

Compression stopped being a research topic and became a *runtime feature*.

One month, read narrowly and on purpose: what landed in llama.cpp, what the small-model compression literature actually established, and which open weights are worth the download. The through-line is that the techniques used to shrink models moved out of papers and into the things we run — and that a cluster of results published in the same weeks say we have been measuring the damage wrong.

COMPILED 16 AUG 2026 6 SECTIONS ~70 PRIMARY CITATIONS SOURCES LINKED THROUGHOUT

AT A GLANCE

01

llama.cpp got NVFP4 end to end. `llama-quantize` can now produce it #26556 and CUDA has repacked-weight plus mixed matrix-vector kernels to consume it #26311. It is a Blackwell format in practice — on Hopper you are still on FP8 and MXFP4 paths.

02

The month's best free speedup is KV cache cloning. A new `clone_to` action on the `/slots` endpoint shares an already-computed prefix across slots instead of re-prefilling it, reported at up to 2.12× on Apple Silicon #26204. If you run several agents over one long system prompt, this is the single change to pull.

03

Suffix decoding landed — speculative decoding with no draft model. An online suffix tree over the context proposes continuations, so you get the speedup without hosting a second model #26283.

04

On-policy distillation is now the consensus way to make a small model good, displacing "prune the big one". Eight substantive papers in four weeks, and it is a stated ingredient in Qwen3, DeepSeek-V4 and MiMo-V2-Flash. The most useful of them argues it mainly buys sampling efficiency, not new capability 2608.11829.

05

KV cache compression is where the research volume went — a dozen serious papers in a month, one claiming $20\times$ at 99% accuracy 2608.02901, and the 2-bit floor no longer holding 2608.07915. For long-context and agent workloads this is higher leverage than weight quantization.

06

Compression is not behaviourally neutral, and your evals will not catch it. Four independent papers this month found compressed models passing standard quality guards while inventing procedure steps, degrading reasoning-chain validity, and stereotyping more in open-ended generation. See §2, last part — it is the finding with the most operational consequence in this issue.

CONTENTS

- §1 llama.cpp — numerics, decoding, silicon
- §2 Small models — compression, pruning, distillation
- §3 Weights that shipped
- §4 Elsewhere — architecture, memory, decoding
- §5 What I would actually do
- §6 Glossary — the underlying technology

Numerics, decoding, silicon

Roughly 100–150 pull requests a week, tagged builds most days, b10451 by 16 August. Reading a month of that as a changelog is useless, so here it is grouped by what it changes for someone actually running the thing.

Numerics: 4-bit became a first-class citizen

NVFP4 is now supported through the whole path rather than as an experiment. You can quantize to it #26556, CUDA consumes it via linear weight repacking with mixed matrix-vector quantization for cache and data reuse #26311, and W4A4 activation quantization gained per-channel scaling #25730. That last one matters: weight-only 4-bit has been routine for years; getting activations down too is what actually moves memory bandwidth.

Below that, a lot of unglamorous kernel work. New matrix-engine GEMM kernels for Q4_0, Q5_K and Q8_0 with static-reuse dequantization and fused bias report $1.04\times$ to $9.5\times$ on prefill depending on shape (#26323, #26326, #26371). Q2_0 reached CUDA #25603 and then got HIP dot-product optimizations for AMD #26753. And q8_0's value range was widened to reduce quantization error while keeping outputs bit-identical #25493 — the kind of change that is invisible until you diff two runs and they match.

Decoding: two changes worth rebuilding for

KV cache cloning. The `/slots` endpoint gained a `clone_to` action that copies a computed cache between slots, so a shared prefix is prefilled once rather than per-request — up to $2.12\times$ reported on Apple Silicon #26204. Any workload that fans several requests out over one long system prompt or one loaded document gets this almost for free.

Suffix decoding. Model-free speculation built on online suffix trees #26283. Classic speculative decoding needs a draft model resident in memory and roughly aligned with the target; a suffix tree over what has already been generated needs neither. It pairs with an adaptive draft-length heuristic for multi-turn work #25726 and with Gemma 4 multi-token-prediction speculation #25679.

