

FLE: Fast Local Experts

An 89 GB Mixture-of-Experts model at ~90 tokens/s on one 32 GB GPU

Max Bitzer (sirfyyyn)

AI assistance: engine patches, measurement scripts and this text were written together with Claude (Anthropic, Claude Code). arXiv does not accept an AI system as an author; the author takes responsibility for every claim and number.

Version 1.0.2, 2026-10-10: version 1.0 of 2026-10-09 with film 6 corrected ([Appendix B](#), version 1.0.1), the archive DOI and FLE-IQ3_M-REAP448 in its own model repository (Availability, [Section 9](#)); the text is unchanged otherwise. (Final configuration p49-ring ; final measurement series of 2026-10-06: B1e against upstream, B3e1-B3e5 all quantizations, S against Strata; expert pruning and MKQA knowledge probe of 2026-10-08; FLE-IQ3_M-REAP448 speed, bracket B3e6, of 2026-10-09). Rules for this text: every number is a measurement on our own setup, paired inside one measurement bracket where speed is concerned, with raw data, script and build named in the evidence table ([Appendix A](#)); every paired bracket is listed in [Appendix C](#). Estimates and readings that were not measured are marked as such.

Cite as: Max Bitzer (sirfyyyn). FLE: Fast Local Experts — an 89 GB Mixture-of-Experts model at ~90 tokens/s on one 32 GB GPU. Technical report, 2026. doi:10.5281/zenodo.23272400, <https://doi.org/10.5281/zenodo.23272400>.

Generated on 2026-10-10 from `file-paper.md` (SHA-256 8cad25e7d058).

Abstract

We run Qwen3.8-Flash-Next, a 512-expert Mixture-of-Experts (MoE) language model, on a single workstation with one 32 GB GPU, 120 GB of system RAM and 16 CPU cores. Two pieces make this practical. First, **FLE-IQ3_M** (internally G3), a hybrid quantization that combines the Unsloth UD-IQ4_XS release with expert layers from ISTA-DASLab's GSQ-RCO release: 88.87 GB instead of 93.68 GB, with a KL divergence against BF16 that stays within the measurement noise on three texts, one of them held out. Second, **FLE**, a fork of llama.cpp that borrows ideas and some code from ik_llama.cpp, FreeToken, KTransformers and Strata ([Section 8](#)): a GPU expert cache with frequency-gated admission and oldest-first eviction, overlap of GPU cache hits with CPU work on cache misses, fused CUDA kernels, and speculative decoding with the model's own multi-token-prediction (MTP) head, its experts requantized to IQ4_NL. Against current upstream llama.cpp (commit `a8c9a4e7c`, newer than FLE's base), with the same FLE-IQ3_M file, the same card and the same 16 threads (no expert cache, no MTP), the final FLE configuration decodes **2.49 times as fast** as upstream with micro-batch 1024 (−59.9 % [95 % CI −61.2 .. −58.5] for upstream in a paired bracket, B1e) and 2.78 times as fast as upstream with micro-batch 4096 (−64.0 % [−65.2 .. −62.9]); FLE runs at about 100 tokens/s on these prompts. Prefill is where FLE still loses: upstream with micro-batch 4096 processes 4k and 8.6k prompts 24 % and 12 % faster, and Strata, the closest public engine, 2.3 to 2.7 times as fast, while FLE decodes 23 % faster than Strata. Against our own previous production setup (stock build with Unsloth UD-IQ4_XS, ~56 tokens/s) FLE was +58.5 % faster [+55.8 .. +61.6] already before the last steps (bracket END2). On three benchmarks from Qwen's model card, one run each, FLE-IQ3_M on FLE is level with the card on IFBench (80.3 strict / 83.0 loose against 81.3) and GPQA Diamond (88.9 against 91.7, inside the 95 % interval) and below it on LiveCodeBench v6 (85.5 against 91.9, outside the interval; 10 of the 19 failures are answers cut off at the 81,920-token limit). Besides FLE-IQ3_M we release two smaller hybrids and FLE-IQ3_M-REAP448, with 64 of 512 experts per layer pruned by REAP; a 400-question MKQA knowledge probe shows that KLD on a few texts understates what pruning removes (at 384 experts per layer the KLD rose moderately, while correct answers fell by 5.8 / 11.8 points in German / English). We report the negative results as well: expert prefetching by pre-gating, n-gram drafting, low-rank kernels and fully self-made quantizations all lost in our measurements, and so did copying part of the cache misses over PCIe as Strata does (−4.5 %) and a rebase of the fork onto current upstream (−5.7 %). A re-trained

MTP head (recursive self-distillation) gained +4.54 % only on the tasks its training corpus contained; on 16 unseen tasks the untrained head was +2.95 % [+1.96 .. +3.93] faster than it, so the release ships the untrained head. The speed numbers above were measured with the re-trained head c2 (Section 4.3).

1 Introduction

Large MoE models activate only a few experts per token, so most of their weights are idle at any moment. On a workstation this invites a split: keep what every token needs on the GPU and leave the rarely used experts in system RAM. The split is easy to state and hard to make fast. A token that needs an expert which is not on the GPU waits for the CPU (or for a PCIe transfer), and with ten active experts out of 512 per layer, such misses happen in almost every layer of almost every token.

This report describes how we made that split fast for one concrete model on one concrete machine, and how we checked every step. It is not a general system. It is a measured account of what worked, what did not, and by how much, written so that someone with similar hardware can reproduce it.

Contributions:

1. **FLE-IQ3_M**, a hybrid quantization of Qwen3.8-Flash-Next that is 4.81 GB smaller than Unsloth UD-IQ4_XS at the same measured quality (Section 3).
2. **An expert cache inside llama.cpp** with admission and eviction policies, and overlap of GPU hits with CPU misses (Sections 4.1–4.2).
3. **The model's MTP head in llama.cpp**, its experts requantized to IQ4_NL, with a reduced draft vocabulary, and a negative result: re-training the head by recursive self-distillation on the model's own outputs did not generalize to unseen tasks (Section 4.3).
4. **A measurement method** for speed on a noisy single machine: paired brackets (reference before, candidates, reference after) with drift limits and bootstrap intervals (Section 2.3).
5. **Negative results** with the same rigor, including a pre-gating prefetcher that a simulation predicted to be the largest lever and that measured −10 % to −27 % (Section 6).

2 Setup and method

2.1 Hardware

One workstation ("Titan"): AMD Ryzen 9 9950X3D (16 cores, 32 threads), 120 GB DDR5 RAM, NVIDIA RTX PRO 4500 Blackwell (32 GB) as the only GPU used for inference; a second GPU (RTX PRO 2000, 16 GB) took the early KLD measurements. Linux kernel 7.0, NVIDIA driver 610.43. The CPU experts run on cores 0–15 (one thread per physical core). In the final configuration the expert weights live in a pinned host buffer (`--load-mode none`) instead of a memory-mapped file in the page cache (Section 4.6).

2.2 Model

Qwen3.8-Flash-Next [Qwen Team, arXiv:2608.30320] is a sparse MoE model with 125 B parameters, 6 B of them active per token, plus 51 B parameters of n-gram embedding tables that Qwen keeps off the accelerator. Its 48 layers repeat three Gated DeltaNet (GDN) layers followed by one Qwen Sparse Attention (QSA) layer, and every layer has a MoE block with 512 routed experts, 10 active per token, and one shared expert. The residual stream is widened to four branches and read and written through Qwen's Gated Residual, a member of the hyper-connection family [Zhu et al., arXiv:2409.19606]; the configuration and our kernels call it hyper-connections (`hc_count` 4). A one-layer MTP module ships with the weights. The n-gram embedding (Qwen's term; `ple_*` keys in the Hugging Face configuration and tensor `per_layer_token_embd` in the GGUF, hence "PLE table" below) is a single hashed bigram/trigram lookup at the second decoder layer, stored as 16 sub-tables of about 20 M rows with 160 values each, 51.2 B weights in total.

2.3 Measuring quality

Quality is the mean KL divergence (KLD) of the quantized model's next-token distribution against BF16, plus top-1 agreement, following the argument that equal accuracy can hide many flipped answers, so that a distance metric such as KL divergence should be reported as well [Dutta et al., arXiv:2407.09141].

Development texts (F, E, V). FLE-IQ3_M was selected on three internal texts: F (used for selecting layers), E (check), V (held out until the end). Acceptance criterion, fixed in advance: mean KLD on F and E at most two combined standard errors above the reference ($\approx +0.0078$), top-1 at most 0.5 percentage points worse; V must confirm. F and V contain internal material and are not published; they remain an internal cross-check.

Public texts (E, W, K) for the comparison. All numbers in [Section 7](#) use three texts anyone can rebuild with `scripts/fle/bench/b6_texte.py` (sources and SHA-256 in `quellen.json`): E, the Python language reference (py-doc topics, PSF license, 63 KB); W, four German Wikipedia articles at fixed revisions (quantization in signal processing, floating-point number, main memory, graphics processor; CC BY-SA 4.0, 81 KB); K, seven files of the German Kubernetes documentation (`content/de/docs/concepts` at commit `4472af456106`; CC BY 4.0, 47 KB).

Neutral meter. KLD for the comparison is computed with upstream `llama-perplexity` at commit `a8c9a4e7c` without any FLE patch, so that no quantization is measured with our own kernels: `-c 2048 -b 2048 -ub 2048 -t 16 -ngl 99 -cmoe -fit off` with `LLAMA_ATTN_ROT_DISABLE=1`, GPU0, cores 0–15, inference server stopped. The BF16 reference (354 GB GGUF, converted from the release) is computed with the same build and settings; one pass over 2048 tokens takes 95 s. As a check of the meter, FLE-IQ3_M on the internal texts gives F 0.1505 / E 0.0516 / V 0.0929 with the neutral build against 0.1475 / 0.0511 / 0.0905 with our own build. This check ran earlier, on GPU1 with 8 threads, and reused the BF16 reference logits computed with our own build, so it tests the quantized side only.

Noise floor. The same file, text and build reproduce bit-exactly, but this model amplifies any change in computation order because expert selection flips: two runs of FLE-IQ3_M on GPU0 and GPU1 (16 and 8 threads, builds a few days apart) give 0.0905 and 0.0975 on text V; switching all CUDA fusions off (`GGML_CUDA_DISABLE_FUSION=1`) changes V by -0.0025 ; and the logits at micro-batch 2048 measured against those at micro-batch 4096 (context 4096) differ by a KLD of 0.062 (top-1 agreement 92.7 %). We therefore compare KLD only on the same card, build and settings, and treat differences below ~ 0.01 on a single text as no finding.

Knowledge probe (MKQA). KLD on three short texts can miss knowledge that a change removes, which matters for expert pruning. The pruned files are therefore also asked 400 factual questions from MKQA [Longpre et al., arXiv:2007.15207] (open-domain export `nthakur/mkqa-open-domain`, answerable questions only), the same questions in German and in English. The system prompt asks for the answer only, or exactly "I don't know"; sampling as on Qwen's card (temperature 1.0, top-p 0.95, top-k 20, min-p 0), thinking off, up to 256 tokens, one seed per question shared by all files (paired). An answer is correct when one of MKQA's German or English answer aliases appears in it after lower-casing and removing accents and punctuation; abstentions count as not correct. Differences are paired bootstrap intervals (2,000 draws). Server: the production FLE build (patch 51), experts on the CPU, no MTP

2.4 Measuring speed: paired brackets

Decode speed on this machine drifts by several percent over a day. Every speed claim in this report comes from a **bracket**: reference run, candidate runs, reference run again, all on the same prompts and server settings; the bracket is valid only if the two reference runs differ by at most 5 %. Candidates are compared with the reference per request (paired) and summarized with a bootstrap 95 % interval (`m2_gepaart.py`). Numbers from different brackets are never compared directly. The standard prompt set has 24 requests (code 9, math 6, physics 3, prose 6). Where a result comes from alternating arms (B A B A) without this pairing, it is marked as such.

3 Quantization: the FLE-IQ3_M hybrid

Start from Unsloth UD-IQ4_XS (93.68 GB file, 64.87 GB resident without the PLE table). Replace the expert tensors of 34 of the 48 MoE layers with the corresponding tensors from ISTA-DASLab's GSQ-RCO GGUFs (IQ3_S / IQ3_XXS) [GSQ arXiv:2604.18556, RCO arXiv:2605.00649]. Result: 88.87 GB file, 60.06 GB resident. KLD against BF16, relative to UD-IQ4_XS, as used for the acceptance decision: F -0.0079 , E $+0.0031$, V $+0.0005$, top-1 $+0.11$ / $+0.08$ /

+0.81 pp, so FLE-IQ3_M passes. These values were measured in decode mode, the way the files run in production (our build, GPU0, micro-batch 4; FLE-IQ3_M with 173 cache slots, the cache bias and the cache policy of [Section 4.1](#), UD-IQ4_XS with 160 slots and no bias). The files alone pass as well: $-0.0077 / +0.0029 / -0.0026$ with our build at micro-batch 2048, and $-0.0035 / +0.0038 / +0.0011$ with the neutral meter ([Section 2.3](#)).

What did not pass: our own full quantizations (KLD on F 0.259–0.269 against 0.155 for Unsloth), Q5_0 dense weights (F +0.024 against Unsloth, $z = +5.8$, top-1 -1.63 pp), fewer active experts ($k = 8$: +0.054 and +0.084 on F, depending on how the router weights are renormalized), a cache bias of 0.002 together with the cache policy (top-1 on F -0.54 pp), and the larger hybrids G4 (35 donor layers, 0.8 GB smaller than FLE-IQ3_M; top-1 on F -0.51 pp, just over the limit) and G6 (40 donor layers, 2.2 GB smaller; KLD on F and E more than two standard errors above the reference, top-1 on F -0.93 pp).

