

La compresión dejó de ser un tema de investigación y pasó a ser una *función del runtime*.

Un mes, leído en corto y a propósito: qué entró en llama.cpp, qué estableció realmente la literatura de compresión de modelos pequeños y qué pesos abiertos merecen la descarga. El hilo conductor es que las técnicas para encoger modelos salieron de los papers y entraron en las cosas que ejecutamos — y que un grupo de resultados publicados esas mismas semanas dice que hemos estado midiendo el daño de forma equivocada.

COMPILADO EL 16 AGO 2026 6 SECCIONES ~70 CITAS PRIMARIAS
FUENTES ENLAZADAS EN TODO EL TEXTO

DE UN VISTAZO

01

llama.cpp tiene NVFP4 de principio a fin. llama-quantize ya puede producirlo #26556 y CUDA tiene kernels de repacking lineal de pesos y matriz-vector mixto para consumirlo #26311. En la práctica es un formato Blackwell: en Hopper sigues con FP8 y MXFP4.

02

La mejor aceleración gratis del mes es el clonado de KV cache. Una nueva acción clone_to en el endpoint /slots comparte un prefijo ya calculado entre slots en vez de volver a hacer prefill, con hasta 2,12× reportado en Apple Silicon #26204. Si lanzas varios agentes sobre un mismo system prompt largo, este es el cambio que hay que traerse.

03

Llegó el suffix decoding: speculative decoding sin draft model. Un suffix tree online sobre el contexto propone continuaciones, así que consigues la aceleración sin alojar un segundo modelo #26283.

04

La on-policy distillation es ya la forma consensuada de hacer bueno un modelo pequeño, desplazando al "poda el grande". Ocho papers sustanciales en cuatro semanas, y es un ingrediente declarado en Qwen3, DeepSeek-V4 y MiMo-V2-Flash. El más útil de todos sostiene que lo que compra sobre todo es eficiencia de muestreo, no capacidad nueva 2608.11829.

05

La compresión de KV cache es donde se fue el volumen de investigación — una docena de papers serios en un mes, uno reclamando $20\times$ con 99% de precisión 2608.02901, y el suelo de 2 bits ya no se sostiene 2608.07915. Para cargas de contexto largo y agentes esto tiene más palanca que cuantizar pesos.

06

La compresión no es neutral en comportamiento, y tus evaluaciones no lo van a detectar. Cuatro papers independientes encontraron este mes modelos comprimidos que pasan los controles de calidad habituales mientras se inventan pasos de procedimiento, degradan la validez de la cadena de razonamiento y estereotipan más en generación abierta. Ver [§2](#), última parte — es el hallazgo con más consecuencias operativas de este número.

CONTENIDO

- §1 llama.cpp — numéricos, decoding, silicio
 - §2 Modelos pequeños — compresión, pruning, destilación
 - §3 Los pesos que se publicaron
 - §4 En otros frentes — arquitectura, memoria, decoding
 - §5 Qué haría yo de verdad
 - §6 Glosario — la tecnología de base
-

Numéricos, decoding, silicio

Unos 100–150 pull requests por semana, builds etiquetadas casi a diario, b10451 a 16 de agosto. Leer un mes de eso como changelog no sirve de nada, así que aquí va agrupado por lo que cambia para quien realmente ejecuta esto.

Numéricos: 4 bits pasó a ser ciudadano de primera

NVFP4 ya está soportado en todo el camino, no como experimento. Puedes cuantizar a él #26556, CUDA lo consume mediante repacking lineal de pesos con cuantización matriz-vector mixta para aprovechar caché y reutilización de datos #26311, y la cuantización de activaciones W4A4 ganó escalado por canal #25730. Esto último importa: los 4 bits solo en pesos llevan años siendo rutina; bajar también las activaciones es lo que de verdad mueve el ancho de banda de memoria.

Por debajo, mucho trabajo de kernel poco vistoso. Nuevos kernels GEMM de matrix engine para Q4_0, Q5_K y Q8_0 con dequantización de reutilización estática y bias fusionado reportan de $1,04\times$ a $9,5\times$ en prefill según la forma (#26323, #26326, #26371). Q2_0 llegó a CUDA #25603 y después recibió optimizaciones de producto escalar en HIP para AMD #26753. Y se amplió el rango de valores de q8_0 para reducir el error de cuantización manteniendo salidas idénticas bit a bit #25493 — ese tipo de cambio que es invisible hasta que comparas dos ejecuciones y coinciden.

Decoding: dos cambios que justifican recompilar

Clonado de KV cache. El endpoint `/slots` ganó una acción `clone_to` que copia una caché ya calculada entre slots, de modo que un prefijo compartido se hace prefill una vez y no una por petición — hasta $2,12\times$ reportado en Apple Silicon #26204. Cualquier carga que abra varias peticiones sobre un mismo system prompt largo o un documento cargado se lleva esto casi gratis.

Suffix decoding. Especulación sin modelo, construida sobre suffix trees online #26283. El speculative decoding clásico necesita un draft model residente en memoria y más o menos alineado con el modelo objetivo; un suffix tree sobre lo ya generado no necesita ninguna de las dos cosas. Se combina con una heurística de longitud de draft adaptativa para trabajo multi-turno #25726 y con la especulación por multi-token prediction de Gemma 4 #25679.