Around those: sparse KV indices in MMA flash attention #25917, disaggregated prefill workers so a long prompt stops blocking a decode slot #25675, separate sampling parameters for reasoning and non-reasoning spans #25709, and a `--reasoning-preserve` flag that keeps reasoning content in chat history so a later turn can build on it rather than re-deriving it.

Silicon: the NPU work is getting serious

The Hexagon backend was overhauled into a fully asynchronous design — multi-NPU devices, async graph compute, events, cross-device synchronisation #26501 — on top of an L2 cache rework with dirty-bit tracking and lazy flushing #25762. That is phone-class and laptop-class NPU inference being treated as a real target rather than a demo.

A new backend appeared for Esperanto's ET-SOC-1 #24179 and filled out fast with K-quant support, a GEMV decode rewrite and double-buffered matrix-engine kernels (#26446, #26327, #26328). Elsewhere: OpenCL decode-side K-quant and flash-attention work on Adreno plus Intel Xe-LP tuning worth up to 32% throughput (#26477, #26428, #26438); SYCL GLU kernel consolidation for +14% on Arc B70 #26354; Metal chunked SSD matmul giving Mamba-2 prefill about +12% #26647 plus a `mul_mm_id` NaN fix and thermal-cooldown retuning #26223; Vulkan specialization-constant matmul #25773 and XOR swizzling to kill shared-memory bank conflicts #25635; and an experimental CUTLASS MoE prefill path for SM120 #26704.

Models

Kimi K3's text model landed in b10448 on 15 August — hybrid attention, cross-layer residuals, MXFP4 routed experts, its own chat template. Also this month: Motif 3 beta with GDLA attention, grouped PolyNorm and manifold-constrained hyper-connections #26298; Qwen3-TTS #26254 with a matching streaming `/tts` server endpoint #26603; per-layer sliding-window attention for Qwen3 #26286; BailingMoE 3 and Ling 3.0 flash with speculative decoding #26608; and a fused hyper-connection sinkhorn op that collapses a chunk of the DeepSeek-V4 graph #25585. Nemotron-3-Puzzle, Tencent's Hy3 299B heterogeneous MoE, Qwen3-Next multi-token prediction and Granite 4 vision all arrived or were fixed up in the same window.

The server is quietly becoming a platform

An OpenAI-compatible gateway with authentication, RBAC and audit logging #25662. Outbound MCP client support, so the server can reach external tool servers #25736. Multimodal slot save and restore #26640. Model routing with `--models-dir` discovery of MTP draft models. Error classes that distinguish client-side from server-side context exhaustion #26848. Read those together and llama-server is no longer just a demo harness around the library.

THE REJECTION WORTH KNOWING ABOUT

PR #26794 tried to bring training and compression *into* llama.cpp in one go: QLoRA for MoE models via `MUL_MAT_ID` backward passes, quantization-aware training with straight-through-estimator backward for Q8_0/Q6_K/Q4_0/MXFP4, quantized AdamW variants, critical-token weighted SFT, GRPO-style reward-weighted SFT, gradient checkpointing, and a `llama-prune` tool for calibration-based soft and hard expert pruning of MoE models.

It was closed on 11 August — 215 commits touching CUDA, Metal and Vulkan at once, rejected as "way too many changes" rather than on merit. No benchmark numbers were published with it. Worth watching for resubmission in pieces: a runtime that can do QAT and expert pruning locally, on the same quant formats it serves, would change what "fine-tune a small model" costs for anyone without a cluster.

Rough edges, so you do not find them the hard way

- `GGML_OP_TOP_K` falls back to CPU on HIP/ROCm, costing about 6.4×.
- Vulkan `DeviceLostError` crashes on Strix Halo with DeepSeek-V4-Flash.
- SYCL regressions on Arc Pro B60 with hybrid Qwen3.5-class models; garbled output from Gemma 4 12B on Arc Pro B70 with large prompts.
- Security fixes landed for symlink handling in `/slots?action=save`, the grammar parser, and LoRA tensor bounds (b10451). If you expose llama-server anywhere, update.