Positioning: FLE-IQ3_M is smaller than UD-IQ4_XS, not smaller than pure ISTA quants; the claim is "IQ4_XS quality at less size", and ISTA is the donor of most expert layers.

Layer selection. For every candidate donor and every MoE layer we predict the damage of swapping that layer's expert tensors as the relative output error $\text{err} = |y_Q - y_{\text{BF16}}|^2 / |y_{\text{BF16}}|^2$, measured on captured layer inputs (`quant_sensitivity.py`, 4096 tokens). The file is assembled from whole expert layers (gate, up and down together) with `gguf_splice.py`; everything else comes from the base file.

Smaller stages (FLE-IQ3_XXS, FLE-IQ2_M). To give a curve of size against KLD against speed instead of a single point, we build more hybrids with the same tools (two of three planned classes are buildable, see below). The selection is a knapsack: for a fixed target size it picks per layer the donor with the smallest predicted error (dynamic programming over 16 MB steps, `tools/fle-hybrid/fle_select.py` in the engine repository, with `gguf_splice.py` and `gguf_provenance.py`). FLE-IQ3_M itself was chosen the other way round, as the largest saving under an error bound. Recomputing it with the knapsack at its size of 88.87 GB keeps 44 of the 48 layers on the same source: one base layer moves to ISTA IQ3_S and three ISTA IQ3_S layers move to ISTA IQ3_XXS, which gives 35 instead of 34 donor layers and 88.51 GB.

Stage	Target / built size	Base (layers kept)	Donor layers chosen
FLE-IQ2_M	72 / 71.62 GB	ISTA IQ2_XS (24)	ISTA IQ3_XXS (18), Unsloth UD-Q2_K_XL (6)
FLE-IQ3_XXS	80 / 79.48 GB	ISTA IQ3_XXS (18)	ISTA IQ3_S (20), Unsloth UD-IQ3_XXS (10); UD-IQ4_XS offered, not chosen
FLE-IQ3_M (Q4 class)	— / 88.87 GB	Unsloth UD-IQ4_XS (14)	ISTA IQ3_S (28), ISTA IQ3_XXS (6)

The per-layer lists are in the model card and in `FLE-Q2.auswahl.json` / `FLE-Q3.auswahl.json` (the selection files keep the build names Q2 and Q3 of FLE-IQ2_M and FLE-IQ3_XXS).

Names and bit allocation. The files are named after the expert bit width, as Unsloth names its "UD" mixtures; `FLE-` marks the mixture and is not a standard llama.cpp file type (`fctype`). During development the three files were called G3 (the third hybrid candidate), Q3 and Q2, which sounded like the K-quant types Q3_K and Q2_K; the Q4 class of the table above is FLE-IQ3_M. The file average is higher than the expert width because the PLE table (`per_layer_token_embd`, 28.8 GB in FLE-IQ3_M, 29 % of the parameters, about 4.5 bits per weight in every file) is stored at fixed precision. Bits per weight, measured from the GGUF headers on 2026-10-09 (176.94 B parameters including the 51.2 B PLE table, 125.7 B without; Qwen reports 125 B + 51 B):

File	Total bpw	Experts	PLE table	Attention / shared expert / norms	Embeddings / output
FLE-IQ3_M	4.02	3.62	4.50	9.05	7.54
FLE-IQ3_XXS	3.59	3.08	4.51	7.32	4.53
FLE-IQ2_M	3.24	2.59	4.51	6.61	4.31
FLE-IQ3_M-REAP448	4.05	3.62	4.50	9.00	7.54

FLE-IQ3_M-REAP448 (161.84 B parameters, 64 of 512 experts per layer pruned) is FLE-IQ3_M with REAP-pruned experts.

Expert pruning (FLE-IQ3_M-REAP448). REAP [Lasby et al., arXiv:2510.13999] ranks the experts of each layer by their mean router-weighted output norm over the tokens routed to them. We compute it with `llama-fle-reap` over a calibration text of $256 \times 2,048$ tokens (FLE-IQ3_M's own answers to public prompts: 45 % German conversation, 25 % code and tool calls, 18 % English conversation, 12 % math; the KLD texts and the probe questions are not in it) and remove the lowest-ranked experts per layer by byte copy of the kept expert slices and router rows (`reap_prune.py`, no re-quantization). At 384 of 512 experts we compared REAP with the gate-free mean activation norm MAN [Liu et al., arXiv:2606.15716]; the winner on German MKQA was then built with 448 experts. KLD with the neutral meter (Section 2.3), MKQA as described there, all measured 2026-10-08:

File	Size (GB)	KLD E / W / K	MKQA correct DE / EN	vs FLE-IQ3_M, DE / EN (points, paired 95 %)
FLE-IQ3_M	88.87	0.051 / 0.050 / 0.129	23.2 / 33.0 %	—
FLE-IQ3_XXS	79.48	0.090 / 0.093 / 0.227	21.5 / 30.8 %	-1.8 [-4.2 .. +0.5] / -2.2 [-5.0 .. +0.2]
REAP, 448 experts (FLE-IQ3_M-REAP448)	82.00	0.080 / 0.086 / 0.185	21.5 / 29.8 %	-1.8 [-4.5 .. +0.8] / -3.2 [-6.5 .. +0.0]
REAP, 384 experts	75.13	0.135 / 0.132 / 0.261	17.5 / 21.2 %	-5.8 [-9.2 .. -2.5] / -11.8 [-16.0 .. -7.8]
MAN, 384 experts	75.13	0.167 / 0.165 / 0.325	11.2 / 15.5 %	-12.0 [-15.8 .. -8.5] / -17.5 [-21.8 .. -13.2]

The 384-expert REAP file is the instructive row: its KLD is only somewhat above FLE-IQ3_XXS, but its loss of correct answers is three to five times as large. A few short texts do not exercise the facts that rarely routed experts carry. At 448 experts the loss is about that of FLE-IQ3_XXS. REAP beat MAN clearly at equal size, although Liu et al. recommend MAN for task-agnostic pruning; this is one model and one probe. On the calibration text the 448 kept experts receive 96.1 % of the summed router weight on average (lowest layer 90.5 %). All five files repeated an invented rare full name correctly in 60 of 60 short messages per language. The absolute MKQA rates are low because the questions come from Natural Questions (many US-specific) and the model abstains often (FLE-IQ3_M: 91 of 400 German, 21 of 400 English answers). Speed: at the same cache budget FLE-IQ3_M-REAP448 decodes as fast as FLE-IQ3_M (+0.71 % [-0.32 .. +1.75], bracket B3e6, [Section 7](#)). With a warm profile its hit rate in the first 512 steps is the same (93.8 % against 93.7 %), as expected if the removed experts are the ones that rarely reach the cache; the smaller file saves 6.9 GB of RAM, not time.

ISTA Q2_0 is not used as a base because llama.cpp has no fast CPU kernel for it; it appears only as a comparison.

No 1-bit class from public files. A planned FLE-Q1 at 63 GB cannot be built from the public releases. Every low-bit file carries the same non-expert part of about 32.6 GB, of which 28.8 GB is the PLE table (`per_layer_token_embd`); and Unsloth's UD-IQ1_M keeps many layers at higher precision, so its experts (41.9 GB) are larger than those of ISTA IQ2_XS (35.5 GB):

File	Experts	Rest	of which <code>per_layer_token_embd</code>	Total
ISTA IQ2_XS	35.5 GB	32.6 GB	28.8 GB	68.0 GB
Unsloth UD-IQ1_M	41.9 GB	32.7 GB	28.8 GB	74.5 GB
ISTA Q2_0	34.0 GB	32.4 GB	28.8 GB	66.4 GB

The smallest hybrid these sources allow is about 66.6 GB, barely below IQ2_XS and with the worse Q2_0 experts. A lower-precision `per_layer_token_embd` does not help within the GGUF formats either: the table has rows of only 160 values (shape [160, 320 001 536], stored as IQ4_NL in every release), so only types with blocks of 32 fit. K-quants, IQ2/IQ3 (blocks of 256) and the new Q2_0/Q1_0 (blocks of 64/128) cannot store it; the smallest fitting type, MXFP4 at 4.25 bits, would save 1.6 GB. Going lower would need a custom type, and the file would no longer load in upstream llama.cpp. The ≈ 33 GB of non-expert weights are therefore a floor set by the format. The predictor gives a summed error of 3.19 for FLE-IQ3_XXS against 4.95 for its base ISTA IQ3_XXS alone, and 5.90 for FLE-IQ2_M against 7.36 for ISTA IQ2_XS (predictions). Measured ([Section 7](#)): FLE-IQ3_XXS matches Unsloth UD-IQ3_XXS (KLD 0.090 / 0.093 / 0.227 against 0.091 / 0.091 / 0.229) at 2.5 GB less, and FLE-IQ2_M is clearly better

than the larger Unsloth UD-IQ1_M (0.160 / 0.163 / 0.339 against 0.189 / 0.186 / 0.413). Against the ISTA files both stages lie about on the line between IQ2_XS, IQ3_XXS and IQ3_S: they add size steps, not a better size–quality curve. Speed of the stages at the same cache budget (Section 7, bracket B3e5): FLE-IQ3_XXS -0.69% [$-2.07 .. +0.57$] and FLE-IQ2_M $+11.60\%$ [$+9.78 .. +13.39$] against FLE-IQ3_M.

4 Engine

4.1 Expert cache on the GPU

Every MoE layer whose experts stay in host memory (46 layers for FLE-IQ3_M; the memory fit decides the number) gets a cache of expert slots on the GPU, 173 per layer before the byte budget below. Admission is ranked: the misses of a step are sorted by an aged request count (counters halved every 2,000 observations per layer, `FLE_MOE_ADMIT_FREQ=2000`), and at most one expert per layer and step enters the cache. Eviction takes the least frequent of the 32 oldest entries (`FLE_MOE_EVICT_OLDEST=32`). A small router bias towards cached experts (0.001 in probability units) changes which experts are chosen, not their weights, and keeps the outputs within the KLD criterion (Section 3). Bracket HZ2 (reference Unsloth UD-IQ4_XS; all arms with 160 slots per layer and n-max 1; reference drift $+0.38\%$): FLE-IQ3_M alone $+4.05\%$ [$+2.92 .. +5.19$], FLE-IQ3_M with bias and policy **$+12.15\%$** [$+10.68 .. +13.57$].

The smaller FLE-IQ3_M file frees GPU memory, but only turning that memory into cache slots makes it faster. Bracket V6b (Section 4.4) measures the slope: 146 instead of 173 slots cost -4.39% , 120 slots -10.04% . An earlier bracket that compared slot counts against UD-IQ4_XS (HY) is not cited: its three candidate arms ran under memory pressure (cgroup memory-high events in all three, none in the reference arms), with half the output limit and 19 instead of 24 paired requests (Appendix A).

The final build replaces the fixed slot count by a **global byte budget** (patch 44): the cache gets a number of bytes, the slots are distributed over all layers with host experts by a usage profile (118 to 248 slots per layer for FLE-IQ3_M in the final brackets B3e5 and B3e6, mean 174), and the profile is saved every 600 s and loaded at the next start (warm start). The effect is measured in Section 5.3 (bracket CG): $+1.49\%$ [$+0.64 .. +2.41$] with the profile, and a much smaller loss when memory gets tight.

4.2 Overlap and fusion

GPU cache hits run while the CPU computes the misses (`FLE_MC_OVERLAP=2`), and the Gated Residual runs in fused CUDA kernels (`FLE_HC_FUSED=1`). Bracket ALL2 (reference FLE-IQ3_M at n-max 2 without both, 14.6 ms per token; drift -0.44% ; 24 paired requests):

Candidate	Speed vs reference	95 % CI	Acceptance
combined build + overlap	$+15.93\%$	$+14.60 .. +17.42$	76.8 %
+ IQ4_NL MTP head	$+19.46\%$	$+18.00 .. +21.03$	76.5 %
+ Q4 MTP head	$+16.63\%$	$+15.25 .. +18.14$	78.2 %
+ IQ4_NL head, Q6_K dense weights	$+19.14\%$	$+15.76 .. +22.14$	75.5 %

The Q6_K variant passes the quality criterion but is not faster than the IQ4_NL head alone; it ran with 45 cache layers of 179 slots, the other arms with 46 of 173.

4.3 Speculative decoding with the MTP head, and a re-training that did not generalize

Draft-and-verify [Leviathan et al., arXiv:2211.17192; Chen et al., arXiv:2302.01318] with the model's own MTP module [DeepSeek-AI, arXiv:2412.19437; Gloeckle et al., arXiv:2404.19737]. With the shipped head, the conditional acceptance derived from the tokens per cycle (Section 4.4) is about 0.86 / 0.78 / 0.78 for the first, second and third drafted token, and two drafted tokens (n-max 2) give the shortest time per token in that measurement. The bracket meant to confirm the choice directly (NM2) drifted by -5.88% and is invalid (Appendix A).

The released head is the untrained module: Unsloth's `MTP/mtp-Qwen3.8-Flash-Next-shared-Q8_0.gguf` (revision `eb2a07ecb33b`, 2026-09-01) with only the three expert tensors `blk.48.ffn_down_exps` / `ffn_gate_exps` / `ffn_up_exps` requantized from Q8_0 to IQ4_NL (relative error against the source on the first 8,192 rows 0.0772 / 0.0785 / 0.0781); the other 29 of 32 tensors are unchanged (1,528,277,056 bytes). This is the IQ4_NL head of [Section 4.2](#). A 32k draft vocabulary adds +4.14 % [+3.04 .. +5.24] (bracket VK, measured with head c2 below).