Alrededor de eso: índices KV dispersos en flash attention MMA #25917, workers de prefill desagregados para que un prompt largo deje de bloquear un slot de decode #25675, parámetros de sampling separados para los tramos de razonamiento y los que no #25709, y un flag `--reasoning-preserve` que mantiene el contenido de

razonamiento en el historial para que un turno posterior pueda construir sobre él en vez de volver a derivarlo.

Silicio: el trabajo en NPU se está poniendo serio

El backend Hexagon se reescribió a un diseño completamente asíncrono — dispositivos multi-NPU, graph compute asíncrono, eventos, sincronización entre dispositivos #26501 — sobre una reforma de la caché L2 con seguimiento de bits sucios y flush perezoso #25762. Eso es inferencia en NPU de móvil y portátil tratada como objetivo real y no como demo.

Apareció un backend nuevo para el ET-SOC-1 de Esperanto #24179 y se completó rápido con soporte de K-quants, reescritura del decode GEMV y kernels de matrix engine con doble buffer (#26446, #26327, #26328). Además: trabajo de K-quants y flash attention en el lado de decode sobre Adreno en OpenCL más ajuste para Intel Xe-LP que vale hasta un 32% de throughput (#26477, #26428, #26438); consolidación de kernels GLU en SYCL para un +14% en Arc B70 #26354; matmul SSD por chunks en Metal que da un +12% aproximado en el prefill de Mamba-2 #26647 más un arreglo de NaN en `mul_mm_id` y reajuste del enfriamiento térmico #26223; matmul con constantes de especialización en Vulkan #25773 y swizzling XOR para eliminar conflictos de banco en memoria compartida #25635; y una ruta experimental de prefill MoE con CUTLASS para SM120 #26704.

Modelos

El modelo de texto de Kimi K3 entró en b10448 el 15 de agosto — atención híbrida, residuales entre capas, expertos enrutados en MXFP4 y su propia plantilla de chat. También este mes: Motif 3 beta con atención GDLA, PolyNorm agrupado e hiperconexiones con restricción de variedad #26298; Qwen3-TTS #26254 con su endpoint `/tts` de streaming en el servidor #26603; sliding window attention por capa para Qwen3 #26286; BailingMoE 3 y Ling 3.0 flash con speculative decoding #26608; y una operación sinkhorn de hiperconexión fusionada que colapsa buena parte del grafo de DeepSeek-V4 #25585. Nemotron-3-Puzzle, el MoE heterogéneo Hy3 de 299B de Tencent, la multi-token prediction de Qwen3-Next y la visión de Granite 4 llegaron o se arreglaron en la misma ventana.

El servidor se está convirtiendo en plataforma sin hacer ruido

Un gateway compatible con OpenAI con autenticación, RBAC y registro de auditoría #25662. Soporte de cliente MCP saliente, para que el servidor pueda llegar a servidores de herramientas externos #25736. Guardado y restauración de slots multimodales #26640. Enrutado de modelos con descubrimiento de draft models MTP vía `--models-dir`. Clases de error que distinguen el agotamiento de contexto del cliente del servidor #26848. Léelo todo junto y llama-server ya no es un arnés de demostración alrededor de la librería.

El PR #26794 intentó meter entrenamiento y compresión *dentro* de llama.cpp de una sola vez: QLoRA para modelos MoE mediante pases hacia atrás de `MUL_MAT_ID`, quantization-aware training con backward de straight-through estimator para Q8_0/Q6_K/Q4_0/MXFP4, variantes cuantizadas de AdamW, SFT ponderado por tokens críticos, SFT ponderado por recompensa al estilo GRPO, gradient checkpointing y una herramienta `llama-prune` para pruning blando y duro de expertos en MoE basado en calibración.

Se cerró el 11 de agosto — 215 commits tocando CUDA, Metal y Vulkan a la vez, rechazado por ser "demasiados cambios", no por el fondo. No se publicaron cifras de benchmark. Conviene vigilarlo por si vuelve troceado: un runtime capaz de hacer QAT y pruning de expertos en local, sobre los mismos formatos de cuantización que ya sirve, cambiaría lo que cuesta hacer fine-tuning de un modelo pequeño a cualquiera que no tenga un clúster.

Aristas, para que no las descubras por las malas

- `GGML_OP_TOP_K` cae a CPU en HIP/ROCm, con un coste de unos 6,4×.
- `DeviceLostError` de Vulkan provoca cuelgues en Strix Halo con DeepSeek-V4-Flash.
- Regresiones de SYCL en Arc Pro B60 con modelos híbridos de la clase Qwen3.5; salida corrupta de Gemma 4 12B en Arc Pro B70 con prompts largos.
- Entraron arreglos de seguridad para el manejo de symlinks en `/slots?action=save`, el parser de gramáticas y los límites de tensores LoRA (`b10451`). Si expones llama-server en algún sitio, actualiza.

Compresión, pruning, destilación

Todos los identificadores de arXiv de abajo se publicaron dentro de la ventana del 16 de julio al 16 de agosto. El volumen es realmente alto; lo que sigue es la parte que cambia la práctica.