§2 · SMALL MODELS

Compression, pruning, distillation

Every arXiv identifier below was published inside the 16 July – 16 August window. The volume is genuinely high; what follows is the part that changes practice.

On-policy distillation ate the field

The old recipe for a good small model was: take a big one, prune it, distil from static teacher outputs. The new one is on-policy distillation — the student generates, and the teacher scores *the student's own* trajectories, so training states match deployment states. It has moved into production pipelines at Qwen3,

DeepSeek-V4 and MiMo-V2-Flash, and reports of matching or beating RL at roughly a tenth of the GPU hours are what pulled everyone in (Thinking Machines, survey 2604.00626).

This month was the consolidation pass. Simple-OPD demystifies the warm-up phase and proposes LoRA initialisation on teacher chain-of-thought 2608.06802. "Mismatch Matters" identifies degenerate token-level agreement and handles student excess and deficit separately 2608.09836. CROP uses counterfactual sensitivity to find which positions are actually worth supervising 2608.13387. I-SDPO routes between self-distillation and reward learning per instance difficulty 2608.12957, and one method drops the teacher artifacts entirely by learning a privileged context end to end 2608.13040.

The paper to read first, though, is the deflationary one: analysed through the lens of test-time scaling, on-policy distillation looks like it primarily improves *sampling efficiency* rather than expanding the reasoning a student can do at all 2608.11829. That is still very valuable — it is most of what you want from a deployed small model — but it sets a ceiling on the claims.

Pruning got more honest about budgets

SNIPER unifies depth and width pruning as a binary knapsack problem, which is the first formulation I have seen that actually hits a stated compression budget precisely instead of approaching it 2608.12953.

Complementing it, two inference-time sparsity results that need no retraining: Prox derives FFN activation sparsity from approximate channel salience for $1.99\times$ decoding at 70% sparsity 2607.27591, and WIDE does token-level dynamic width pruning with kernel co-design for $1.98\times$ prefill and $4.95\times$ decoding 2607.28418.

Two more that are easy to overlook and cheap to apply: corpus-driven vocabulary pruning cutting a 128K vocabulary to 10K for 60% memory savings with no translation quality loss 2608.03480 — the embedding and LM-head tables are a large fraction of a small model — and random-matrix-theory analysis of attention spectra to find genuinely structured components to keep 2608.07921.

Quantization: calibration data turned out to be the lever

The most practically interesting result is ScaleQ-1.58 2608.01078: post-training quantization of reasoning models has been failing not because 1.58-bit is too aggressive but because the calibration set was wrong. Calibrate on the model's own reasoning traces and ternary PTQ becomes viable — 8.97 points absolute on a ternarized Qwen3-4B, off 4M calibration tokens. Same shape of insight, different mechanism, in GaugeQuant, which exploits transformer symmetries to choose a quantization-optimal basis online and takes W4A4 perplexity from 8.22 to 6.73 2607.20757.

Also: ReRound uses a diffusion model to resolve midpoint rounding ambiguity 2608.11045; tied trit-planes constrain ternary decomposition to a uniform nine-level quantizer for MoE 2608.08910; ARCHead compresses the LM head specifically — a quantized low-rank core plus residuals, $3.7\text{--}3.9\times$ 2608.02703; and language-

conditional LoRA corrections recover what quantization steals from multilingual models 2608.11786. On the analysis side, a signal-to-noise decomposition of quantization error by layer 2608.08188 and a margin-shrinkage method that predicts *which specific decisions* a low-bit model will get wrong 2608.06564.

Or: design the small model in from the start

Matryoshka Language Model Suites trains nested sub-models jointly with continuous distillation, so one training run yields the whole size ladder — 36% less compute than training each size independently, and 14–26% better speculative-decoding throughput because the small members are natively aligned drafters for the large ones 2608.09703. That is the most architecturally interesting compression paper of the month.

Alongside it, Opt.Gear ships a hybrid convolutional-attention family spanning 1M to 1B parameters aimed squarely at on-device 2608.01034, and "Wiring Beats Blending" characterises what actually transfers between transformer sizes, finding least-squares compensation beats subcloning at low token budgets 2608.02829.