Re-training (method). We re-trained the head on the model's own answers (recursive self-distillation: the head is rolled out to depth 7 against the main model's top-32 logits). Recipe of head c2 (run `lauf3`): teacher and answer generator are the FLE-IQ3_M file; cross-entropy against the renormalized top-32 teacher logits at each of 7 rollout depths plus an L2 anchor (3e-3) to the shipped head; only the 25 non-expert tensors (88.9 M parameters) are trained; 6,000 steps, sequence 1,024, AdamW, learning rate 3e-5 with cosine decay, bf16, 3.5 hours on one GPU. About 40 % of the prompt pool are the author's own assistant conversations (filtered for secrets, not published), the rest public datasets (OASST, SWE-Gym, SWE-Zero, UltraChat, GSM8K). The training corpus (m2) also contained FLE-IQ3_M's answers to exactly the 8 fixed tasks used for the KTC measurement.

Re-training (result): a negative result.

- *On the training tasks* (bracket KTC, 24 paired requests on the same 8 tasks): c2 +4.54 % [+2.67 .. +6.75] against the untrained head, acceptance 80.2 % vs 77.1 % (an earlier +4.92 % could not be reproduced from the raw data, head c3 gives +4.90 %). A larger run (`lauf5`: 1.7 times the sections plus a fourth capture set, 9,807 steps, trained from the shipped head) did not beat c2: +0.28 % [−0.73 .. +1.48] at n-max 2 (bracket KTE, 48 paired requests; acceptance 79.9 % against 80.4 %). In hindsight the KTC gain is overfitting to the measurement tasks.
- *On unseen tasks, first try* (bracket KTF: 16 tasks not in any corpus, 8 German and 8 English, code, math, prose, knowledge; 48 paired requests, reference c2 before/after): untrained head +1.61 % [+0.88 .. +2.36] against c2, head pub (re-trained on public prompts only) −0.11 % [−1.11 .. +0.86]; acceptance c2 71.35 %, untrained 72.40 %, pub 71.73 %. Confounder: each arm had its own warm-start expert profile (c2's predicted 72 % hits, the others 80 %).
- *On unseen tasks, clean repeat* (bracket KTG, 2026-10-09, same 48 requests, every arm starts from a copy of the same production profile with 79.1 % predicted hits; FLE-IQ3_M, n-max 2, bootstrap 95 %; reference c2 11.740 → 11.298 ms per token, drift −2.38 %): untrained head +2.95 % [+1.96 .. +3.93], pub −1.63 % [−2.35 .. −0.91]; acceptance untrained 71.00 %, pub 71.10 %, c2 70.91 %.

KTG, speed vs c2 (n = 6 each)	de code	de math	de prose	de knowledge	en code	en math	en prose	en knowledge
untrained (released)	+2.7 %	+2.1 %	+1.6 %	−0.5 %	+3.9 %	+5.1 %	+5.1 %	+3.7 %
pub	−1.5 %	−2.0 %	−1.8 %	−3.8 %	−2.7 %	−2.5 %	+1.6 %	−0.3 %

Acceptance is practically the same for all three heads, so the untrained head's lead is not explained by acceptance; its cause is not known. The gain of re-training held only on the tasks in its corpus. The release therefore ships the untrained head; c2 and pub are not published. Lesson: **measurement tasks and training corpus must be disjoint**. The final series (B1e, B3e, S), the benchmarks of [Section 5.4](#) and bracket VK were measured with c2 and are reported as measured; we do not extrapolate them to the released head.

4.4 Where the time goes

Cycle model $T(n) = T_0 + c \cdot M(n+1) + \sum DG_j$ (M = cache misses, DG = extra cost of the j -th drafted token), calibrated on single requests of the base build of 2026-09-30 (bracket NM: untrained head, fused Gated Residual and overlap not yet switched on): $T_0 = 22.6$ ms, about 82 % of a single-token cycle of 27.4 ms. Measured cycles for n-max 1/2/3: 27.4 / 34.6 / 42.8 ms for 1.86 / 2.53 / 3.06 tokens (n-max 3: 18 valid requests). Three parameters on three points describe these measurements but do not test the model. An Nsight Systems profile of the production build of 2026-10-04 (before the final configuration; 15 s window, n-max 2, 77.3 tokens/s) shows the GPU busy 64.5 % of the time; the gaps split into launch 17.3 %, CPU after the GPU 7.6 %, CPU before 6.4 %, wake-up 2.0 % and GPU-internal 0.9 %. A gap between consecutive GPU operations averages 27 μ s; a full GPU–CPU–GPU round trip averages 226 μ s (median 55 μ s).

Sensitivity to cache misses, measured directly (bracket V6b, 2026-10-06; same build and 46 cache layers, only the slot count changes; reference drift +0.68 %): 173 slots give 30.3 misses per step (hit rate 91.2 %), 146 slots 38.5 (+27 %) and 120 slots 52.3 (+72 %); speed -4.39 % [-5.43 .. -3.37] and -10.04 % [-10.74 .. -9.37]. Each percent more misses costs 0.14–0.16 % speed, less than the cycle model predicts; our reading is that the overlap, switched on after the model was calibrated, hides part of the CPU time. The slope is measured only towards more misses.

4.5 Prefill and prompt cache

Lean prefill at micro-batch 1024 (patch 35), bracket P35Q against micro-batch 512 (drift -0.30 %): micro-batch 1024 alone costs decode speed, -19.01 % [-19.76 .. -18.26]; our reading (not measured separately) is that the larger compute buffer takes GPU memory from the expert cache. With the memory fit lowered from 19156 to 18806 MiB, decode is +2.26 % [+1.24 .. +3.29]. The same effect shows in END2: the 1024 candidate without fit lost -20.36 %. Prefill of an 8.6k prompt: about 425 instead of about 280 tokens/s (two cold runs per arm, 2026-10-03; the final configuration reaches about 1,250 tokens/s, [Section 4.6](#)). Prompt-cache protection for the interactive slot (patches 39/39b): a follow-up turn at 7.1k context took 2.4 s (516 tokens recomputed, the rest restored from the RAM cache); computing the same context from scratch took 17.5 s (single probe, 2026-10-03).

4.6 Final configuration: threads, pinned experts, slot release, byte budget, ubatch lending

Three steps after END2 lead to the intermediate configuration `p43-pin`; four more, taken over from Strata or found while measuring against it ([Section 5.3](#)), lead to the final configuration `p49-ring` that all final numbers use:

- **16 threads on cores 0–15** instead of 8 on cores 0–7: +5.23 % [+3.64 .. +6.86] (bracket RB2, paired). The CPU computes the cache misses (≈ 6 % of expert calls at 94 % hits), so more cores shorten the critical path.
- **Slot release (patch 43)**: when a conversation switches slots, the time to first token falls by 202.5 ms (-25.6 %); outputs stay within GPU noise (alternating arms, not a paired bracket).
- **Pinned experts** (`--load-mode none`, no patch): expert weights are read once into a pinned host buffer, so every host-to-device copy runs at pinned-memory speed. Against a warm production run (model fully in the page cache before every reference arm; alternating arms B A B A, medians, not a paired bracket): prefill of 1000 words 446 → 852 tokens/s (+91 %), 3000 words 558 → 1037 tokens/s (+86 %); time to first token -309 ms for prompts of about 50 tokens and -613 ms for about 490; decode 77.4 → 84.5 tokens/s (+9.1 %). Cost: 54 GB of RAM are pinned and can no longer be reclaimed by the page cache (available memory after loading 47.5 instead of 100.5 GiB); loading takes 55 instead of 45 s. Greedy outputs of the longest task differed between all arms, also between arms of the same build (GPU noise). The decode gain is not decomposed and was not repeated in a paired bracket; our candidate explanation is that cache-miss copies also run from pinned memory.
- **Byte budget and warm start for the expert cache (patch 44)**: one budget of 17,035 MiB for all layers instead of a fixed slot count, planned from a saved usage profile; +1.49 % [+0.64 .. +2.41] (bracket CG, [Section 5.3](#)).
- **AVX2 dot product for Q2_0 (patch 45)**: matters only for files with Q2_0 tensors ([Section 6](#), [Section 7](#)).
- **Ubatch lending (patch 48)**: decode runs with a 1,024-token micro-batch and the full cache; a prefill call of 2,048 tokens or more lends whole expert-cache layers from the back for the larger compute buffer and gives them back afterwards (bracket W7: 15 layers and 5,033 MiB at micro-batch 4096, 2,327 slots refilled from pinned RAM in 106 ms; in the final configuration, with the prefetch ring below, 19 layers, 6,409 MiB and 2,952 slots in 136 ms). Bracket W7 (drift -0.26 %): prefill 1k / 4k / 8.6k ± 0 / +30.0 % / +53.7 %; decode -0.25 % [-1.15 .. +0.66].
- **Prefetch ring (patch 49)**: while lending, a second CUDA stream copies the whole weights of the next expert splits into a ring of three staging buffers (450 MiB each, also lent from the cache), so the copy overlaps the computation and the per-layer synchronization on the routed ids disappears. Greedy answers of about 400 characters are identical to the build without it for prompts of 1k, 4k and 8.6k tokens. Bracket W49c (drift +0.35 %): prefill 4k +10.6 %, 8.6k +14.2 % on top of patch 48; decode -0.18 % [-0.91 .. +0.49]. An nsys timeline shows why not more: each layer has three weight splits, up and gate compute shorter than their copy takes at 57.5 GB/s, so about 0.64 s of copy without computation remain per 8.6k prompt; six buffers instead of three did not help (1,220 against 1,249 tokens/s at 8.6k).

After the final measurement one correctness fix went into production (patch 50, a server change): a request that names a slot which is still busy was allowed to load its cached prompt into that slot before it was deferred, which replaced the running generation's state (upstream behaviour of `get_available_slot`; it corrupted our first benchmark runs, [Section 5.4](#)). The fix does not touch the computation; speed was not re-measured with it.

5 Results

5.1 Against upstream llama.cpp

Bracket B1e (2026-10-06, final measurement; 24 paired requests; reference drift +0.44 %). Reference: FLE in the final configuration (`p49-ring`: FLE-IQ3_M, expert cache with a 17,035 MiB byte budget and warm-start profile, overlap, re-trained MTP head c2 at n-max 2, 32k draft vocabulary, 16 threads, pinned experts, ubatch lending with prefetch ring). Measured with the internal head c2; the published untrained head was +2.95 % [+1.96 .. +3.93] faster than c2 on unseen prompts (bracket KTG, [Section 4.3](#)). Candidates: upstream llama.cpp at commit `a8c9a4e7c` without any patch, a current upstream state that is newer than the FLE base `6a1a922d` (2026-09-05; FLE rebased onto `a8c9a4e7c` was 5.7 % slower, [Section 6](#)), with micro-batch 1024 and with micro-batch 4096; same FLE-IQ3_M file, same card, 16 threads, `--load-mode none`; without an expert cache upstream fills the GPU statically (fit target 1024 MiB: at micro-batch 1024 that is 18,195 MiB of experts and 5,362 MiB of dense weights on GPU0, at micro-batch 4096 only 9,458 MiB of experts).

	Speed vs FLE	95 % CI	FLE is
FLE <code>p49-ring</code> (median 9.9–10.1 ms per token, about 100 tokens/s)	—	—	—
upstream llama.cpp <code>a8c9a4e7c</code> , micro-batch 1024	-59.87 %	-61.24 .. -58.50	2.49× as fast
upstream llama.cpp <code>a8c9a4e7c</code> , micro-batch 4096	-64.01 %	-65.20 .. -62.86	2.78× as fast

The gap holds for every prompt type (micro-batch 1024: -59.0 % code, -63.6 % math, -58.6 % physics, -57.9 % prose). The FLE reference accepts 82–86 % of the drafted tokens.

Prefill is the other side. Per arm in the same bracket (medians, not paired), tokens/s:

Prompt	FLE <code>p49-ring</code>	upstream, micro-batch 1024	upstream, micro-batch 4096
~1k tokens	416–442	715	517
~4k tokens	928–934	900	1,222
~8.6k tokens	1,248–1,256	967	1,416

Time to first token for prompts of 20–192 tokens: FLE 0.71–1.03 s (one outlier of 1.41 s at 192 tokens in the first reference arm), upstream (micro-batch 1024) 0.64–0.92 s. FLE is 24 % (4k) and 12 % (8.6k) behind upstream with micro-batch 4096 and about 40 % behind upstream with micro-batch 1024 at 1k. Our reading, not measured: upstream keeps part of the experts statically on the GPU (9,458 MiB, 18 % of them, at micro-batch 4096) and needs no copies for those, while FLE copies nearly all experts through PCIe in every large micro-batch ([Section 4.6](#)); below 2,048 tokens FLE does not lend and takes the old path that copies all experts of a layer with the small compute buffer.

The mainline arm runs without speculative decoding because it cannot use the public MTP files: upstream's MTP support for this model (#29761) takes the MTP block from the target GGUF or from an MTP-only GGUF that carries its own token embedding and output head, while Unsloth's "shared" head, the basis of the FLE head, has neither. That head fails to load (`token_embd.weight not found`); Unsloth's complete head loads but aborts at the first compute step (`GGML_ASSERT(buffer)` in `process_ubatch`), on the 16 GB card as well as on the 32 GB card with the head on the GPU, so it is not a memory problem (`scripts/fle/bench/b1_mtp_probe.sh`). For a user of the public files, the arm without MTP is the realistic default.

5.2 Against the previous production

End-to-end bracket END2 (24 paired requests; code 9, math 6, physics 3, prose 6; reference drift +1.44 %):

	ms per token (median)	Speed vs previous production	95 % CI	Acceptance
previous production (stock build, Unsloth UD-IQ4_XS)	17.9	—	—	76.5 %
FLE (FLE-IQ3_M, cache policy, overlap, IQ4_NL MTP head, n-max 2)	11.3	+58.53 %	+55.83 .. +61.58	77.0 %
same, micro-batch 1024 without memory fit	—	−20.36 %	−22.32 .. −18.36	79.1 %

Per category FLE gains +57.4 % (code), +58.1 % (math), +54.7 % (physics) and +62.6 % (prose).

