35-4b-rotq4-gptq.gguf -ngl 99                          # quality
GGML_INT4_PREFILL=fast llama-server -m qwen35-4b-rotq4-gptq.gguf -ngl 99    # fast

# measure
llama-bench -m qwen35-4b-rotq4-gptq.gguf -ngl 99 -p 512,2048 -n 128 -d 0,128 -r 9
llama-perplexity -m qwen35-4b-bf16.gguf -f wiki.test.raw -c 512 -ngl 99 --chunks 24 --kl-divergence-base
    base.bin     # BF16 reference
llama-perplexity -m qwen35-4b-rotq4-gptq.gguf -f wiki.test.raw -c 512 -ngl 99 --chunks 24 --kl-divergence-
    base base.bin --kl-divergence
```

The Q4_K_M baseline: `llama-imatrix` on the BF16 GGUF (WikiText-2 train, `-c 1024 --chunks 250`),

then `llama-quantize` with that matrix, measured with the same commands.

10 Limitations

- **Target.** RDNA3 only (gfx1100 tested). The files are not interchangeable with Q4_0: any other backend decodes the rotated codes as unrotated and produces garbage.
- **Scope.** Two models, both 4B members of the Qwen3.5 architecture. Other families and sizes are untested, and the decode analysis is specific to this architecture's layer mix.
- `fast`. 2.0× prefill for ΔPPL +8.7% and KLD 0.097; it is not equivalent to `quality`.
- **GPU comparison.** The comparison with the RTX 5080 and the other GPUs in Section 2 is not like-for-like. The INT4 `fast` path has a higher perplexity and KLD than standard Q4 quantizations (Section 5), and the leaderboard publishes neither its prompt length nor its other benchmark settings (only a Q4 quantization class), so the two sets of throughput figures were not obtained under identical conditions.
- **Calibration and metrics.** GPTQ calibration is text only; accuracy is reported on WikiText-2 (24 chunks), with no downstream benchmark for the stock model.
- **Measurement.** Decode throughput varies by ±1.5 tokens/s with device temperature; baselines are therefore taken in the same session. Decode-path KLD references cover 4 chunks. The in-kernel combine is HIP-only and keeps 16 KB of static device counters. The speculative-decoding figures predate the final build.
- **Untested.** Multi-GPU, repeated load/unload in one process, and partial offload.

Artifacts

Fork `SILVERTHRONE32/llama.cpp-ossicle`, branch `ossicle` (decode series `8546fe90a…ad3d82c29`) and branch `decode/matvec-prologue-attn-fold` (`e96773ca4`, `bb04ca964`: the matvec prologue and the in-kernel attention combine); files `SILVERTHRONE/Ossicle-4B-GGUF` (stock model: `rdna3/qwen35-4b-rotq4-gptq.gguf`). License: Apache 2.0. The full technical report (Ossicle-4B, all experiments and caveats) accompanies this document.

Affiliation

This work was carried out independently of the author's university. UC Berkeley had no role in it, and the university email address is provided for contact only.