The KV cache is where the leverage moved

For long-context and agentic work the cache, not the weights, is the memory problem, and the literature has followed. AnchorKV represents the cache as anchors plus residuals for $20\times$ compression at 99% accuracy 2608.02901. SPECTRA applies spectral transforms to decorrelate channels and concentrate bits where they matter, pushing past what had been treated as a hard 2-bit cliff 2608.07915. QEvict makes quantized eviction *recoverable* when future attention reactivates a discarded region 2608.05326.

Then a cluster on allocation rather than representation: RippleKV allocates budget across layers by measuring how perturbations propagate to the output distribution 2608.08684; GraceKV treats it as global resource allocation between local resolution and coverage 2608.07001; DistillCache learns an eviction policy by RL against a KL reward 2608.08878. Two are aimed directly at agents: CommitKV uses agent event boundaries to tell dormant states from completed ones 2608.07855, and cross-model KV transfer maps a cache between models in a family by closed-form ridge regression so switching models does not mean re-prefilling 2608.03893. RotaryQuant is the blunt-practical one: mixed-precision plus isotropising cache transforms to fit a 120B MoE on consumer hardware 2608.08081.

READ THIS PART TWICE

Four independent papers this month, using different methods, reached the same uncomfortable conclusion: **a compressed model can pass every check you currently run and still be behaviourally different in ways that matter.**

"Fidelity Is Not Safety" finds gently low-rank-compressed models clearing standard quality guards while inventing procedure steps during agentic execution 2607.28196. "Does Accuracy Equal Evidence?" finds KV token-eviction preserving final-answer accuracy while degrading the validity of the reasoning chain that produced it 2608.01631. QuantiBias finds quantized models passing standard safety checks yet stereotyping measurably more in open-ended generation, across eight languages 2607.21063. And a direct benchmark of pre-trained versus compressed SLMs finds quantization preserves trustworthiness better than pruning or distillation do 2608.11981.

Two adjacent results sharpen it: sequential memory edits catastrophically degrade 4-bit models unless explicitly stabilised 2607.28292, and 4-bit quantization substantially damages the internal confidence signals of vision-language models even where task accuracy holds 2607.24440.

The converging practical rule, restated across all of them and consistent with the earlier pruning-versus-quantization literature: **quantize rather than prune; pruning damages reasoning specifically; and validate on agentic traces, not on perplexity and multiple choice.**

§3 · RELEASES

Weights that shipped

Open weights only, and only what is relevant to running things yourself. Dates are announcement dates; several of these staged the actual download separately, which is noted.

DATE	MODEL	WHY IT MATTERS
10 Aug	Meta Muse Glimmer	Meta back in open weights. 30B dense (~29.6B across 52 layers) plus a 1.8B ViT-G/14 encoder, 131K+ context, Apache 2.0, sized for one consumer GPU, aimed explicitly at always-on local agents. Day-zero integrations for llama.cpp, MLX and ExecuTorch — the release most likely to change what you run locally.

DATE	MODEL	WHY IT MATTERS
14 Aug	Qwen3.8-27B	The locally-runnable half of the Qwen3.8 release. 262K native context, ships with a 27-layer vision encoder and its own image/video preprocessing.
3–12 Aug	Qwen3.8-Max	2.4T-parameter MoE, 95B active, 1M context, native text/image/video, Apache 2.0 — the first Max-tier Qwen ever made downloadable. Announced 3 Aug, weights on Hugging Face and ModelScope around 12 Aug; reporting on the staging was inconsistent, so check the repo rather than the coverage.
Jul	Kimi K3	2.78T total, 104B active. Kimi Delta Attention (linear attention with channel-wise decay control), attention residuals that let a layer pull from arbitrary earlier layers, and Stable LatentMoE routing 896 experts down a 3,584-dim latent before dispatch with 16 active plus 2 shared. Claimed $\sim 2.5\times$ the scaling efficiency of K2. llama.cpp support landed 15 Aug.
14 Aug	GLM-5.3	743B MoE, $\sim 40B$ active — the <i>same base</i> as GLM-5.2, with every reported gain attributed to expanded post-training. 66.9% on DeepSWE, twenty points above 5.2. MIT weights, but staged roughly two weeks after launch.
4 & 13 Aug	Liquid LFM2.5	LFM2.5-2.6B (34T training tokens, built for on-device agent loops) and LFM2.5-VL-3B, a 3.1B vision-language model that reads screens, grounds objects and calls tools locally. The serious edge-tier option this month.