[Appendix C](#) lists every paired bracket this paper cites (date, reference, candidate, effect, 95 % CI, reference drift, n), generated from the paired evaluations by `scripts/fle/release/klammer_tabelle.py`.

5.3 Against Strata

Strata 0.1.39 (Niko1221, MIT) is the closest public engine: it also keeps a GPU expert cache, computes misses on the CPU and drafts with the model's MTP head. Both engines ran FLE-IQ3_M on the same machine with the same tasks; FLE before and after the Strata arms. Strata ran with its own defaults (131,072 context, fp16 KV in RAM, one slot, 20.9 GiB of expert cache); FLE keeps 262,144 context, q8 KV and two slots on the GPU and therefore has only 16.6 GiB (17,035 MiB) of cache. The comparison was run twice: first against `p43-pin` (2026-10-06 night, `scripts/fle/fusion/tore_kette.sh`), which started the takeover below, and again in the final measurement against `p49-ring` (comparison S, 2026-10-06 17:01–17:13, Strata `--spec 4` on 16 cores, `scripts/fle/bench/end_kette.sh`). FLE drafted with the internal head c2 in the final comparison ([Section 4.3](#)).

	FLE <code>p43-pin</code>	FLE <code>p49-ring</code> (final)	Strata 0.1.39
decode, tokens/s (mean of the four per-task medians)	90.8 / 90.4 (before / after)	91.2 / 91.3 (before / after)	75.8 then, 74.0 in S (<code>--spec 4</code>)
prefill 1,000 / 4,000 / 8,600 tokens, tokens/s	447 / 653 / 716	443 / 922–933 / 1,179–1,181	1,190 / 2,218 / 2,740 then, 1,183 / 2,239 / 2,673 in S
time to first token, prompts of 20–192 tokens	0.69–1.03 s	0.69–1.04 s	0.39–0.50 s

In the final comparison FLE decodes about 23 % faster; Strata prefills 2.7 (1k), 2.4 (4k) and 2.3 (8.6k) times as fast (before the takeover 2.7, 3.4 and 3.8 times) and answers short prompts 0.3–0.5 s earlier. The decode figures are unpaired means of the per-task medians, not a paired bracket. We took four Strata ideas as candidates and gated each by a measurement before building it:

- **Global byte budget for the expert cache with a warm-start profile** (patch 44; Strata keeps a usage profile and fills the cache by rank). Bracket CG (24 paired requests, drift +0.28 %): budget alone +0.01 % [−0.49 .. +0.48], with the saved profile **+1.49 %** [+0.64 .. +2.41]; hit rate in the first 512 steps 93.7 % instead of 67.6 % cold. With 2.6 GB less budget (the memory a rebased build lost in [Section 6](#)) speed falls by only −3.26 % [−4.19 .. −2.32] instead of −47 %, because every layer shrinks a little instead of whole layers losing their cache. Adopted.
- **Doorbell window graph** (Strata enqueues a whole verification window and lets the host signal each layer). Gate: Strata's fixed time per window must be clearly below ours. A regression of Strata's own window timings over the drafted tokens T gives $11.9 + 7.6 \cdot T$ ms, intercept 95 % 8.7–15.4 ms (bootstrap over 24 segments), against our fixed share of about 15.2 ms; the largest fixed item in Strata is itself a wait for the GPU. The doorbell hand-off costs 1.8 μ s (P50, P99 2.0 μ s; microbenchmark on the 16 GB card), so it is not the obstacle. Gate missed, not built ([Section 6](#)).
- **PCIe share of the misses** (Strata copies 55 % of the missed experts to the GPU instead of computing them on the CPU). Gate: the speed must depend on the miss count. Bracket V6b with 173/146/120 slots: misses per step 30.3 / 38.5 / 52.3, speed −4.39 % [−5.43 .. −3.37] and −10.04 % [−10.74 .. −9.37], i.e. 0.14 to 0.16 % speed per percent of misses. Gate passed; built as patch 47 (a ring of three slots per layer into which part of the missed experts is copied ahead of the GPU chain). It worked as designed, 31.7 % and 40.6 % of the non-cached experts were served through the ring, but decode lost: −4.46 % [−5.73 .. −3.32] and −4.47 % [−5.53 .. −3.46]

(bracket PC, drift +2.37 %). The copies sit on the stream of the hit chain, so every layer waits for about 2 MB of weights, which costs more than the CPU saves. Our written forecast before the measurement was +3 .. +10 %, and +7 .. +9 % after bracket V6b. Not adopted ([Section 6](#)).

- **Larger micro-batches for long prompts** (Strata borrows cache memory for prefill). Gate: the prefill gap above. Built as patches 48 and 49 ([Section 4.6](#)): prefill 4k and 8.6k +30.0 % and +53.7 % (bracket W7), then +10.6 % and +14.2 % on top (bracket W49c), decode unchanged within the interval. Adopted.

The doorbell window graph is the one idea that stays open ([Section 6](#)).

5.4 Standard benchmarks

To check that FLE-IQ3_M on FLE keeps the model's task quality, we run three benchmarks from Qwen's model card with the sampling of that card (temperature 1.0, top-p 0.95, top-k 20, min-p 0, no presence penalty, up to 81,920 new tokens, thinking on), against the production server (profile `p50-fix`): IFBench (prompt-level, card 81.3), GPQA Diamond (card 91.7) and LiveCodeBench v6 (card 91.9). One run per benchmark.

Benchmark	Tasks	FLE-IQ3_M on FLE, % [Wilson 95 %]	Qwen card, %	Measured
IFBench, prompt-level strict	300	80.3 [75.5 – 84.4]	81.3 (strict or loose not stated)	2026-10-07 12:25 – 19:49
IFBench, prompt-level loose	300	83.0 [78.3 – 86.8]	81.3 (strict or loose not stated)	same run
GPQA Diamond	198	88.9 [83.8 – 92.5]	91.7	2026-10-07 19:49 – 2026-10-08 06:44
LiveCodeBench v6, 2025-02-01 to 2025-05-31	131	85.5 [78.5 – 90.5]	91.9	2026-10-08 06:46 – 21:02

Source of the card values: the model card of Qwen3.8-Flash-Next (<https://huggingface.co/Qwen/Qwen3.8-Flash-Next>, revision `de4b8e4d43b917e7706784d8bb445c9af86a3540`, column "Qwen3.8-Flash-Next" of the language benchmark table, read 2026-10-08). The card gives one number per benchmark and no number of runs; it states no output budget and no LiveCodeBench window. The intervals are Wilson 95 % intervals for a binomial proportion with the number of tasks as n; they describe the sampling of tasks only, not the run-to-run spread of one task at temperature 1.0, which we did not measure.

Method. All three ran through the production server (`llama-server`, port 18420, FLE-IQ3_M file, the internal re-trained MTP head c2, not the released untrained head ([Section 4.3](#)), two drafted tokens, 262,144 tokens of context shared by two slots, i.e. 131,072 per slot) with the build that contains patch 50 (profile `p50-fix`, in production since 2026-10-07 12:23). The harness (`scripts/fle/bench/rhea_gen.py`) sends each task once as a chat request with the card's thinking-mode sampling and a limit of 81,920 new tokens (an earlier Qwen recommendation for competition problems; the 262,144 reasoning tokens that the card's best-practice section recommends do not fit into a 131,072-token slot). Two requests ran in parallel; for LiveCodeBench, one from 2026-10-08 12:00 on. The harness ran on cores 16–31, the server on cores 0–15. Scoring: IFBench with the official evaluator of allenai/IFBench (strict and loose, prompt- and instruction-level); GPQA Diamond with our own runner in Qwen's answer format `"answer": "C"` and the answer order of simple-evals; LiveCodeBench with the official `lcb_runner` (`release_v6`, prompt `format_prompt_generation`, code extraction `extract_code`, `codegen_metrics`, `pass@1` from one sample). The LiveCodeBench window 2025-02-01 to 2025-05-31 is taken from Qwen's earlier model cards (Qwen3-2507), not from this card. Each stage began with a one-task smoke test. No answer was re-drawn or filtered. Requests that failed at transport level would be repeated by a second pass; that pass found nothing open in any of the three runs.

Invalid and truncated answers. An answer that reaches the token limit before its final answer counts as wrong and stays in the denominator. IFBench: none empty, none truncated. GPQA Diamond: six truncated answers.

LiveCodeBench: ten truncated answers (details below).

IFBench (300 tasks, one run, on the fixed server): all 300 answered, none empty, none at the length limit; output tokens median 4,427, mean 8,424, maximum 75,605. Instruction-level: strict 82.8 %, loose 85.2 %. Qwen's card does not say whether 81.3 is strict or loose; with a standard error of about 2.3 points for 300 tasks, both of our values are

level with it.

GPQA Diamond (198 questions, one run, Qwen's answer format `"answer": "C"`): 176 of 198 correct, 88.9 % (Wilson 95 % interval 83.8–92.5), so the card's 91.7 lies inside the interval; the gap is 2.8 points against a standard error of about 2.2. Six answers have no letter: all six reached the 81,920-token budget while still reasoning (all chemistry questions) and count as wrong. There were no empty answers and none from another conversation. If all six had been answered correctly the score would be 91.9 %. Output tokens: median 5,289, mean 15,191.

LiveCodeBench v6 (131 problems from 2025-02-01 to 2025-05-31, one run): 112 of 131 solved, pass@1 85.5 % (Wilson 95 % interval 78.5–90.5). By difficulty: hard 72.1 % (44 of 61), medium 94.9 % (37 of 39), easy 100 % (31 of 31); the counts follow from the reported percentages and the total. The card's 91.9 lies outside the interval; the gap is 6.4 points, about 2.1 standard errors (3.1). Of the 19 failed problems, 10 are truncations: the answer reached the 81,920-token budget while still reasoning (`finish_reason = length`), so the final message is empty, no code is extracted and the problem fails. The other 9 produced code that failed the tests. Two more problems finished with 80,241 and 81,272 tokens, just under the limit. Output tokens per problem: median 17,032, mean 26,839 (22,287 over the 121 that finished), 3.5 million in total. If all ten truncated problems had been solved the score would be 122 of 131, 93.1 %, above the card; so the budget could account for the whole gap, but we have not re-run with a larger budget and cannot say that it does. Unlike GPQA Diamond, where the card lies inside the interval, we therefore report LiveCodeBench as below the card.

Limits of these numbers. (1) One run per benchmark; the spread between runs at temperature 1.0 is unknown and the intervals above do not contain it. (2) The 81,920-token limit cut off 6 of 198 GPQA answers and 10 of 131 LiveCodeBench answers that were still reasoning; the card states no budget for its benchmark runs (its best-practice section recommends 262,144 reasoning tokens), so the comparison is not like for like. (3) We did not run the unquantized model, Qwen's own harness or any other quantization on these benchmarks, so a difference to the card mixes quantization, engine, our harness and chance, and we cannot separate them. The benchmarks check the production configuration; they do not measure quantization loss (that is [Section 3](#)). (4) LiveCodeBench has only 131 problems in the window, 61 of them hard; the interval is wide. (5) The card's LiveCodeBench number may come from a different window. (6) Everything ran on one machine in one engine state.

A first run was invalid, and why. The first IFBench and GPQA runs (2026-10-06/07) sent their requests directly to the server while the gateway in front of it assigned interactive requests to slots by `id_slot`. In upstream `get_available_slot`, a request that names a slot which is still processing gets that slot returned; because no cached prompt matches, the function then runs the prompt-cache update on the busy slot (`prompt_save` and `prompt_load`), which replaces the KV and recurrent state of the running generation with the waiting request's cached prompt, on every retry until the slot is free. The running answer then continued inside another conversation or stopped early. Long generations were hit hardest, because their own state exceeds the prompt-cache limit, is not saved, and therefore never evicts the waiting entry. A canary test (`scripts/file/bench/slot_kanarien.py`: a long answer on slot 0, then a request for slot 0 with a different cached conversation) flipped 3 of 3 rounds into the other conversation; with patch 50 (return the busy slot untouched, the caller defers the task) 0 of 3, the waiting request was deferred for 15–17 s and the long answer ran to its full length. Both invalid runs were discarded and all three benchmarks were re-run on the fixed server.

6 Negative results

Pre-gating prefetch (router top-K of layer l+1 on the input of layer l, asynchronous loads): a trace simulation gave +18 .. +25 % for a single stream (+24 .. +35 % with two parallel slots), assuming free room for the prefetch slots and leaving out the cost of the extra copies. Offline, the router inputs predicted 57–81 % of the misses one layer ahead (prefetch width 10 to 24), within the simulated range. Measured: −9.67 % [−11.14 .. −8.22] (bracket VL, reference drift −1.28 %), worse with more prefetch slots (−12.49 %, −27.34 %); the extra copies cost more than the hits save. Registering the memory-mapped model file in place with `cudaHostRegister` failed (0 of 3 ranges); the final configuration instead reads the experts into a separately allocated pinned buffer ([Section 4.6](#)).

Smaller items. N-gram drafting before MTP: −13.77 % [−18.86 .. −8.58] and −23.34 % [−30.02 .. −16.12] (bracket NG; acceptance falls from 78 % to 66–71 %). Low-rank kernels for the Gated Residual (patches 30/30b) were slower than the existing kernel chain in micro-benchmarks (49 against 28 μ s per mix) and are not enabled.

`GRAPH_OPT=1` changed the step cost by -1.7 to -0.4 %, inside the spread of the reference runs (up to 7 %). Q6_K dense weights pass quality but give no speed.

Re-training the MTP head. Head c2, re-trained by recursive self-distillation, was $+4.54$ % faster than the untrained head on the 8 tasks whose answers were in its training corpus (bracket KTC); on 16 unseen tasks the untrained head was $+2.95$ % [$+1.96$.. $+3.93$] faster than c2 (bracket KTG), and a head trained on public prompts only (pub) was -1.63 % [-2.35 .. -0.91] against c2. Details in [Section 4.3](#). The release ships the untrained head; measurement tasks and training corpus must be disjoint.