La on-policy distillation se comió el campo

La receta antigua para un buen modelo pequeño era: coge uno grande, pódalo y destila a partir de salidas estáticas del profesor. La nueva es la on-policy distillation — el alumno genera y el profesor puntúa *las*

trayectorias del propio alumno, de modo que los estados de entrenamiento coinciden con los de despliegue. Ha entrado en pipelines de producción en Qwen3, DeepSeek-V4 y MiMo-V2-Flash, y los informes de igualar o superar a RL con aproximadamente una décima parte de las horas de GPU son lo que arrastró a todo el mundo (Thinking Machines, survey 2604.00626).

Este mes fue el de consolidación. Simple-OPD desmitifica la fase de calentamiento y propone inicialización LoRA sobre el chain-of-thought del profesor 2608.06802. "Mismatch Matters" identifica el acuerdo degenerado a nivel de token y trata por separado el exceso y el déficit del alumno 2608.09836. CROP usa sensibilidad contrafactual para encontrar qué posiciones merece la pena supervisar 2608.13387. I-SDPO enruta entre auto-distilación y aprendizaje por recompensa según la dificultad de cada instancia 2608.12957, y un método prescinde por completo de los artefactos del profesor aprendiendo un contexto privilegiado de extremo a extremo 2608.13040.

El paper que hay que leer primero, sin embargo, es el desinflador: analizada desde la óptica del test-time scaling, la on-policy distillation parece mejorar sobre todo la *eficiencia de muestreo* más que ampliar el razonamiento que un alumno es capaz de hacer 2608.11829. Sigue siendo muy valioso — es casi todo lo que quieres de un modelo pequeño desplegado — pero pone techo a las afirmaciones.

El pruning se volvió más honesto con los presupuestos

SNIPER unifica el pruning en profundidad y en anchura como un problema de mochila binaria, que es la primera formulación que he visto que de verdad alcanza con precisión un presupuesto de compresión declarado en vez de aproximarse 2608.12953. Como complemento, dos resultados de esparsidad en tiempo de inferencia que no necesitan reentrenamiento: Prox deriva esparsidad de activaciones en el FFN a partir de la saliencia aproximada de canales para $1,99\times$ en decoding con un 70% de esparsidad 2607.27591, y WIDE hace pruning dinámico de anchura a nivel de token con co-diseño de kernels para $1,98\times$ en prefill y $4,95\times$ en decoding 2607.28418.

Dos más fáciles de pasar por alto y baratos de aplicar: pruning de vocabulario dirigido por corpus que recorta un vocabulario de 128K a 10K con un 60% de ahorro de memoria y sin pérdida de calidad en traducción 2608.03480 — las tablas de embeddings y de la LM head son una fracción grande de un modelo pequeño — y análisis con teoría de matrices aleatorias sobre los espectros de atención para encontrar los componentes realmente estructurados que conviene conservar 2608.07921.

Cuantización: la palanca resultó ser el dato de calibración

El resultado más interesante en la práctica es ScaleQ-1.58 2608.01078: la cuantización post-entrenamiento de modelos de razonamiento venía fallando no porque 1,58 bits sea demasiado agresivo, sino porque el conjunto de calibración era el equivocado. Calibra sobre las propias trazas de razonamiento del modelo y la PTQ ternaria pasa a ser viable — 8,97 puntos absolutos en un Qwen3-4B ternarizado, con 4M de tokens de calibración. La misma forma de intuición, con otro mecanismo, en GaugeQuant, que explota simetrías del

transformer para elegir online una base óptima de cuantización y lleva la perplejidad W4A4 de 8,22 a 6,73 2607.20757.

Además: ReRound usa un modelo de difusión para resolver la ambigüedad de redondeo en el punto medio 2608.11045; los tied trit-planes restringen la descomposición ternaria a un cuantizador uniforme de nueve niveles para MoE 2608.08910; ARCHead comprime específicamente la LM head — un núcleo cuantizado de bajo rango más residuales, $3,7-3,9 \times$ 2608.02703; y correcciones LoRA condicionadas por idioma recuperan lo que la cuantización le roba a los modelos multilingües 2608.11786. En el lado del análisis, una descomposición señal-ruido del error de cuantización por capa 2608.08188 y un método de contracción de márgenes que predice *qué decisiones concretas* se le van a torcer a un modelo de pocos bits 2608.06564.

O bien: diseña el modelo pequeño desde el principio

Matryoshka Language Model Suites entrena submodelos anidados de forma conjunta con destilación continua, así que una sola tirada de entrenamiento produce toda la escalera de tamaños — un 36% menos de cómputo que entrenar cada tamaño por separado, y un 14–26% más de throughput en speculative decoding porque los miembros pequeños son drafters nativamente alineados con los grandes 2608.09703. Es el paper de compresión arquitectónicamente más interesante del mes.

Junto a él, Opt.Gear publica una familia híbrida convolución-atención de 1M a 1B de parámetros apuntando directamente a on-device 2608.01034, y "Wiring Beats Blending" caracteriza qué transfiere realmente entre tamaños de transformer, encontrando que la compensación por mínimos cuadrados supera al subcloning con presupuestos bajos de tokens 2608.02829.