Two pieces of out-of-window context that everything above is reacting to. NVIDIA's Nemotron-3-Puzzle-75B-A9B (6 July, just before this window) is a compressed variant of a 120B-class model produced by an "Iterative Puzzle" post-training compression framework — compression-at-scale as a shipping product, not a paper. And DeepSeek-V4 (April) remains the architecture the inference stacks are chasing: interleaved Compressed Sparse Attention and Heavily Compressed Attention bringing 1M-token inference down to 27% of V3.2's FLOPs and 10% of its KV cache. Much of the Vulkan and CUDA work in [§1](#) exists to make that architecture run on hardware you own.

§4 · ELSEWHERE

Architecture, memory, decoding

Post-training is now a scaling axis in its own right

GLM-5.3 is the cleanest evidence available: identical 743B base as its predecessor, twenty points of DeepSWE improvement, all of it attributed to post-training. When the base is held fixed and the gain is that

large, the interesting budget question stops being "how much pre-training compute" and becomes "how much post-training, and of what kind".

Hybrid and linear attention went mainstream

Kimi K3 shipping KDA at 2.78T parameters is the headline, but the supporting literature arrived the same month. MixFormer combines multiple memory experts with time-aware linear attention [2608.09468](#); a block-hybrid scheme retrofits linear attention into diffusion language models, keeping exact softmax within a block and going linear across previous blocks [2608.06628](#); MARCH scales recurrent memory with content-routed state anchors [2608.12435](#); and an empirical study characterises the massive-activation patterns these hybrids produce, which is exactly the knowledge quantization needs to not fall over on them [2608.12149](#).

Memory and continual learning

Macaron-V1 attacks open continual learning with self-improvement and a mixture of LoRAs [2608.09819](#); BDH-CQ explores in-context learning with recurrent latent reasoning [2608.09888](#); Consolidator learns to move short-term into long-term memory across context boundaries [2608.11701](#). Still early — but this is the direction that would eventually make "the model does not remember last week" stop being an architecture problem you route around with retrieval.

Diffusion language models grew a decoding stack

The interesting work stopped being "can diffusion do language" and became "how fast can it decode". DARTree extends autoregressive correction to draft trees, training-free [2608.13524](#); Ripple-Pivot exploits the ripple effect by committing mid-entropy positions to parallelise decoding [2608.11742](#); LibraSpec reframes speculation length as expected-speedup optimisation [2608.08721](#); Archer reuses cached hidden states asymmetrically for a $2.57\times$ mean speedup [2608.08086](#).

Agent infrastructure became a research area

Serving and training systems for agents are now producing real systems papers: TideRL schedules agentic RL by readiness with continuous task batching [2608.10402](#); MISA-T schedules mixed RL rollouts beyond prefix locality with adaptive KV allocation [2608.11152](#); CRISP trains search agents by distinguishing necessary evidence-gathering from redundant turns, cutting turns 15–33% [2608.01867](#); HetRoute does cost-aware collaborative routing for distributed edge MoE inference, reporting 59% lower latency and 72% less traffic [2608.00577](#). On the adversarial side, OpenART scales agent red-teaming through open-ended environment evolution [2608.00677](#).

What I would actually do

Sorted by how ready each thing is, not by how interesting it is.

ADOPT

Rebuild llama.cpp and turn on cache cloning. If a workload fans multiple requests over a shared system prompt or a loaded document — which describes most agent setups — `/slots clone_to` is close to free throughput. Pair it with suffix decoding, which needs no draft model to host, and with disaggregated prefill if long prompts are starving decode slots.