Rebase onto current upstream. We rebased the fork's changes onto a newer upstream commit. A first bracket (RB2) gave -47.21 %, but the rebased build fitted only 39 instead of 46 cache layers on the GPU. Under equal conditions (bracket RB3b: fit target 18300, 165 slots, 47 cache layers against 46, both 16 threads; reference drift $+2.4$ %) the rebased build is still -5.74 % [-6.87 .. -4.61], across all prompt types (code -5.0 , math -6.0 , physics -9.3 , prose -4.7 %), although the rebased build accepted slightly more drafted tokens (82.1 against 81.1 %). The same bracket RB2 shows that the slot count alone does not explain it: our build with 165 instead of 173 slots -0.23 % [-1.38 .. $+1.08$]. We stay on the older base. Whether the loss comes from upstream changes or from differences in the rebased stack is not decomposed.

Copying part of the misses over PCIe (patch 47). Strata copies about half of the missed experts to the GPU instead of computing them on the CPU. Our version (a three-slot ring per layer, filled ahead of the GPU chain) served up to 40.6 % of the non-cached experts and lost -4.5 % decode speed ([Section 5.3](#)), because the copies delay the hit chain on the same stream. A separate copy stream with one layer of lead would be the next variant; it is not built.

Strata's window graph. Enqueuing a whole verification window on the GPU and signalling layers from the host (Strata's design) was not built: Strata's fixed time per window, 11.9 ms (95 % 8.7–15.4 ms), is not clearly below ours (about 15.2 ms), and the largest fixed item in it is itself a wait for the GPU ([Section 5.3](#)). The hand-off mechanism itself is cheap (1.8 μ s), so the idea stays available if a later measurement shows a gap.

Q2_0 without a vector kernel. Before patch 45 the Q2_0 dot product had only a scalar implementation on x86, which made every file with Q2_0 experts slow on the CPU side: ISTA Q2_0 -25.72 %, FLE-IQ2_M -5.11 % against FLE-IQ3_M at 173 slots; with the AVX2 kernel $+3.84$ % and $+5.49$ % at 173 slots (earlier engine state), and $+8.49$ % and $+11.60$ % in the final measurement at equal budget ([Section 7](#)).

7 Comparison with public quantizations and engines

Every public GGUF of this model that fits the comparison is measured on the same chain: KLD with the neutral meter on the public texts E/W/K ([Section 2.3](#)), and speed with the FLE production configuration, where only the model file differs, in paired brackets against FLE-IQ3_M (groups of up to three candidates). Unsloth UD-Q6_K_XL (169 GB) does not fit into 120 GB of RAM and serves only as a quality ceiling for KLD.

Speed is measured with the final engine (`p49-ring`) and **equal budget**: every file gets the same cache budget in bytes (17,035 MiB, what FLE-IQ3_M uses), and the slot count per layer follows from the expert size. A fixed slot count would not be fair, because a smaller file would leave GPU memory unused (FLE-IQ2_M with FLE-IQ3_M's 173 slots: 11.7 GB of cache instead of 17,035 MiB). Final brackets B3e1–B3e5 (2026-10-06 17:52–20:27, 24 paired requests each, groups of up to three candidates against FLE-IQ3_M, reference drift -0.30 .. $+1.16$ %, no invalid requests; `scripts/fle/bench/end_kette.sh`). FLE-IQ3_M-REAP448 followed on 2026-10-09 in bracket B3e6 (22:26–22:50, same build, budget and protocol, reference drift $+1.64$ %). Quants that contain Q2_0 tensors (ISTA Q2_0, FLE-IQ2_M, FLE-IQ3_XXS) depend on patch 45 ([Section 6](#)), which the final build contains.

Quant	Source	Size (GB)	KLD E	KLD W	KLD K	Top-1 E / W / K (%)	Speed vs FLE-IQ3_M, equal budget	Bracket
UD-Q6_K_XL	Unsloth	169.2	0.0116	0.0164	0.0363	98.5 / 95.1 / 93.0	does not fit	—
AD 4.27 bpw	AtomicChat	94.5	0.079	0.071	0.207	96.0 / 89.9 / 85.7	-22.52 % [-23.28 .. -21.72] ¹	B3e3

Quant	Source	Size (GB)	KLD E	KLD W	KLD K	Top-1 E / W / K (%)	Speed vs FLE-IQ3_M, equal budget	Bracket
UD-IQ4_XS	Unsloth	93.7	0.049	0.046	0.122	96.8 / 91.7 / 88.6	-34.74 % [-35.42 .. -34.04] ¹	B3e1
FLE-IQ3_M	ours	88.9	0.051	0.050	0.129	96.9 / 91.5 / 89.1	reference	all
IQ3_XXS	bartowski	88.0	0.181	0.106	0.292	92.8 / 87.6 / 82.4	-1.10 % [-2.42 .. +0.19]	B3e2
AD 3.84 bpw	AtomicChat	84.9	0.199	0.185	0.447	92.6 / 83.6 / 78.6	-5.99 % [-7.29 .. -4.68] ^{1 2}	B3e2
IQ3_S	ISTA GSQ-RCO	83.6	0.069	0.061	0.183	96.4 / 90.6 / 86.8	-4.33 % [-5.54 .. -3.17]	B3e1
FLE-IQ3_M-REAP448	ours	82.0	0.080	0.086	0.185	96.0 / 89.7 / 87.2	+0.71 % [-0.32 .. +1.75] ⁴	B3e6
UD-IQ3_XXS	Unsloth	82.0	0.091	0.091	0.229	95.5 / 88.3 / 84.6	+8.61 % [+7.39 .. +9.84]	B3e2
FLE-IQ3_XXS	ours	79.5	0.090	0.093	0.227	95.7 / 88.1 / 84.2	-0.69 % [-2.07 .. +0.57]	B3e5
UD-Q2_K_XL	Unsloth	78.9	0.133	0.125	0.299	94.4 / 85.5 / 81.7	+0.57 % [-0.83 .. +1.90]	B3e4
IQ3_XXS	ISTA GSQ-RCO	75.8	0.118	0.112	0.264	95.0 / 87.4 / 83.6	-0.37 % [-2.18 .. +1.40] ³	B3e1
UD-IQ1_M	Unsloth	74.5	0.189	0.186	0.413	92.6 / 83.4 / 79.5	+2.26 % [+0.41 .. +3.94] ³	B3e4
FLE-IQ2_M	ours	71.6	0.160	0.163	0.339	94.0 / 84.7 / 80.9	+11.60 % [+9.78 .. +13.39] ³	B3e5
IQ2_XS	ISTA GSQ-RCO	68.0	0.179	0.191	0.382	93.0 / 83.6 / 80.3	+13.85 % [+11.22 .. +16.67] ³	B3e3
Q2_0	ISTA GSQ-RCO	66.4	0.223	0.236	0.485	92.4 / 82.3 / 77.7	+8.49 % [+6.23 .. +10.76] ³	B3e3

¹ Too large for the budget: only 48 (AD 3.84), 47 (AD 4.27) or 46 (UD-IQ4_XS) of 49 layers fit on the GPU. ² 1 of 24 answers hit the length limit. ³ 2 of 24 answers hit the length limit (FLE-IQ2_M: 3 of 24); FLE-IQ3_M: none. ⁴ Measured cold like all candidates (see below); with its own warm-start profile +0.53 % [-0.59 .. +1.73]. All arms ran with the internal MTP head c2 (Section 4.3). MTP acceptance in B3e5: FLE-IQ3_M 81.6 %, FLE-IQ2_M 81.5 %, FLE-IQ3_XXS 77.9 %. Prefill at 8.6k tokens: 1,197–1,363 tokens/s for all files that fit completely (FLE-IQ3_M 1,233–1,254).

Reading: at equal budget the speed order roughly follows the expert size, as expected for a decoder whose misses run on the CPU, but the CPU kernel per quant type decides the details: ISTA's IQ3_XXS and IQ3_S, the main donors of FLE-IQ3_XXS, are not faster than FLE-IQ3_M despite being 13 and 5 GB smaller, while Unsloth's UD-IQ3_XXS is +8.6 %. Quality follows the KLD columns. FLE-IQ3_XXS matches UD-IQ3_XXS in KLD at 2.5 GB less but is slower (-0.7 % against +8.6 %); FLE-IQ2_M is +11.6 % at a KLD below every public file of similar size; ISTA IQ2_XS is slightly faster (+13.9 %) at a clearly higher KLD. The small files hit the length limit more often in the greedy speed prompts, a hint of repetition loops at low bit widths. The cause of the per-type differences (CPU kernel, slot count, MTP acceptance) is not decomposed.

Cold candidates. The warm-start profile (Section 4.1) belongs to FLE-IQ3_M; every other file rejects it at start (written for a different model) and begins with equal slot counts. In the first 512 steps the cold candidates hit 67–68 % of their lookups in the cache, the warm reference 93.7 %. The warm start is worth +1.49 % [+0.64 .. +2.41] for FLE-IQ3_M (bracket CG), so the candidates in B3e1–B3e5 may be disadvantaged by up to about this amount. B3e6 measured FLE-IQ3_M-REAP448 both ways, cold (+0.71 %) and with the profile its cold arm had written (+0.53 %); with 24 requests the two cannot be told apart. Differences of one to two percent in the table should not be read as a ranking.

Sizes of the public files are the totals of their GGUF splits as listed on Hugging Face; the FLE stages are the sizes of the built files. Publishers' own quality numbers (AtomicChat's KLD on its own corpus, ISTA's task benchmarks) are not comparable to ours and are not used.

Engines: upstream llama.cpp is measured in [Section 5.1](#) and Strata in [Section 5.3](#). ik_llama.cpp and KTransformers were not measured, and we make no claim about their speed on this model.

8 Related work

Offloading MoE experts between GPU and CPU: Fiddler [Kamahori et al., arXiv:2402.07033] orchestrates CPU and GPU so that experts in host memory are computed on the CPU instead of moving their weights; Mixtral offloading [Eliseev & Mazur, arXiv:2312.17238] combines an LRU expert cache with speculative loading and mixed quantization; MoE-Infinity [Xue et al., arXiv:2401.14361] guides expert caching and prefetching by traced activation patterns; HybriMoE [Zhong et al., arXiv:2504.05897] balances expert computation between CPU and GPU with impact-driven prefetching and score-based caching. HOBBIT [Tang et al., arXiv:2411.01433] replaces less critical cache-miss experts with low-precision copies and adds layer-level prefetching and a multidimensional expert cache on top of llama.cpp. MoE-SpAc [Li et al., arXiv:2603.09983] uses speculative decoding as a look-ahead for expert utility that drives prefetching, eviction and CPU/GPU balancing, and SpecMoEOff [Wang et al., arXiv:2508.21706] uses it to raise the work per offloaded expert. Like HOBBIT, FLE is built on llama.cpp; it differs in running IQ-quantized GGUF files of one model on a single 32 GB card, in drafting with the model's own MTP head, and in measuring every step in paired brackets, including an attempt to re-train that head that did not generalize ([Section 4.3](#)).

FLE builds directly on four projects. **ik_llama.cpp** (Kawrakow, MIT): the trellis quant types IQ2/3/4_KT and their CUDA dequantization are taken over essentially verbatim (`NOTICE-ik_llama.md` in the engine repository). **FreeToken** [Yang et al., arXiv:2608.16157; code by FlashML-org, Apache 2.0]: the bandwidth-matched split of missed experts between PCIe transfers and CPU execution, the default rule of its profiler that chooses hybrid execution when the CPU bandwidth exceeds twice the PCIe bandwidth, and the hybrid expert path are re-implemented in C++, with a provenance note in each file. **KTransformers** (Apache 2.0): the idea of computing cold experts with optimized CPU operators next to a GPU-resident hot set; an experimental port of its CPU operators was never part of the build and is not shipped. **Strata** (Niko1221, MIT): the slot-release change ([Section 4.6](#)) and further scheduling ideas, with file and line references in each commit. The pairing of two quant blocks per 512-bit vector in llama.cpp PR #27590 ("ggml-cpu: add AVX-512 and VNNI paths for Q5_K/Q6_K dot products", still open) informed our AVX-512 IQ4_NL dot product on the CPU side of the expert path. Quantization: importance matrix and IQ quant in llama.cpp [Kawrakow, llama.cpp #4861 (importance matrix), #4897 (IQ2 quantization with an importance matrix), #5590 (IQ4_NL), #5676 (IQ3_S, IQ3_M)]; GSQ and RCO as above.

9 Limitations and reproducibility

One model, one machine; effects of policies tuned on this hardware may not transfer. The task benchmarks ([Section 5.4](#)) are single runs with a 81,920-token output limit, without a run of the unquantized model, so they do not separate quantization loss from engine, harness and chance; LiveCodeBench is below Qwen's card. The MKQA probe ([Section 3](#)) checks short factual answers only, with one sampled answer per question, and was run on the pruned files, FLE-IQ3_M and FLE-IQ3_XXS, not on FLE-IQ2_M or the public files. Builds, patches, scripts and raw data are listed in [Appendix A](#); the gaps found in the inventory are listed there too.

Availability. The engine is published as FastLocalExperts at <https://github.com/sirfyyn/FastLocalExperts> under the MIT license of llama.cpp; tag `v1.0` is the configuration measured here (final profile `p49-ring`) plus the slot fix of patch 50 and the crash report of patch 51, neither of which changes the computation. Tag `v1.0.1` has the same engine code plus version 1.0.1 of this report and its archive metadata; Zenodo archives it as doi:10.5281/zenodo.23272400 (<https://doi.org/10.5281/zenodo.23272400>, all versions). FLE-IQ3_M, FLE-IQ3_XXS, FLE-IQ2_M, the MTP head, the films and the SHA-256 of every file are at <https://huggingface.co/sirfyyn/Qwen3.8-Flash-Next-FLE-GGUF>; FLE-IQ3_M-REAP448 and its SHA-256 are at

<https://huggingface.co/sirfyn/Qwen3.8-Flash-Next-FLE-REAP448-GGUF> (moved there on 2026-10-10 so that the Hub lists it on its own). All model files are under the Qwen Community License 1.0 of the base model. The raw measurement data are not part of this release.