La KV cache es donde se movió la palanca

Para trabajo de contexto largo y agentes, el problema de memoria es la caché, no los pesos, y la literatura ha ido detrás. AnchorKV representa la caché como anclas más residuales para $20\times$ de compresión con 99% de precisión 2608.02901. SPECTRA aplica transformadas espectrales para descorrelacionar canales y concentrar bits donde importan, superando lo que se trataba como un límite duro de 2 bits 2608.07915. QEvict hace que el desalojo cuantizado sea *recuperable* cuando la atención futura reactiva una región descartada 2608.05326.

Después, un grupo sobre asignación más que sobre representación: RippleKV reparte presupuesto entre capas midiendo cómo se propagan las perturbaciones a la distribución de salida 2608.08684; GraceKV lo trata como asignación global de recursos entre resolución local y cobertura 2608.07001; DistillCache aprende una política de desalojo por RL contra una recompensa KL 2608.08878. Dos apuntan directamente a agentes: CommitKV usa las fronteras de eventos del agente para distinguir estados latentes de estados completados 2608.07855, y la transferencia de KV entre modelos mapea una caché entre modelos de una misma familia mediante regresión ridge en forma cerrada, de modo que cambiar de modelo no obligue a rehacer el prefill 2608.03893. RotaryQuant es el práctico y directo: precisión mixta más transformadas isotropizantes de la caché para meter un MoE de 120B en hardware de consumo 2608.08081.

Cuatro papers independientes este mes, con métodos distintos, llegaron a la misma conclusión incómoda: **un modelo comprimido puede pasar todas las comprobaciones que haces hoy y aun así comportarse de forma distinta en aspectos que importan.**

"Fidelity Is Not Safety" encuentra modelos comprimidos suavemente por bajo rango que superan los controles de calidad habituales mientras se inventan pasos de procedimiento durante la ejecución agéntica 2607.28196. "Does Accuracy Equal Evidence?" encuentra que el desalojo de tokens de la KV cache preserva la precisión de la respuesta final mientras degrada la validez de la cadena de razonamiento que la produjo 2608.01631. QuantiBias encuentra modelos cuantizados que pasan los controles de seguridad estándar y sin embargo estereotipan de forma medible más en generación abierta, en ocho idiomas 2607.21063. Y un benchmark directo de SLMs preentrenados frente a comprimidos encuentra que la cuantización preserva mejor la fiabilidad que el pruning o la destilación 2608.11981.

Dos resultados adyacentes lo afinan: las ediciones secuenciales de memoria degradan catastróficamente los modelos de 4 bits salvo que se estabilicen de forma explícita 2607.28292, y la cuantización a 4 bits daña de forma sustancial las señales internas de confianza de los modelos visión-lenguaje incluso donde la precisión en tarea aguanta 2607.24440.

La regla práctica convergente, repetida en todos ellos y coherente con la literatura previa de pruning frente a cuantización: **cuantiza en lugar de podar; el pruning daña específicamente el razonamiento; y valida sobre trazas agénticas, no sobre perplejidad y test de opción múltiple.**

§3 · PUBLICACIONES

Los pesos que se publicaron

Solo pesos abiertos, y solo lo relevante para ejecutar cosas por tu cuenta. Las fechas son de anuncio; varios de estos escalonaron la descarga real, y se indica.

FECHA	MODELO	POR QUÉ IMPORTA
10 ago	Meta Muse Glimmer	Meta vuelve a los pesos abiertos. 30B denso (~29,6B en 52 capas) más un encoder ViT-G/14 de 1,8B, contexto de 131K+, Apache 2.0, dimensionado para una sola GPU de consumo y orientado explícitamente a agentes locales siempre activos. Integraciones desde el día cero para llama.cpp, MLX y ExecuTorch — la publicación con más probabilidades de cambiar lo que ejecutas en local.
14 ago	Qwen3.8-27B	La mitad ejecutable en local de la publicación Qwen3.8. Contexto nativo de 262K, viene con un encoder de visión de 27 capas y su propio preprocesado de imagen y vídeo.
3–12 ago	Qwen3.8-Max	MoE de 2,4T de parámetros, 95B activos, contexto de 1M, texto/imagen/ vídeo nativos, Apache 2.0 — el primer Qwen de nivel Max descargable. Anunciado el 3 de agosto, pesos en Hugging Face y ModelScope hacia el 12; la cobertura sobre el escalonamiento fue inconsistente, así que mira el repositorio y no las noticias.
jul	Kimi K3	2,78T totales, 104B activos. Kimi Delta Attention (atención lineal con control del decaimiento por canal), residuales de atención que permiten a una capa recuperar representaciones de capas anteriores arbitrarias, y enrutado Stable LatentMoE que proyecta 896 expertos a una latente de 3.584 antes del despacho, con 16 activos más 2 compartidos. Reclama ~2,5× la eficiencia de escalado de K2. El soporte en llama.cpp entró el 15 de agosto.
14 ago	GLM-5.3	MoE de 743B, ~40B activos — la <i>misma base</i> que GLM-5.2, con todas las mejoras reportadas atribuidas a más post-entrenamiento. 66,9% en DeepSWE, veinte puntos por encima de 5.2. Pesos MIT, pero escalonados unas dos semanas después del lanzamiento.
4 y 13 ago	Liquid LFM2.5	LFM2.5-2.6B (34T tokens de entrenamiento, construido para bucles de agente on-device) y LFM2.5-VL-3B, un modelo visión-lenguaje de 3,1B que lee pantallas, localiza objetos y llama herramientas en local. La opción sería de gama edge este mes.