ADOPT

Quantize; do not prune. Two independent lines of evidence this month, on top of the existing literature, say quantization preserves reasoning and trustworthiness better than pruning at equivalent compression. If you need more than quantization gives you, the next move is on-policy distillation into a smaller model — not structured pruning of the larger one.

ADOPT

Change how compressed models are validated. Perplexity and multiple-choice benchmarks demonstrably miss the failure modes: invented procedure steps under agentic execution, reasoning chains that no longer support their own correct answers, increased stereotyping in open generation. A compressed model should be re-run against real task traces with the chain inspected, not just the final answer scored.

WATCH

KV cache compression. The results are strong and the volume is high, but nearly all of it is a month old and none of it has landed in a mainstream runtime yet. The allocation-based methods (RippleKV, GraceKV) and the agent-aware ones (CommitKV) are the ones I would expect to reach a serving stack first. Revisit next issue.

WATCH

llama.cpp PR #26794, if it comes back in pieces. Local QAT and expert pruning on the same quant formats the runtime already serves would change what fine-tuning a small model costs anyone without a cluster. It was rejected on size, not on merit.

Ternary and 1.58-bit post-training quantization, even with the ScaleQ-1.58 calibration insight, and **diffusion-LLM decoding**. Both are genuinely moving; neither has a deployment story worth relying on this quarter.

NOTE

NVFP4 is a Blackwell format. The llama.cpp NVFP4 path assumes hardware support that Hopper does not have — on H100s, FP8 and MXFP4 are the precisions that apply, not NVFP4. Worth checking before quantizing a model for the wrong target.

NOTE

Local model shortlist, if you are refreshing one: Muse Glimmer 30B for agentic work on one GPU, Qwen3.8-27B where vision and long context matter, LFM2.5-2.6B or VL-3B for genuinely on-device. All Apache 2.0.

The underlying technology

The substrate the rest of this issue assumes. Grouped by area and ordered so each term builds on the ones above it, rather than alphabetically — it is meant to be readable start to finish, not only looked up.

How a model serves a request

Prefill / decode

The two phases of generation. *Prefill* processes the entire prompt in one parallel pass and is limited by raw compute. *Decode* emits one token at a time, each depending on the last, and is limited by memory bandwidth. They have opposite bottlenecks, which is why every speedup in this issue is quoted for one or the other — a 9.5× prefill gain may do nothing for decode.

KV cache

Each generated token attends to every token before it. Rather than recompute the key and value vectors of all those earlier tokens at every step, they are stored. The cache grows linearly with context length, and at long contexts it routinely exceeds the size of the weights themselves — which is why compressing it ([§2](#)) beats compressing weights for long-context work.

Prefix caching

When many requests begin identically — the same system prompt, the same loaded document — the KV cache for that shared opening can be computed once and reused instead of re-prefilled per request. llama.cpp's cache cloning ([§1](#)) is this idea exposed as a server endpoint.

Context window

The maximum number of tokens the model can attend over at once. The headline numbers in [§3](#) (262K, 1M) are what the architecture supports, not necessarily what it uses well or what fits in your memory.

Attention, and why there are so many kinds

Attention(full, dense, softmax)

The core operation: every token compares itself against every other token to decide what to draw on. Exact, and quadratic — double the sequence, quadruple the cost. Everything below is an attempt to avoid paying that.

Flash attention

Not an approximation. The same exact maths, computed in tiles so the full attention matrix is never written to memory. Pure win, which is why it is everywhere.

MHA / GQA / MLA

Multi-head attention runs several attention operations in parallel. *Grouped-query* has heads share key/value vectors, shrinking the cache. *Multi-head latent* compresses keys and values into a low-rank latent space, shrinking it further. Mostly these are cache-size reductions, not compute reductions.

Sliding-windowattention (SWA)

Each token attends only to the last N tokens. The cache stops growing and becomes fixed-size — at the cost of the model genuinely not being able to see further back from that layer.

Sparse attention

Attend to a chosen subset of earlier tokens rather than all of them, with the selection either structural or learned. DeepSeek-V4's Compressed Sparse Attention consolidates the cache into blocks and then picks the top-k blocks, which is how it reaches a million tokens at 27% of the previous generation's compute.