Appendix A: evidence table

State of the inventory of 2026-10-03, extended 2026-10-07. Paths are on the measurement machine unless they start with `docs/` or `scripts/` (this repository). Snapshot of the scripts: `scripts/fle/titan-stand-20261003/`.

Claim	Raw data	Evaluation	Bracket spec	Status
END2 +58.53 %	<code>/mnt/perf/rhea-m2-20260929/</code> (END2 runs)	<code>docs/fle/paper/</code> <code>belege/END2-</code> <code>gepaart.txt</code>	missing	recomputed 2026-10-03
ALL2 overlap / MTP head	same	<code>docs/fle/paper/</code> <code>belege/ALL2-</code> <code>gepaart.txt</code>	<code>m2spec-ALL2.txt</code>	recomputed 2026-10-03
P35Q micro-batch 1024	same	<code>docs/fle/paper/</code> <code>belege/P35Q-</code> <code>gepaart.txt</code>	<code>m2spec-P35Q.txt</code>	recomputed; <code>compare-P35Q.json</code> is empty
NM2 n-max 1/3/6	same	<code>docs/fle/paper/</code> <code>belege/NM2-</code> <code>gepaart.txt</code>	missing	invalid (drift -5.88 %), not cited
HZ2 cache policy +12.15 %	same	<code>docs/fle/paper/</code> <code>belege/HZ2-</code> <code>gepaart.txt</code>	missing	recomputed 2026-10-04, confirmed
VK 32k vocabulary +4.14 %	same	<code>docs/fle/paper/</code> <code>belege/VK-</code> <code>gepaart.txt</code>	<code>m2spec-VK.txt</code>	recomputed 2026-10-04, confirmed
VL pre-gating -9.67 %	same	<code>docs/fle/paper/</code> <code>belege/VL-</code> <code>gepaart.txt</code>	<code>m2spec-VL.txt</code>	recomputed 2026-10-04; drift -1.28 % (status document right, compare file wrong)
NG n-gram drafting	same	<code>docs/fle/paper/</code> <code>belege/NG-</code> <code>gepaart.txt</code>	<code>m2spec-NG.txt</code>	recomputed 2026-10-04
KTC head c2 +4.54 % (on its training tasks)	<code>/mnt/perf/rhea-mtp-train-20261001/</code>	<code>docs/fle/paper/</code> <code>belege/KTC-</code> <code>gepaart.txt</code>	<code>m2spec-KTC.txt</code>	recomputed 2026-10-04; corrected from +4.92 % (not reproducible; c3 gives +4.90 %)
KTF untrained head +1.61 %, pub -0.11 % vs c2 (unseen tasks, separate warm-start profiles)	<code>/mnt/perf/rhea-mtp-train-20261001/</code>	<code>docs/fle/paper/</code> <code>belege/KTF-</code> <code>gepaart.txt</code>	—	confounded by the separate profiles; repeated as KTG
KTG untrained head +2.95 %, pub -1.63 % vs c2 (unseen tasks, same profile)	<code>/mnt/perf/rhea-mtp-train-20261001/</code>	<code>docs/fle/paper/</code> <code>belege/KTG-</code> <code>gepaart.txt</code>	—	measured 2026-10-09 15:14–16:09; basis of the head choice
B1b mainline -61.29 % (FLE 2.58×, <code>p43-pin</code>)	<code>/mnt/perf/rhea-m2-20260929/</code> (B1b runs)	<code>docs/fle/paper/</code> <code>belege/B1b-</code> <code>gepaart.txt</code>	<code>scripts/fle/bench/b1_kette.sh</code>	measured 2026-10-05; superseded by B1e
B1e mainline ub1024 -59.87 %, ub4096 -64.01 %, prefill (Section 5.1)	<code>/mnt/perf/fle-end/</code>	B1e- <code>gepaart.txt</code> , B1e- <code>prefill.txt</code>	<code>scripts/fle/bench/end_kette.sh</code>	final measurement 2026-10-06
B3e1–B3e5 all quants, equal budget (Section 7)	<code>/mnt/perf/fle-end/</code>	B3e<n>- <code>gepaart.txt</code> , B3e<n>- <code>prefill.txt</code>	<code>scripts/fle/bench/end_kette.sh</code>	final measurement 2026-10-06

Claim	Raw data	Evaluation	Bracket spec	Status
S Strata vs FLE p49-ring (Section 5.3)	/mnt/perf/fle-end/	S-auswertung.txt, scripts/fle/fusion/v12_strata_fle.py	scripts/fle/bench/end_kette.sh	final measurement 2026-10-06, unpaired means of per-task medians
W7 ubatch lending (patch 48)	/mnt/perf/fle-release/	W7-gepaart.txt, W7-prefill.txt	scripts/fle/fusion/w7_kette.sh	measured 2026-10-06
W49c prefetch ring (patch 49), W49n/W49n2 timelines	/mnt/perf/fle-release/	w49_kette.sh output, w49n_fenster.py	scripts/fle/fusion/w49_kette.sh	measured 2026-10-06
PC PCIe share of misses (patch 47) -4.46 %	/mnt/perf/fle-release/	PC-gepaart.txt	scripts/fle/fusion/pc_kette.sh	measured 2026-10-06
Slot contamination and patch 50 (Section 5.4)	/mnt/perf/rhea-bench/ungueutig-20261007/	scripts/fle/bench/slot_kanarie_n.py	scripts/fle/bench/rhea_p50_umstellen.sh	canary 3/3 flips before, 0/3 after, 2026-10-07
Standard benchmarks (Section 5.4)	/mnt/perf/rhea-bench/{ifbench,gpqa,lcb}/(gen.jsonl, gen.log), chain log kette.log	ifbench/eval.txt, gpqa/ergebnis.json, lcb/ergebnis.json, lcb/score.txt; token statistics from */gen.log	scripts/fle/bench/bench_kette.sh, rhea_gen.py, bench_daten.py	measured 2026-10-07/08 on p50-fix; one run each; raw data not copied into the repository
mainline + MTP not loadable	—	—	scripts/fle/bench/b1_mtp_probe.sh, b1_mtp_probe_gpu0.sh	smoke test 2026-10-05
HY slot count (FLE-IQ3_M against UD-IQ4_XS)	/mnt/perf/rhea-m2-20260929/ (HY runs; mem_high_events in the block manifests)	docs/file/paper/belege/HY-gepaart.txt	missing	invalid: memory-high events in all three candidate arms and none in the reference arms, output limit 4,096 instead of 8,192, 19 paired requests; not cited (Section 4.1)
RB2 16 threads +5.23 %, rebase -47.21 %	/mnt/perf/fle-rebase-probe-20261002/	docs/file/paper/belege/RB2-gepaart.txt	—	measured 2026-10-05
RB3b rebase fair -5.74 %	same	docs/file/paper/belege/RB3b-gepaart.txt	scripts/fle/strata-20261005/rb3_kette.sh	measured 2026-10-05
Pinned experts (W3 stage 1b)	/mnt/perf/fle-fusion/messungen/w3s1b-20261005-141156/	scripts/fle/fusion/w3_klammer.py	—	alternating arms, medians; the decode gain was not repeated in a paired bracket
Patch 43 slot release -202.5 ms	/mnt/perf/fle-fusion/messungen/w1-20261005-130045/ (zusammenfassung.txt, ergebnis.json)	docs/current/RH-EA-STRATA-UEBERNAHME.md	—	alternating arms (24 slot switches per arm); bootstrap 95 % over pooled medians -217 .. -164 ms
Public texts E/W/K	/mnt/perf/b6-texte/ (quellen.json with SHA-256)	scripts/fle/bench/b6_texte.py	—	built 2026-10-05
KLD comparison (Section 7)	/mnt/perf/b2-kld/oeff/ (kld-#{E,W,K}.log)	scripts/fle/bench/b2_kld.sh	—	measured 2026-10-05/06, incl. UD-Q6_K_XL
Expert pruning REAP/MAN, KLD and MKQA probe (Section 3)	/mnt/perf/rhea-reap/ (reap-g3.tsv, kld-#{E,W,K}.log, proben-*.jsonl, *.manifest.json, kette.log)	ergebnis.txt / ergebnis.json (bin/reap_auswerten.py)	bin/reap_kette.sh, reap_kalib.py, reap_proben_baue_n.py, reap_probe.py	measured 2026-10-08 21:14-23:01; G3/FLE-Q3 KLD taken from /mnt/perf/b2-kld/oeff/

Claim	Raw data	Evaluation	Bracket spec	Status
FLE-IQ3_XXS / FLE-IQ2_M selection and build (built as Q3 / Q2; the Q1 class was dropped)	/mnt/fastmodels/b8-hybride/ (FLE-Q3.auswahl.json , FLE-Q3.bau.log), /mnt/perf/b8-hybride/ (FLE-Q2.*, FLE-Q1.bau.log), chain log /mnt/perf/b2-kld/oeff/kette.log	scripts/file/bench/b8_auswahl.py (released as tools/fle-hybrid/fle_select.py), gguf_splice.py	—	built 2026-10-05 (FLE-IQ3_XXS 16:52–16:54, FLE-IQ2_M 18:26–18:28), 79,483,799,936 and 71,617,465,600 bytes; SHA-256 of the build files in rhea-publish/SHA256SUMS (2026-10-08) and of the published splits in rhea-publish/hf/SHA256SUMS (2026-10-09); the layer provenance check of 2026-10-07 is described in the model card, its raw output was not archived
Speed comparison, 173 slots (Section 7)	/mnt/perf/b3-tempo/	B3m<n>-gepaart.txt , B3w1-gepaart.txt	scripts/file/bench/b3_kette.sh	measured 2026-10-05/06; superseded by B3e
Speed comparison, equal budget (Section 7)	/mnt/perf/b3-tempo/ , /mnt/perf/fle-fusion/tore/	B3s1 – B3s4-gepaart.txt , B3sP-gepaart.txt	scripts/file/fusion/b3sp_kette.sh	measured 2026-10-06; superseded by B3e
Patch 45 AVX2 Q2_0	/mnt/perf/fle-fusion/tore/	P45-gepaart.txt	—	measured 2026-10-06
Strata vs FLE: decode, prefill, TTFT, window timing (Section 5.3)	/mnt/perf/fle-fusion/tore/ (v12.jsonl)	scripts/file/fusion/v12_auswert.py , v1_achse.py	scripts/file/fusion/tore_kette.sh	measured 2026-10-06, unpaired means of per-task medians
Doorbell hand-off 1.8 µs	/mnt/perf/fle-fusion/tore/	—	a0_doorbell.cu	measured 2026-10-06 (16 GB card)
Patch 44 budget and warm start (bracket CG)	/mnt/perf/fle-fusion/tore/	CG-gepaart.txt	scripts/file/fusion/cg_kette.sh	measured 2026-10-06
Miss sensitivity (bracket V6b)	/mnt/perf/fle-fusion/tore/	V6b-gepaart.txt	scripts/file/fusion/v6_nach.sh	measured 2026-10-06
FLE-IQ3_M KLD (F/E/V), acceptance (Section 3)	decode mode, own build, GPU0 (2026-09-30): rhea-kld-20260929/kld-dec14-{G3-CB001ADMEVs173,U-CTRL}-{F,E,V}.log ; files alone, own build: same folder, kld-G3-*.log , kld-U-*.log , kld-moe2-U-CTRL-V.log ; neutral meter, GPU1 (2026-10-05): /mnt/perf/b2-kld/kld-{g3,unsloth-iq4xs}-{F,E,V}.log	—	—	verified 2026-10-09 against the logs; the folder rhea-kld-20260929 moved to the cold store on TrueNAS (/mnt/Files/titan-coldstore/rhea-quant-history-20260930/ , manifest manifest.tsv)
Rejected quantization candidates (Section 3)	cold store rhea-kld-20260929/: kld-{CTRL,HCQ8,FRONTIER}-F.log (own full quantizations), kld-G3q5-*.log , kld-half-U-K8-*.F.log , kld-dec14-G3-CB002ADMEVs173-*.log , kld-G4-*.log , kld-G6-*.log	—	hybrid_candidate_s2.json (G3 = budget 0.12, G4 = 0.25), hybrid_candidate_s2_cap007.json (G6)	verified 2026-10-09 against the logs
FLE-IQ3_M selection recomputed with the knapsack (Section 3)	/mnt/perf/fle-end/g3probe.auswahl.json	docs/file/paper/belege/g3probe.auswahl.json	scripts/file/bench/b8_auswahl.py (target 88.87 GB, sources UD-IQ4_XS, ISTA IQ3_S, ISTA IQ3_XXS; predictors rhea-heal-20260929/qs*/quant_sensitivity.json)	computed 2026-10-09; compared with the layer lists of hybrid_candidates2.json budget 0.12