Dos piezas de contexto fuera de ventana a las que responde todo lo anterior. El Nemotron-3-Puzzle-75B-A9B de NVIDIA (6 de julio, justo antes de esta ventana) es una variante comprimida de un modelo de la clase 120B producida por un framework de compresión post-entrenamiento llamado "Iterative Puzzle" — compresión a escala como producto que se publica, no como paper. Y DeepSeek-V4 (abril) sigue siendo la arquitectura que persiguen los stacks de inferencia: Compressed Sparse Attention y Heavily Compressed Attention intercaladas que bajan la inferencia a 1M de tokens al 27% de los FLOPs de V3.2 y al 10% de su KV cache. Buena parte del trabajo de Vulkan y CUDA de [§1](#) existe para que esa arquitectura corra en hardware propio.

Arquitectura, memoria, decoding

El post-entrenamiento es ya un eje de escalado por derecho propio

GLM-5.3 es la evidencia más limpia disponible: idéntica base de 743B que su predecesor, veinte puntos de mejora en DeepSWE, y todo atribuido al post-entrenamiento. Cuando la base se mantiene fija y la ganancia es de ese tamaño, la pregunta interesante de presupuesto deja de ser "cuánto cómputo de preentrenamiento" y pasa a ser "cuánto post-entrenamiento, y de qué tipo".

La atención híbrida y lineal se hizo mayoritaria

Que Kimi K3 publique KDA a 2,78T de parámetros es el titular, pero la literatura de apoyo llegó el mismo mes. MixFormer combina varios expertos de memoria con atención lineal consciente del tiempo 2608.09468; un esquema híbrido por bloques reincorpora atención lineal a los modelos de lenguaje por difusión, manteniendo softmax exacto dentro del bloque y pasando a lineal entre bloques anteriores 2608.06628; MARCH escala memoria recurrente con anclas de estado enrutadas por contenido 2608.12435; y un estudio empírico caracteriza los patrones de activaciones masivas que producen estos híbridos, que es justamente el conocimiento que la cuantización necesita para no romperse con ellos 2608.12149.

Memoria y aprendizaje continuo

Macaron-V1 ataca el aprendizaje continuo abierto con auto-mejora y una mezcla de LoRAs 2608.09819; BDH-CQ explora in-context learning con razonamiento latente recurrente 2608.09888; Consolidator aprende a mover memoria de corto a largo plazo cruzando fronteras de contexto 2608.11701. Todavía temprano — pero es la dirección que acabaría haciendo que "el modelo no recuerda la semana pasada" deje de ser un problema de arquitectura que se esquiva con retrieval.

Los modelos de lenguaje por difusión desarrollaron un stack de decoding

El trabajo interesante dejó de ser "¿puede la difusión hacer lenguaje?" y pasó a ser "¿a qué velocidad decodifica?". DARTree extiende la corrección autorregresiva a árboles de draft, sin entrenamiento 2608.13524; Ripple-Pivot explota el efecto de propagación comprometiendo posiciones de entropía media para paralelizar el decoding 2608.11742; LibraSpec reformula la longitud de especulación como optimización de la aceleración esperada 2608.08721; Archer reutiliza estados ocultos en caché de forma asimétrica para una aceleración media de $2,57\times$ 2608.08086.

La infraestructura de agentes se convirtió en área de investigación

Los sistemas de servicio y entrenamiento para agentes ya producen papers de sistemas de verdad: TideRL planifica RL agéntico por disponibilidad con batching continuo de tareas [2608.10402](#); MISA-T planifica rollouts mixtos de RL más allá de la localidad de prefijo con asignación adaptativa de KV [2608.11152](#); CRISP entrena agentes de búsqueda distinguiendo la recogida de evidencia necesaria de los turnos redundantes, recortando turnos un 15–33% [2608.01867](#); HetRoute hace enrutado colaborativo consciente de coste para inferencia MoE distribuida en edge, reportando un 59% menos de latencia y un 72% menos de tráfico [2608.00577](#). En el lado adversarial, OpenART escala el red-teaming de agentes mediante evolución abierta de entornos [2608.00677](#).

§5 · CONCLUSIONES

Qué haría yo de verdad

Ordenado por lo listo que está cada cosa, no por lo interesante que es.

ADOPTAR

Recompila llama.cpp y activa el clonado de caché. Si una carga abre varias peticiones sobre un mismo system prompt o un documento cargado — que es lo que describe la mayoría de montajes con agentes — `/slots clone_to` es throughput casi gratis. Combínalo con suffix decoding, que no necesita alojar draft model, y con prefill desagregado si los prompts largos están matando de hambre a los slots de decode.

ADOPTAR

Cuantiza; no podes. Dos líneas de evidencia independientes este mes, sobre la literatura ya existente, dicen que la cuantización preserva mejor el razonamiento y la fiabilidad que el pruning a compresión equivalente. Si necesitas más de lo que da la cuantización, el siguiente movimiento es on-policy distillation hacia un modelo más pequeño — no pruning estructurado del grande.