Linear attention

Reformulates attention to cost linearly rather than quadratically, by carrying a fixed-size recurrent state instead of an ever-growing cache. Cheap and constant-memory — but the state is lossy, so it forgets. The whole design question is what it forgets and how fast.

Hybrid attention

Mixing two or more of the above inside one model, usually by interleaving layers. The trade is direct: linear and sliding-window layers are cheap but lossy, full and sparse layers are exact but expensive. Interleaving them captures most of the cost saving while keeping enough exact layers for genuine long-range recall. This is *the* dominant architecture of 2026 — Kimi K3 pairs KDA with MLA, DeepSeek-V4 interleaves compressed sparse with heavily compressed attention, Qwen3 and Granite interleave sliding-window layers with full ones. When this issue says "hybrid", that is what it means.

KDA

Kimi Delta Attention, K3's linear variant. Its distinguishing feature is per-channel control over how fast each dimension of the recurrent state decays — finer-grained forgetting than a single global decay rate.

RoPE

Rotary position embeddings: how a token's position gets encoded, by rotating its query and key vectors by an angle proportional to position. The standard approach, and the thing that extending a context window usually has to stretch.

Model structure

Dense vs MoE

A *dense* model runs every parameter for every token. A *mixture of experts* holds many parallel feed-forward "experts" and a router selects a few per token, so most of the model sits idle on any given token.

Total vs active parameters

Why MoE models are quoted as "2.4T total, 95B active". *Total* determines how much memory you need to hold the model; *active* determines how much compute each token costs. This is the entire reason trillion-parameter models are servable, and the reason they are still enormous to host.

Router /shared experts

The router is the small network choosing which experts handle each token — training it to distribute load evenly is a persistent difficulty. *Shared* experts run for every token regardless, holding general capability so the routed experts can specialise.

MTP

Multi-token prediction: training the model to predict several tokens ahead rather than one. The extra prediction heads then double as a built-in drafter for speculative decoding, which is why MTP support and speculation support keep appearing together in [§1](#).

Residual and hyper-connections

Residual connections are the skip paths that let signal and gradient flow through a deep stack unimpeded. Recent work generalises them — manifold-constrained hyper-connections, attention residuals that let a layer pull representations from any earlier layer — because in deep sparse models a plain skip path is a bottleneck.

Making a model smaller

Quantization

Storing weights, and sometimes activations, at lower numeric precision. The dominant compression technique, and per [§2](#) the one that damages model behaviour least.

PTQ vs QAT

Post-training quantization converts a finished model — cheap, minutes to hours, no training. *Quantization-aware training* simulates the low precision during training so the model adapts to it — expensive, but the only thing that works at genuinely aggressive bit widths.

Calibration

PTQ needs a sample of data to choose its scaling factors. Long treated as a detail; [§2](#) is largely the story of that assumption breaking — calibrating on the right data is what made ternary quantization of reasoning models work at all.

Bit formats

FP16 and BF16 are the 16-bit baseline; FP8 and INT8 halve it; 4-bit halves it again. *Q4_K* and friends are llama.cpp's own block-quantized formats. *MXFP4* and *NVFP4* are 4-bit microscaling formats where a small block of values shares one scale factor — NVFP4 uses smaller blocks, capturing local dynamic range better, and needs Blackwell-generation hardware to run natively.

Weight-only vs W4A4

Weight-only stores weights at 4 bits but converts back up to compute, so it saves memory but not much bandwidth. *W4A4* quantizes the activations too, so the arithmetic itself runs at 4 bits. That is the harder problem and the one that actually unlocks low-precision matrix engines.

STE

Straight-through estimator: the trick that makes QAT trainable at all. Rounding has a gradient of zero everywhere, so the backward pass simply pretends the rounding step was the identity function and passes the gradient through unchanged.

Pruning

Deleting parts of the model outright. *Unstructured* pruning zeroes individual weights, producing sparsity that is hard for hardware to exploit. *Structured* pruning removes whole units — heads, layers, experts, vocabulary entries — which genuinely shrinks the model. Per [§2](#), it also damages reasoning more than quantization does.