Claim	Raw data	Evaluation	Bracket spec	Status
Noise floor (Section 2.3)	/mnt/perf/rhea-p35-ursache-20261003/ (kld-g1aus-V.log , kld-g1aus2-V.log , kld-fa- aus-V.log , kld-c4-ub2048-V.log); GPU0 value from the cold store, kld-G3-V.log	—	scripts/fle/titan-kld-20260929/kld_g1_p35*.sh	measured 2026-10-03 (GPU0 value 2026-09-30)
Prefill at 8.6k about 425 against 280 tokens/s (patch 35), follow-up turn 2.4 s / 17.5 s (patches 39/39b)	/mnt/perf/rhea-p35-reihe-20261003/ (pf-p35aus-512.txt , pf-p35an-1024.txt , prefill.jsonl), /mnt/perf/rhea-p35p-reihe-20261003/ (pf-p1024ft.txt , reihe.log); follow-up turn: rhea.service journal of 2026-10-03 17:08 and /mnt/perf/rhea-zeitprofil-20261003/folgeturn.log	prefill- kurz.txt in both folders; docs/fle/paper/belege/folgeturn_n.log	docs/fle/paper/belege/folgeturn_probe_20261003.py	measured 2026-10-03; two cold runs per arm; the follow-up turn is a single probe whose output went to the terminal, its numbers come from the journal and the observer log
Cycle model and Nsight profile (Section 4.4)	NM blocks /mnt/perf/rhea-m2-20260929/blocks/NM-*/results.jsonl , compare-MA.json ; /mnt/perf/rhea-zeitprofil-20261003/ (ns-prod.txt , uebergaben-prod.txt)	docs/fle/rhea-verifikation-mathe-2026-09-30.md (cycle model); docs/fle/paper/belege/ns-prod.txt , uebergaben-prod.txt	—	NM measured 2026-09-30 (base build), profile 2026-10-04 22:37 (production build rhea-prod-20261003); cycles recomputed 2026-10-09
Pre-gating simulation and router recall (Section 6)	trace rhea-expert-trace-long-20260923 ; /mnt/perf/rhea-pregating-20261003/20261003-020130.ergebnis.txt	scripts/fle/verify_sim_ab.py , output docs/fle/verify-math-2026-09-30/verify_sim_ab-2026-10-02.txt	—	simulation 2026-10-02; extra copies not modelled
FLE-IQ3_M-REAP448 speed, cold and warm (bracket B3e6)	/mnt/perf/fle-end/ (pause-20261009.log , profil-B3e6-*.txt), blocks /mnt/perf/rhea-m2-20260929/blocks/B3e6-*	docs/fle/paper/belege/B3e6-gepaart.txt	m2spec-B3e6.txt , pause_b3e6.sh	measured 2026-10-09 22:26-22:50 with the p49-ring build; the old compare-quant step refuses brackets whose reference arms load different slot plans (compare-B3e6.json is empty), as in all budget brackets
Engine build	/mnt/perf/fle-p3334-20260930/	—	—	build folders are not git repositories; patches 26/29/37/38/39 not in the overlay; manifest missing

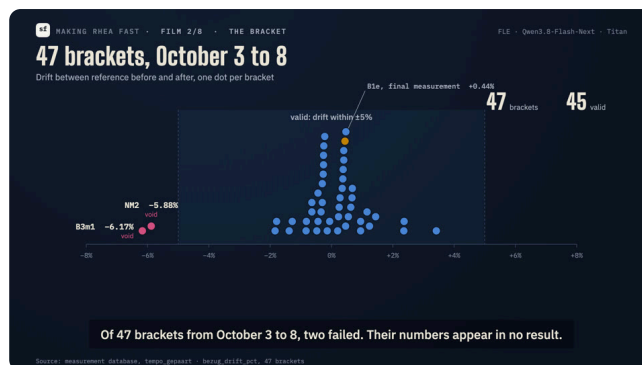
Appendix B: supplementary films

Eight short films, *Making Rhea Fast* (English, subtitles burned in, 6 min 59 s in total), retell [Sections 2.4](#) to 6 for readers who prefer to watch. They were generated from the measurement database (6,130 measurements, state of 2026-10-08) and contain no number that is not in this paper or its evidence; where a film words or rounds a number differently, the text and [Appendix A](#) are the reference. Files and SHA-256 are in the model repository under `media/` (<https://huggingface.co/sirfynn/Qwen3.8-Flash-Next-FLE-GGUF/tree/main/media>); the same films are on YouTube as one playlist (<https://www.youtube.com/playlist?list=PLMYhpsBSEhIQ>).

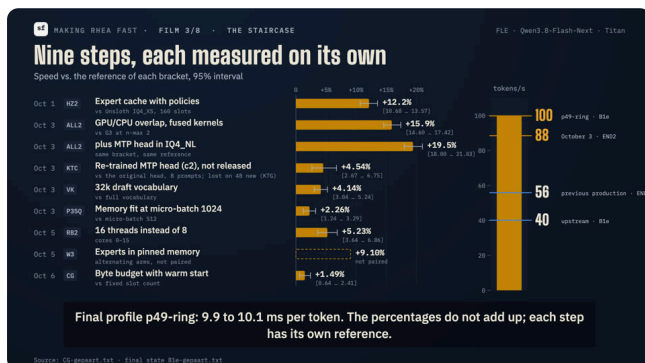
Film	Retells	File
1 The race	Section 5.1 (B1e), 5.2 (END2), 5.3	<code>making-rhea-fast-1.mp4</code>
2 The bracket	Section 2.4 , the 47 brackets of 2026-10-03 to 08 and the two void ones (NM2, B3m1)	<code>making-rhea-fast-2.mp4</code>
3 The staircase	Sections 4.1 to 4.6, step by step	<code>making-rhea-fast-3.mp4</code>
4 The expert cache	Sections 4.1 and 4.4, bracket V6b, warm start (CG)	<code>making-rhea-fast-4.mp4</code>
5 Draft and verify	Sections 4.3 and 4.4; shows how the re-trained head c2 learned (made before brackets KTF/KTG; the release ships the untrained head because c2's gain held only on its training tasks, Section 4.3)	<code>making-rhea-fast-5.mp4</code>
6 What did not work	Section 6	<code>making-rhea-fast-6.mp4</code>
7 The rebench	Section 5.4	<code>making-rhea-fast-7.mp4</code>
8 Why we do this	Section 2.1 , the release	<code>making-rhea-fast-8.mp4</code>
all eight in one file		<code>making-rhea-fast-complete.mp4</code>



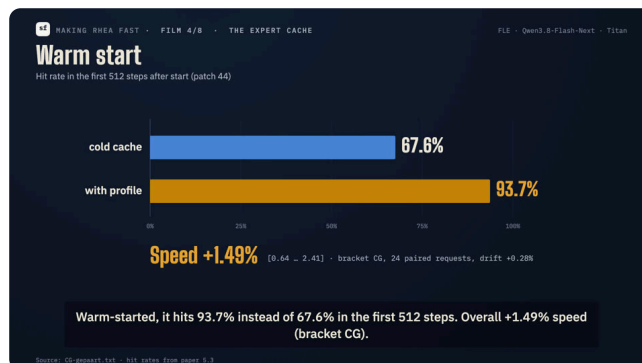
Film 1/8 · The race · 0:33

youtu.be/g4fWtuupwHo · making-rhea-fast-1.mp4


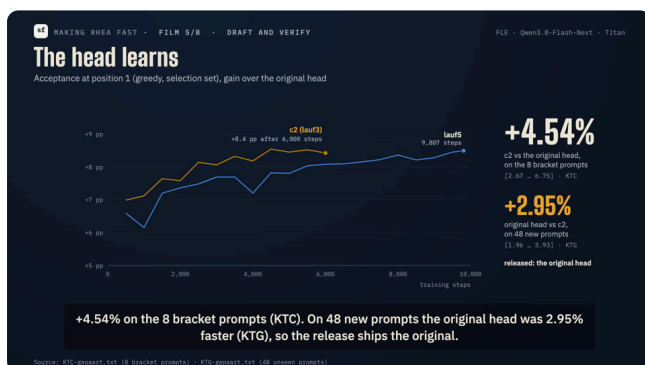
Film 2/8 · The bracket · 0:34

youtu.be/Bf6UE6DU2b4 · making-rhea-fast-2.mp4


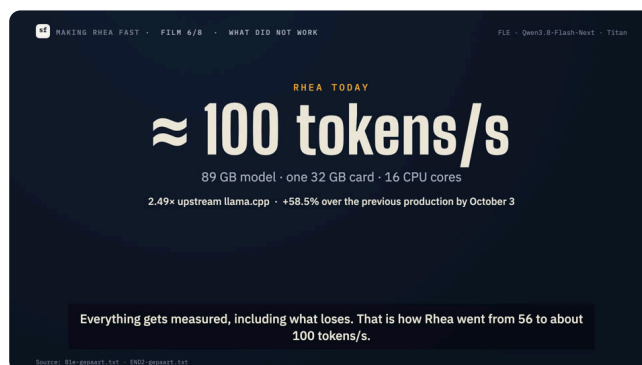
Film 3/8 · The staircase · 1:03

youtu.be/cEv8TZ9OZPA · making-rhea-fast-3.mp4


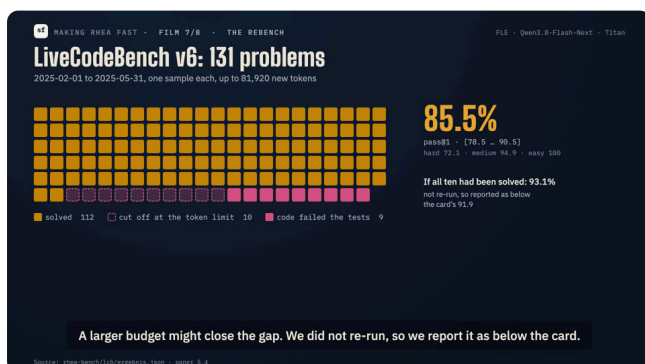
Film 4/8 · The expert cache · 0:59

youtu.be/tY-tDv-9uzY · making-rhea-fast-4.mp4


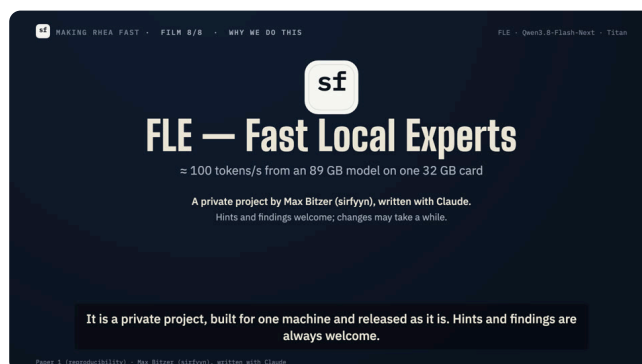
Film 5/8 · Draft and verify · 1:00

youtu.be/smgBd6764m8 · making-rhea-fast-5.mp4


Film 6/8 · What did not work · 0:56

youtu.be/EQAaw5UroaSS · making-rhea-fast-6.mp4


Film 7/8 · The rebench · 0:58

youtu.be/YNLKC_LNz44 · making-rhea-fast-7.mp4


Film 8/8 · Why we do this · 0:58

youtu.be/2ZznZt1iCUw · making-rhea-fast-8.mp4

 All 8 in one file: [making-rhea-fast-complete.mp4](#) (6:59, 43.1 MB). Playlist: <https://www.youtube.com/playlist?list=PLMYhpsBSEhIQ>

Film 6 was corrected on 2026-10-10. Checking the paper against the raw data after the films were made found two errors in its first version: it gave the simulated gain of pre-gating as +18 ... +35 %, but +35 % holds only for two parallel slots, a single stream gives +18 .. +25 % (Section 6); and it gave a forecast of ± 1.5 % for copying misses over PCIe, while the forecast written down before that measurement was +3 .. +10 % (Section 5.3). Film 6 and the complete film now show +18 .. +25 % and +3 .. +10 %.

Appendix C: all paired brackets

Every paired bracket whose evaluation is in `docs/fle/paper/belege/`, in the order of measurement. *Speed* is the paired change in decode speed of the candidate against the reference (positive = faster) with its bootstrap 95 % interval; *drift* is the change in time per token of the reference between its run before and its run after the candidates; *n* is the number of paired requests. A bracket is valid only if the drift stays within ± 5 % (Section 2.4). *Cited in* lists the sections that name the bracket (A and B are the appendices). Dates are local time (CEST) of the first reference run. The table is generated by `scripts/fle/release/klammer_tabelle.py` from the evaluations and `belege/klammern.json`. Strata against FLE (S, Section 5.3) and the cycle-model runs (NM, Section 4.4) are not paired brackets and are not listed.