ADOPTAR

Cambia cómo se validan los modelos comprimidos. La perplejidad y los benchmarks de opción múltiple se pierden demostrablemente los modos de fallo: pasos de procedimiento inventados en ejecución agéntica, cadenas de razonamiento que ya no sostienen sus propias respuestas correctas, más estereotipos en generación abierta. Un modelo comprimido debería volver a pasarse por trazas de tarea reales inspeccionando la cadena, no solo puntuando la respuesta final.

VIGILAR

Compresión de KV cache. Los resultados son fuertes y el volumen alto, pero casi todo tiene un mes y nada ha aterrizado todavía en un runtime mayoritario. Los métodos basados en asignación (RippleKV, GraceKV) y los conscientes de agentes (CommitKV) son los que esperaba ver llegar antes a un stack de servicio. Lo revisitamos en el próximo número.

VIGILAR

El PR #26794 de llama.cpp, si vuelve troceado. QAT y pruning de expertos en local sobre los mismos formatos de cuantización que el runtime ya sirve cambiaría lo que cuesta hacer fine-tuning de un modelo pequeño a cualquiera que no tenga un clúster. Se rechazó por tamaño, no por el fondo.

SOLO LABORATORIO

Cuantización post-entrenamiento ternaria y de 1,58 bits, incluso con la intuición de calibración de ScaleQ-1.58, y **decoding de LLMs por difusión.** Ambos se mueven de verdad; ninguno tiene una historia de despliegue en la que merezca la pena apoyarse este trimestre.

NOTA

NVFP4 es un formato Blackwell. La ruta NVFP4 de llama.cpp asume soporte de hardware que Hopper no tiene — en H100, las precisiones aplicables son FP8 y MXFP4, no NVFP4. Conviene comprobarlo antes de cuantizar un modelo para el objetivo equivocado.

NOTA

Lista corta de modelos locales, si estás renovando la tuya: Muse Glimmer 30B para trabajo agéntico en una GPU, Qwen3.8-27B donde importen la visión y el contexto largo, LFM2.5-2.6B o VL-3B para on-device de verdad. Todos Apache 2.0.

La tecnología de base

El sustrato que da por supuesto el resto del número. Agrupado por área y ordenado para que cada término se apoye en los anteriores, no

alfabéticamente — está pensado para leerse de principio a fin, no solo para consultarlo.

Cómo un modelo sirve una petición

Prefill / decode

Las dos fases de la generación. El *prefill* procesa el prompt entero en una pasada paralela y está limitado por cómputo puro. El *decode* emite un token cada vez, cada uno dependiendo del anterior, y está limitado por ancho de banda de memoria. Tienen cuellos de botella opuestos, y por eso cada aceleración de este número se indica para una u otra — una ganancia de $9,5\times$ en prefill puede no hacer nada por el decode.

KV cache

Cada token generado atiende a todos los anteriores. En vez de recalcular en cada paso los vectores de clave y valor de todos esos tokens previos, se almacenan. La caché crece linealmente con la longitud de contexto, y en contextos largos supera de forma habitual el tamaño de los propios pesos — por eso comprimirla ([§2](#)) rinde más que comprimir pesos en trabajo de contexto largo.

Prefix caching

Cuando muchas peticiones empiezan igual — el mismo system prompt, el mismo documento cargado — la KV cache de esa apertura compartida se puede calcular una vez y reutilizar en lugar de rehacer el prefill en cada petición. El clonado de caché de llama.cpp ([§1](#)) es esta idea expuesta como endpoint del servidor.

Ventana de contexto

El número máximo de tokens que el modelo puede atender a la vez. Las cifras de titular de [§3](#) (262K, 1M) son lo que la arquitectura soporta, no necesariamente lo que aprovecha bien ni lo que te cabe en memoria.

La atención, y por qué hay tantos tipos

Atención(completa, densa, softmax)

La operación central: cada token se compara con todos los demás para decidir de qué tirar. Exacta, y cuadrática — duplica la secuencia y cuadruplica el coste. Todo lo de abajo es un intento de no pagar eso.

Flash attention

No es una aproximación. Las mismas matemáticas exactas, calculadas por baldosas para que la matriz de atención completa nunca se escriba en memoria. Ganancia pura, y por eso está en todas partes.

MHA / GQA / MLA

La atención multi-cabeza ejecuta varias operaciones de atención en paralelo. La *grouped-query* hace que las cabezas compartan vectores de clave y valor, encogiéndola la caché. La *multi-head latent* comprime claves y valores a un espacio latente de bajo rango, encogiéndola más. Sobre todo son reducciones de tamaño de caché, no de cómputo.

Sliding window attention (SWA)

Cada token atiende solo a los últimos N tokens. La caché deja de crecer y pasa a ser de tamaño fijo — a costa de que el modelo realmente no pueda ver más atrás desde esa capa.

Sparse attention

Atender a un subconjunto elegido de tokens anteriores en vez de a todos, con una selección estructural o aprendida. La Compressed Sparse Attention de DeepSeek-V4 consolida la caché en bloques y luego escoge los k mejores, que es como llega al millón de tokens con el 27% del cómputo de la generación anterior.

Atención lineal

Reformula la atención para que cueste linealmente en vez de cuadráticamente, llevando un estado recurrente de tamaño fijo en lugar de una caché que crece sin parar. Barata y de memoria constante — pero el estado pierde información, así que olvida. Toda la pregunta de diseño es qué olvida y a qué velocidad.

