AYI NEDJIMI Consultants

FlashQuant : Compression KV-Cache 7.5x C

10 mai 2026 • Mis à jour le 17 mai 2026 • 13 min de lecture • 270 mots

> FlashQuant est un projet open source C++17/CUDA qui compresse le KV-c ratio 7.5x via quantization INT4 et codebooks par tete. Integration native vL (<1%), VRAM divisee par 7.5 sur Llama 3.1 70B (40 Go → 5.3 Go).

FlashQuant est un projet open source écrit en C++17 et CUDA qui s'attaque au go critique des LLM modernes en inférence : la consommation mémoire du KV-cache comme Llama 3.1 70B servi avec un contexte de 32K tokens, le KV-cache occupe disponible, dépassant régulièrement les 40 Go par requête longue. FlashQuant int compression extrême combinant quantization INT4 par groupe, dictionnaires de c CUDA dédiés, atteignant un ratio de compression moyen de 7.5x avec une dégrad % sur MMLU et HumanEval. Conçu comme un plugin pour vLLM 0.6.x+, FlashQua PagedAttention et permet de servir des contextes longs sur GPU consumer (RTX 4 le modèle ne tenait précédemment que sur des H100 80 Go. Cette page projet doc benchmarks, l'installation et la roadmap. Le code so ublié sous l github.com/ayinedjimi/FlashQuant.

À RETENIR

A retenir

Ratio 7.5x de compression KV-cache (FP16 → INT4 + codebooks) sur Llam

VRAM divisee par 7.5 : 40 Go → 5.3 Go pour 32K tokens de contexte.

Perplexite preservee : degradation inferieure a 1 % sur MMLU et HumanEv

Latence overhead inferieur a 5 % grace a la dequantization on-the-fly fusi

Plugin vLLM compatible PagedAttention, sm_80+ requis (Ampere et superi

Le probleme : le KV-cache devore la VRAM

Lorsqu'un LLM autoregressif genere du texte, chaque nouveau token doit acceder Value de tous les tokens precedents. Pour eviter de recalculer ces tenseurs a cha stocke un KV-cache. Sa taille croit lineairement avec la longueur de contexte, le n de tetes d'attention et la dimension par tete. Pour Llama 3.1 70B en FP16, le KV-ca 1K tokens, soit plus de 40 Go pour un contexte de 32K. Sur des modeles MoE ou d explose au-dela des 100 Go par sequence active. Resultat : meme un H100 80 Go session simultanement, et les GPU consumer sont totalement exclus du jeu. Cette batching, augmente le cout par token et empeche le deploiement de RAG long-co abordable.

Concretement, le calcul est implacable : pour Llama uches, 8 K Attention), dimension 128 par tete, en FP16 (2 octets nsomme 8

Réponse sous 24h

Devis gratuit →