Distillation

Training a small *student* model to reproduce the behaviour of a large *teacher*. Distinct from compression proper: you are training a new small model, not shrinking the big one.

Off-policy vs on-policy distillation

Off-policy trains the student on text the teacher produced. *On-policy* has the student generate, and the teacher grade the student's own attempts — so the states the student trains on are the states it will actually encounter when deployed. That mismatch is what off-policy distillation gets wrong, and closing it is the reason on-policy took over the field this year.

LoRA / QLoRA

Fine-tuning by training small low-rank adapter matrices alongside frozen weights, instead of updating the whole model. *QLoRA* does that on top of an already-quantized base, which is what makes fine-tuning large models on one GPU possible.

Making inference faster

Speculative decoding

Something cheap proposes the next several tokens; the real model then verifies all of them in a single parallel pass and keeps the ones it agrees with. Because verification is parallel and generation is serial, this converts memory-bound decode steps into compute-bound ones. Critically, the output distribution is unchanged — it is a speedup, not an approximation.

Draft model

The small model doing the proposing. It must be resident in memory and roughly aligned with the target model, which is the main cost of the technique.

Suffix decoding

Speculation with no draft model at all: an index over text already seen proposes continuations, which works well because real generation repeats itself — code, names, quoted context. Removes the draft model's memory cost entirely (§1).

Disaggregated prefill

Running the prefill phase on separate workers so that one long prompt does not stall token generation for everyone else sharing the server.

Training stages

Pre- vs post-training

Pre-training is the large unsupervised run that builds the base model. *Post-training* is everything after — supervised fine-tuning, reinforcement learning, distillation. GLM-5.3 moving twenty points on the same base (§4) is why post-training is now treated as a scaling axis of its own.

SFT

Supervised fine-tuning: training on demonstrations of the desired behaviour. The simplest and still the backbone of post-training.

RL / RLVR / GRPO

Reinforcement learning optimises against a reward rather than against fixed targets. *RLVR* uses rewards that can be checked automatically — the tests pass, the answer is right — which is why it works so well for code and maths. *GRPO* is the group-relative policy optimisation algorithm that became the standard way to run it.

Sources

LLAMA.CPP

[llama.cpp releases](#) — build tags and per-build notes (b10444–b10451 cited).

[Weekly GitHub report, 7–14 Jul 2026](#)

[Weekly GitHub report, 13–20 Jul 2026](#)

[Weekly GitHub report, 27 Jul–3 Aug 2026](#)

Weekly GitHub report, 4–11 Aug 2026

PR #26794 — MoE training, QAT and `llama-prune` ; closed 11 Aug 2026.

Individual PRs are linked inline throughout §1.

COMPRESSION, PRUNING, DISTILLATION

All arXiv identifiers in §2 and §4 link to `arxiv.org/abs/` and were retrieved via the arXiv API filtered to 16 Jul – 16 Aug 2026.

Thinking Machines Lab — On-Policy Distillation

A Survey of On-Policy Distillation for Large Language Models

UniComp: unified evaluation of compression via pruning, quantization and distillation

Revisiting Pruning vs Quantization for Small Language Models — the prior result the new work keeps confirming.

Survey on system-aware KV cache optimization

RELEASES

Kimi K3 technical blog

CNBC — Meta launches Muse Glimmer and InfoQ — architecture detail

Qwen3.8-Max specs and Qwen3.8-27B specs and release date

Liquid AI LFM2.5-VL-3B

GLM-5.3 architecture and benchmarks

DeepSeek-V4 — CSA and HCA (April 2026, out of window, cited as context)

Compiled 16 August 2026 from llama.cpp's release feed and weekly GitHub reports, the arXiv API filtered to this window, and release coverage. Figures quoted as speedups or compression ratios are the authors' own reported numbers on their own benchmarks — treat them as claims, not measurements, until we reproduce them. Anything dated outside 16 July – 16 August 2026 is labelled inline as context.



MONTEVIVE.AI

LLM Radar is published weekly by Montevive. All issues · ai@montevive.ai