Atención híbrida

Mezclar dos o más de las anteriores dentro de un mismo modelo, normalmente intercalando capas. El intercambio es directo: las capas lineales y de ventana deslizante son baratas pero pierden información, las completas y dispersas son exactas pero caras. Intercalarlas captura casi todo el ahorro de coste conservando suficientes capas exactas para un recuerdo real a larga distancia. Esta es *la* arquitectura dominante de 2026 — Kimi K3 combina KDA con MLA, DeepSeek-V4 intercala atención dispersa comprimida con atención muy comprimida, y Qwen3 y Granite intercalan capas de ventana deslizante con capas completas. Cuando este número dice "híbrida", esto es lo que quiere decir.

KDA

Kimi Delta Attention, la variante lineal de K3. Su rasgo distintivo es el control por canal de la velocidad a la que decae cada dimensión del estado recurrente — un olvido de grano más fino que una única tasa de decaimiento global.

RoPE

Rotary position embeddings: cómo se codifica la posición de un token, rotando sus vectores de query y key un ángulo proporcional a la posición. El enfoque estándar, y lo que normalmente hay que estirar para ampliar una ventana de contexto.

Estructura del modelo

Denso vs MoE

Un modelo *denso* ejecuta todos sus parámetros para cada token. Una *mixture of experts* tiene muchos "expertos" feed-forward en paralelo y un router selecciona unos pocos por token, de modo que la mayor parte del modelo está inactiva en cualquier token dado.

Parámetros totales vs activos

Por qué los modelos MoE se anuncian como "2,4T totales, 95B activos". Los *totales* determinan cuánta memoria necesitas para alojar el modelo; los *activos*, cuánto cómputo cuesta cada token. Esta es toda la razón por la que un modelo de un billón largo de parámetros se puede servir, y también por la que sigue siendo enorme de alojar.

Router /expertos compartidos

El router es la red pequeña que elige qué expertos atienden cada token — entrenarlo para que reparta carga de forma pareja es una dificultad persistente. Los expertos *compartidos* se ejecutan para todos los tokens, sosteniendo la capacidad general para que los enrutados puedan especializarse.

MTP

Multi-token prediction: entrenar al modelo para predecir varios tokens por delante en vez de uno. Las cabezas de predicción extra hacen luego de drafter integrado para speculative decoding, y por eso el soporte de MTP y el de especulación aparecen juntos una y otra vez en [§1](#).

Residuales e hiperconexiones

Las conexiones residuales son los atajos que dejan fluir señal y gradiente por una pila profunda sin obstáculos. El trabajo reciente los generaliza — hiperconexiones con restricción de variedad, residuales de atención que permiten a una capa recuperar representaciones de cualquier capa anterior — porque en modelos dispersos y profundos un atajo simple es un cuello de botella.

Cómo hacer un modelo más pequeño

Cuantización

Almacenar pesos, y a veces activaciones, con menor precisión numérica. La técnica de compresión dominante y, según [§2](#), la que menos daña el comportamiento del modelo.

PTQ vs QAT

La *cuantización post-entrenamiento* convierte un modelo ya terminado — barata, de minutos a horas, sin entrenar. El *quantization-aware training* simula la baja precisión durante el entrenamiento para que el modelo se adapte — caro, pero es lo único que funciona con anchos de bit realmente agresivos.

Calibración

La PTQ necesita una muestra de datos para elegir sus factores de escala. Se trató mucho tiempo como un detalle; [§2](#) es en buena medida la historia de esa suposición rompiéndose — calibrar con los datos adecuados es lo que hizo funcionar la cuantización ternaria de modelos de razonamiento.

Formatos de bits

FP16 y BF16 son la referencia de 16 bits; FP8 e INT8 la reducen a la mitad; 4 bits la reducen otra vez. *Q4_K* y compañía son los formatos de cuantización por bloques propios de llama.cpp. *MXFP4* y *NVFP4* son formatos de microescalado de 4 bits en los que un bloque pequeño de valores comparte un factor de escala — NVFP4 usa bloques más pequeños, captura mejor el rango dinámico local y necesita hardware de la generación Blackwell para ejecutarse de forma nativa.

Solo pesos vs W4A4

Solo pesos almacena los pesos a 4 bits pero vuelve a subir de precisión para calcular, así que ahorra memoria pero no tanto ancho de banda. *W4A4* cuantiza también las activaciones, de modo que la aritmética misma corre a 4 bits. Ese es el problema difícil y el que de verdad desbloquea los matrix engines de baja precisión.

STE

Straight-through estimator: el truco que hace que el QAT sea entrenable siquiera. El redondeo tiene gradiente cero en todas partes, así que la pasada hacia atrás simplemente finge que el redondeo era la función identidad y deja pasar el gradiente sin tocarlo.

Pruning

Borrar partes del modelo directamente. El pruning *no estructurado* pone a cero pesos individuales, produciendo una esparsidad que al hardware le cuesta aprovechar. El *estructurado* elimina unidades enteras — cabezas, capas, expertos, entradas de vocabulario — y sí encoge el modelo de verdad. Según [§2](#), también daña el razonamiento más que la cuantización.