Bracket	Date	Reference (drift)	Candidate	Speed vs reference [95 % CI]	n	Cited in
ALL2	2026-09-30	FLE-IQ3_M, n-max 2, without overlap and fused kernels (-0.44 %)	overlap + fused kernels	+15.93 % [+14.60 .. +17.42]	24	4.2, A
ALL2			same + IQ4_NL MTP head	+19.46 % [+18.00 .. +21.03]	24	
ALL2			same + Q4_0 MTP head	+16.63 % [+15.25 .. +18.14]	24	
ALL2			IQ4_NL head + Q6_K dense weights	+19.14 % [+15.76 .. +22.14]	24	
HY	2026-09-30	Unsloth UD-IQ4_XS, 160 slots (+3.42 %); <i>invalid: memory-high events in all candidate arms, output limit 4,096, 19 requests</i>	FLE-IQ3_M, 160 slots	-5.66 % [-9.11 .. -2.70]	19	4.1, A
HY			same + cache bias and policy	+3.66 % [-1.53 .. +8.60]	19	
HY			FLE-IQ3_M, 173 slots + cache bias	+6.53 % [+4.92 .. +8.24]	19	
HZ2	2026-09-30	Unsloth UD-IQ4_XS, 160 slots, n-max 1 (+0.38 %)	FLE-IQ3_M alone	+4.05 % [+2.92 .. +5.19]	24	4.1, A
HZ2			FLE-IQ3_M + cache bias and policy	+12.15 % [+10.68 .. +13.57]	24	
NG	2026-09-30	FLE-IQ3_M, cache policy, MTP n-max 2 (-0.44 %)	n-gram drafting (ngram-map-k) before MTP	-13.77 % [-18.86 .. -8.58]	24	6, A
NG			n-gram drafting (ngram-mod) before MTP	-23.34 % [-30.02 .. -16.12]	24	
NM2	2026-09-30	FLE-IQ3_M, MTP n-max 2 (-5.88 %); <i>invalid: reference drift -5.88 %</i>	n-max 1	-6.74 % [-8.38 .. -5.08]	24	4.3, A, B
NM2			n-max 3, p-min 0.7	-1.29 % [-3.11 .. +0.50]	24	
NM2			n-max 6, p-min 0.7	-30.72 % [-33.91 .. -27.28]	24	
END2	2026-10-01	previous production: stock build, Unsloth UD-IQ4_XS (+1.44 %)	FLE: FLE-IQ3_M, cache policy, overlap, IQ4_NL head, n-max 2	+58.53 % [+55.83 .. +61.58]	24	4.5, 4.6, 5.2, A, B
END2			same, micro-batch 1024 without memory fit	-20.36 % [-22.32 .. -18.36]	24	
KTC	2026-10-03	untrained MTP head, n-max 2 (-1.84 %)	re-trained head c2 (training tasks)	+4.54 % [+2.67 .. +6.75]	24	4.3, 6, A

Bracket	Date	Reference (drift)	Candidate	Speed vs reference [95 % CI]	n	Cited in
KTC			head c2, n-max 3	+4.90 % [+2.16 .. +8.19]	24	
KTC			head b, n-max 3	+5.59 % [+2.66 .. +8.78]	24	
P35Q	2026-10-03	micro-batch 512 (−0.30 %)	lean prefill, micro-batch 1024	−19.01 % [−19.76 .. −18.26]	24	4.5, A
P35Q			same, memory fit 18806 MiB	+2.26 % [+1.24 .. +3.29]	24	
VK	2026-10-03	head c2, full draft vocabulary (+1.24 %)	16k draft vocabulary	+2.88 % [+1.57 .. +4.03]	24	4.3, A
VK			32k draft vocabulary	+4.14 % [+3.04 .. +5.24]	24	
VK			32k draft vocabulary, n-max 3	+4.09 % [+1.98 .. +6.08]	24	
VL	2026-10-03	production p38 (−1.28 %)	pre-gating prefetch, width 10	−9.67 % [−11.14 .. −8.22]	24	6, A
VL			width 10, 8 prefetch slots	−12.49 % [−13.84 .. −11.21]	24	
VL			width 16, 8 prefetch slots	−27.34 % [−28.81 .. −25.83]	24	
KTE	2026-10-04	head c2, n-max 2 (−0.83 %)	larger re-training (lauf5), n-max 2	+0.28 % [−0.73 .. +1.48]	48	4.3
KTE			same, n-max 3	+1.22 % [−0.38 .. +2.91]	48	
KTE			same, n-max 4	−0.58 % [−3.19 .. +2.06]	48	
RB2	2026-10-05	production, 8 threads, 173 slots (−1.79 %)	MTP penalties off	−0.62 % [−1.92 .. +0.86]	24	4.6, 6, A
RB2			rebased build	−47.21 % [−48.20 .. −46.12]	24	
RB2			165 slots	−0.23 % [−1.38 .. +1.08]	24	
RB2			16 threads on cores 0-15	+5.23 % [+3.64 .. +6.86]	24	
RB3b	2026-10-05	production, 16 threads (+2.36 %)	rebased build, fit target 18300, 165 slots	−5.74 % [−6.87 .. −4.61]	24	6, A
B1b	2026-10-05	FLE p43-pin (+0.40 %); <i>superseded by B1e</i>	upstream llama.cpp a8c9a4e7c, no MTP	−61.29 % [−62.56 .. −60.11]	24	A
B3m1	2026-10-05	FLE-IQ3_M, 173 slots per layer (−6.17 %); <i>invalid: reference drift −6.17 %; fixed slot count, superseded by B3e</i>	Unsloth UD-IQ4_XS, 173 slots	−31.38 % [−32.22 .. −30.53]	24	A, B
B3m1			ISTA IQ3_S, 173 slots	−4.73 % [−5.86 .. −3.62]	24	
B3m1			ISTA IQ3_XXS, 173 slots	−6.96 % [−8.52 .. −5.35]	24	
B3m2	2026-10-05	FLE-IQ3_M, 173 slots per layer (−0.27 %); <i>fixed slot count, superseded by B3e</i>	Unsloth UD-IQ3_XXS, 173 slots	+7.21 % [+5.89 .. +8.36]	24	A
B3m2			bartowski IQ3_XXS, 173 slots	+1.37 % [+0.23 .. +2.45]	24	
B3m2			AtomicChat AD 3.84 bpw, 173 slots	−10.55 % [−15.12 .. −7.73]	24	

Bracket	Date	Reference (drift)	Candidate	Speed vs reference [95 % CI]	n	Cited in
B3m3	2026-10-05	FLE-IQ3_M, 173 slots per layer (−0.14 %); <i>fixed slot count, superseded by B3e</i>	AtomicChat AD 4.27 bpw, 173 slots	−22.64 % [−23.41 .. −21.79]	24	6, A
B3m3			ISTA IQ2_XS, 173 slots	−6.45 % [−9.64 .. −3.36]	24	
B3m3			ISTA Q2_0, 173 slots, scalar Q2_0 kernel	−25.72 % [−28.97 .. −22.37]	24	
B3m4	2026-10-05	FLE-IQ3_M, 173 slots per layer (−0.26 %); <i>fixed slot count, superseded by B3e</i>	Unsloth UD-Q2_K_XL, 173 slots	−1.97 % [−2.91 .. −1.02]	24	A
B3m4			Unsloth UD-IQ1_M, 173 slots	−0.47 % [−1.98 .. +0.95]	24	
B3m5	2026-10-06	FLE-IQ3_M, 173 slots per layer (−0.64 %); <i>fixed slot count, superseded by B3e</i>	FLE-IQ3_XXS, 173 slots	−6.10 % [−7.35 .. −4.93]	24	6, A
B3m5			FLE-IQ2_M, 173 slots, scalar Q2_0 kernel	−5.11 % [−6.72 .. −3.42]	24	
B3w1	2026-10-06	FLE-IQ3_M, 173 slots per layer (+0.20 %); <i>repeat of B3m1, superseded by B3e</i>	Unsloth UD-IQ4_XS, 173 slots	−34.46 % [−35.11 .. −33.83]	24	A
B3w1			ISTA IQ3_S, 173 slots	−7.74 % [−8.81 .. −6.81]	24	
B3w1			ISTA IQ3_XXS, 173 slots	−9.29 % [−10.82 .. −7.73]	24	
CG	2026-10-06	production p43-pin, 173 slots (+0.28 %)	patch-44 build without budget	−0.08 % [−0.57 .. +0.37]	24	4.1, 4.6, 5.3, 7, A, B
CG			byte budget 17,035 MiB, no profile	+0.01 % [−0.49 .. +0.48]	24	
CG			byte budget + warm-start profile	+1.49 % [+0.64 .. +2.41]	24	
CG			budget 2.6 GB smaller (14,435 MiB) + profile	−3.26 % [−4.19 .. −2.32]	24	
B3s1	2026-10-06	FLE-IQ3_M, 173 slots per layer (+0.55 %); <i>earlier engine state, superseded by B3e</i>	FLE-IQ3_XXS, equal budget	−1.83 % [−2.69 .. −0.97]	24	A
B3s1			FLE-IQ2_M, equal budget	+5.13 % [+3.22 .. +7.06]	24	
B3s1			Unsloth UD-IQ3_XXS, equal budget	+12.06 % [+10.89 .. +13.33]	24	
B3s2	2026-10-06	FLE-IQ3_M, 173 slots per layer (+0.47 %); <i>earlier engine state, superseded by B3e</i>	ISTA IQ3_S, equal budget	−5.96 % [−7.03 .. −4.93]	24	A
B3s2			ISTA IQ3_XXS, equal budget	−2.74 % [−4.57 .. −0.90]	24	
B3s3	2026-10-06	FLE-IQ3_M, 173 slots per layer (−0.64 %); <i>earlier engine state, superseded by B3e</i>	Unsloth UD-Q2_K_XL, equal budget	+2.75 % [+1.28 .. +4.11]	24	A
B3s3			Unsloth UD-IQ1_M, equal budget	+4.64 % [+3.29 .. +6.03]	24	
B3s3			AtomicChat AD 3.84 bpw, equal budget	−3.93 % [−5.11 .. −2.78]	24	
B3s4	2026-10-06	FLE-IQ3_M, 173 slots per layer (−0.25 %); <i>earlier engine state, superseded by B3e</i>	ISTA IQ2_XS, equal budget	+8.84 % [+6.23 .. +11.56]	24	A

Bracket	Date	Reference (drift)	Candidate	Speed vs reference [95 % CI]	n	Cited in
B3s4			ISTA Q2_0, equal budget	-5.14 % [-8.63 .. -1.85]	24	
P45	2026-10-06	FLE-IQ3_M, 173 slots per layer (+0.95 %)	FLE-IQ3_XXS, AVX2 Q2_0 kernel (patch 45)	-2.68 % [-4.29 .. -1.04]	24	6, A
P45			FLE-IQ2_M, AVX2 Q2_0 kernel	+5.49 % [+3.13 .. +7.86]	24	
P45			ISTA Q2_0, AVX2 Q2_0 kernel	+3.84 % [+1.43 .. +6.22]	24	
V6b	2026-10-06	FLE-IQ3_M, 173 slots per layer (+0.68 %)	146 slots	-4.39 % [-5.43 .. -3.37]	24	4.1, 4.4, 5.3, A, B
V6b			120 slots	-10.04 % [-10.74 .. -9.37]	24	
PC	2026-10-06	production p45-pin (+2.37 %)	patch 47, PCIe share 0.35	-4.46 % [-5.73 .. -3.32]	24	5.3, A
PC			patch 47, PCIe share 0.55	-4.47 % [-5.53 .. -3.46]	24	
W7	2026-10-06	production p45-pin (-0.26 %)	ubatch lending, batch/micro-batch 4096	-0.25 % [-1.15 .. +0.66]	24	4.6, 5.3, A
W7			ubatch lending, batch 4096, micro-batch 2048	-0.81 % [-1.65 .. +0.05]	24	
W49c	2026-10-06	build with patch 48 (p48) (+0.35 %)	prefetch ring (patch 49)	-0.18 % [-0.91 .. +0.49]	24	4.6, 5.3, A
B1e	2026-10-06	FLE p49-ring (+0.44 %)	upstream llama.cpp a8c9a4e7c, micro-batch 1024, no MTP	-59.87 % [-61.24 .. -58.50]	24	4.3, 5.1, A, B
B1e			same, micro-batch 4096	-64.01 % [-65.20 .. -62.86]	24	
B3e1	2026-10-06	FLE-IQ3_M, p49-ring, budget 17,035 MiB (+0.37 %)	Unslloth UD-IQ4_XS	-34.74 % [-35.42 .. -34.04]	24	7, A
B3e1			ISTA IQ3_S	-4.33 % [-5.54 .. -3.17]	24	
B3e1			ISTA IQ3_XXS	-0.37 % [-2.18 .. +1.40]	24	
B3e2	2026-10-06	FLE-IQ3_M, p49-ring, budget 17,035 MiB (-0.30 %)	Unslloth UD-IQ3_XXS	+8.61 % [+7.39 .. +9.84]	24	7
B3e2			bartowski IQ3_XXS	-1.10 % [-2.42 .. +0.19]	24	
B3e2			AtomicChat AD 3.84 bpw	-5.99 % [-7.29 .. -4.68]	24	
B3e3	2026-10-06	FLE-IQ3_M, p49-ring, budget 17,035 MiB (+0.25 %)	AtomicChat AD 4.27 bpw	-22.52 % [-23.28 .. -21.72]	24	7
B3e3			ISTA IQ2_XS	+13.85 % [+11.22 .. +16.67]	24	
B3e3			ISTA Q2_0	+8.49 % [+6.23 .. +10.76]	24	
B3e4	2026-10-06	FLE-IQ3_M, p49-ring, budget 17,035 MiB (+0.23 %)	Unslloth UD-Q2_K_XL	+0.57 % [-0.83 .. +1.90]	24	7
B3e4			Unslloth UD-IQ1_M	+2.26 % [+0.41 .. +3.94]	24	
B3e5	2026-10-06	FLE-IQ3_M, p49-ring, budget 17,035 MiB (+1.16 %)	FLE-IQ3_XXS	-0.69 % [-2.07 .. +0.57]	24	3, 4.1, 7, A
B3e5			FLE-IQ2_M	+11.60 % [+9.78 .. +13.39]	24	

Bracket	Date	Reference (drift)	Candidate	Speed vs reference [95 % CI]	n	Cited in
KTF	2026-10-09	head c2, unseen tasks, own warm-start profile per arm (−1.11 %); <i>confounded by separate profiles, repeated as KTG</i>	untrained head	+1.61 % [+0.88 .. +2.36]	48	4.3, A, B
KTF			head pub (public prompts only)	−0.11 % [−1.11 .. +0.86]	48	
KTG	2026-10-09	head c2, unseen tasks, same profile for all arms (−2.38 %)	untrained head (released)	+2.95 % [+1.96 .. +3.93]	48	4.3, 5.1, 6, A, B
KTG			head pub (public prompts only)	−1.63 % [−2.35 .. −0.91]	48	
B3e6	2026-10-09	FLE-IQ3_M, p49-ring, budget 17,035 MiB (+1.64 %)	FLE-IQ3_M-REAP448, cold	+0.71 % [−0.32 .. +1.75]	24	3, 4.1, 7, A
B3e6			FLE-IQ3_M-REAP448, own warm-start profile	+0.53 % [−0.59 .. +1.73]	24	