Destilación

Entrenar un modelo *alumno* pequeño para que reproduzca el comportamiento de un *profesor* grande. Distinto de la compresión propiamente dicha: estás entrenando un modelo pequeño nuevo, no encogiéndolo grande.

Destilación off-policy vs on-policy

La *off-policy* entrena al alumno sobre texto que produjo el profesor. La *on-policy* hace que el alumno genere y el profesor califique los intentos del propio alumno — así los estados con los que el alumno entrena son los que se va a encontrar desplegado. Ese desajuste es lo que la destilación off-policy hace mal, y cerrarlo es la razón de que la on-policy se impusiera este año.

LoRA / QLoRA

Hacer fine-tuning entrenando pequeñas matrices adaptadoras de bajo rango junto a pesos congelados, en vez de actualizar el modelo entero. *QLoRA* lo hace sobre una base ya cuantizada, que es lo que permite hacer fine-tuning de modelos grandes en una sola GPU.

Cómo hacer la inferencia más rápida

Speculativ decoding

Algo barato propone los siguientes tokens; el modelo real los verifica todos en una única pasada paralela y se queda con los que acepta. Como la verificación es paralela y la generación es secuencial, esto convierte pasos de decode limitados por memoria en pasos limitados por cómputo. Y algo crítico: la distribución de salida no cambia — es una aceleración, no una aproximación.

Draft model

El modelo pequeño que hace las propuestas. Tiene que estar residente en memoria y más o menos alineado con el modelo objetivo, que es el coste principal de la técnica.

Suffix decoding

Especulación sin draft model alguno: un índice sobre el texto ya visto propone continuaciones, cosa que funciona bien porque la generación real se repite a sí misma — código, nombres, contexto citado. Elimina por completo el coste en memoria del draft model ([§1](#)).

Prefix desagregado

Ejecutar la fase de prefix en workers separados para que un prompt largo no pare la generación de tokens del resto de usuarios del servidor.

Etapas de entrenamiento

Pre- vs post-entrenamiento

El *preentrenamiento* es la gran tirada no supervisada que construye el modelo base. El *post-entrenamiento* es todo lo posterior — fine-tuning supervisado, aprendizaje por refuerzo, destilación. Que GLM-5.3 se mueva veinte puntos sobre la misma base ([§4](#)) es la razón de que el post-entrenamiento se trate ya como un eje de escalado propio.

SFT

Supervised fine-tuning: entrenar sobre demostraciones del comportamiento deseado. Lo más simple y todavía la columna vertebral del post-entrenamiento.

RL / RLVR / GRPO

El aprendizaje por refuerzo optimiza contra una recompensa en vez de contra objetivos fijos. *RLVR* usa recompensas comprobables de forma automática — los tests pasan, la respuesta es correcta — y por eso funciona tan bien en código y matemáticas. *GRPO* es el algoritmo de optimización de política relativa a grupo que se convirtió en la forma estándar de ejecutarlo.

Fuentes

LLAMA.CPP

[Releases de llama.cpp](#) — etiquetas de build y notas por build (se citan b10444–b10451).

[Informe semanal de GitHub, 7–14 jul 2026](#)

[Informe semanal de GitHub, 13–20 jul 2026](#)

[Informe semanal de GitHub, 27 jul–3 ago 2026](#)

[Informe semanal de GitHub, 4–11 ago 2026](#)

[PR #26794](#) — entrenamiento MoE, QAT y llama-prune ; cerrado el 11 ago 2026.

Los PRs individuales están enlazados en línea a lo largo de [§1](#).

COMPRESIÓN, PRUNING, DESTILACIÓN

Todos los identificadores de arXiv de [§2](#) y [§4](#) enlazan a [arxiv.org/abs/](#) y se obtuvieron vía la API de arXiv filtrada al 16 jul – 16 ago 2026.

[Thinking Machines Lab — On-Policy Distillation](#)

[A Survey of On-Policy Distillation for Large Language Models](#)

[UniComp: evaluación unificada de compresión por pruning, cuantización y destilación](#)

[Revisiting Pruning vs Quantization for Small Language Models](#) — el resultado previo que el trabajo nuevo no deja de confirmar.

[Survey sobre optimización de KV cache consciente del sistema](#)

[Blog técnico de Kimi K3](#)

[CNBC — Meta lanza Muse Glimmer y InfoQ — detalle de arquitectura](#)

[Especificaciones de Qwen3.8-Max y especificaciones y fecha de Qwen3.8-27B](#)

[Liquid AI LFM2.5-VL-3B](#)

[Arquitectura y benchmarks de GLM-5.3](#)

[DeepSeek-V4 — CSA y HCA \(abril 2026, fuera de ventana, citado como contexto\)](#)

Compilado el 16 de agosto de 2026 a partir del feed de releases y los informes semanales de GitHub de llama.cpp, la API de arXiv filtrada a esta ventana y la cobertura de publicaciones de modelos. Las cifras citadas como aceleraciones o ratios de compresión son las que reportan los propios autores sobre sus propios benchmarks — trátalas como afirmaciones, no como mediciones, hasta reproducirlas. Todo lo fechado fuera del 16 de julio – 16 de agosto de 2026 se indica en línea como contexto.



LLM Radar lo publica semanalmente [Montevive](#). [Todos los números](#) · ai@montevive.ai