AYI NEDJIMI Consultants

vLLM : Moteur d'Inférence LLM Haute Perform 2026

10 mai 2026 • Mis à jour le 17 mai 2026 • 23 min de lecture • 4748 mots •

vLLM est un moteur open-source d'inférence et de service pour LLM, écri Python et CUDA, conçu pour offrir un débit maximal et une latence prévisi GPU et accélérateurs spécialisés. Né en 2023 au Sky Computing Lab de U Berkeley sous l'impulsion de Woosuk Kwon, Zhuohan Li, Ion Stoica et Hao vLLM cumule en mai 2026 plus de 52 000 étoiles GitHub, 1 100 contribute partie de la PyTorch Foundation. Cette page entity-first détaille PagedAtte continuous batching, l'architecture worker/scheduler/executor, les 250+ architectures supportées (Llama 4, Mistral, Mixtral, Qwen 3, DeepSeek V3, Gemma 3, GLM-4.5), les backends CUDA/ROCm/CPU/TPU/Neuron/Gaudi, formats FP8/AWQ/GPTQ/NVFP4, l'API OpenAI-compatible, le speculative d le disaggregated prefill, le prefix caching, le multi-LoRA serving, la vLLM P Stack Helm Kubernetes, le monitoring Prometheus et les benchmarks face TensorRT-LLM, llama.cpp et SGLang.

vLLM est un **moteur d'inférence et de service** pour *large language models* (LLM)
source, écrit en Python et CUDA, conçu pour offrir un **débit (throughput) maximal**
latence prévisible sur GPU et accélérateurs spécialisés. Initialement développé en
au sein du **Sky Computing Lab** de l'**Université de Californie à Berkeley** par **Woosu**
Zhuohan Li et leurs co-auteurs sous la direction des professeurs **Ion Stoica** et **Hao**
vLLM a été rendu public le **20 juin 2023** et a depuis été adopté comme *backend* d
référence par une grande partie de l'industrie : Anyscale, Databricks, Anthropic (p
charges internes), AWS Bedrock (sur certaines familles de modèles), Cloudflare W
Lambda Labs, Together AI, RunPod, Mistral La Plateforme et l'écosystème Red Ha
AI. La version **v0.10.x** de mai 2026 cumule plus de **52 000 étoiles GitHub**, dépass
contributeurs et fait partie des projets phares de la *PyTorch Foundation*, où elle a
en septembre 2025. Son innovation algorithmique principale, **PagedAttention**, tran
pagination mémoire des systèmes d'exploitation au *KV cache* des transformers et
un facteur 2 à 24 sur le débit par rapport à des moteurs naïfs (HuggingFace TGI v1
FasterTransformer historique). vLLM expose nativement une **API HTTP compatible**
(*Chat Completions*, *Completions*, *Embeddings*, *Tools/Function calling*), supporte p
architectures (Llama 2/3/4, Mistral, Mixtral, Qwen 2/2.5/3, DeepSeek V2/V3/R1, Ph
Gemma 2/3, GLM-4, Yi, Command-R, MiniCPM, Aya, Granite, InternLM, OLMo, Sta
Falcon Mamba), accepte les principaux **formats de quantization** (FP16, BF16, INT8
GPTQ, FP8 E4M3/E5M2, GGUF en lecture partielle, SqueezeLLM), tourne sur **CUD**
Hopper/Blackwell/Ada/Ampere), **ROCm** (AMD MI250/MI300X/MI325X), **CPU** (Intel
AVX-512, ARM Neoverse), **TPU** (Google v5e/v5p/Trillium) et **AWS Neuron** (Trainiu
Inferentia2). Cette page entity-first détaille l'histoire, l'algorithme PagedAttention,
batching, l'architecture interne (worker, scheduler, executor), la matrice de compa
modèles/backends/formats, l'API OpenAI-compatible, les techniques avancées (S
Decoding, Disaggregated Prefill, Prefix Caching, multi-LoRA serving), le déploiem
Kubernetes via la *vLLM Production Stack*, le monitoring Prometheus, les benchma
Ollama, TensorRT-LLM, llama.cpp et SGLang, ainsi q[illegible]ns de sécu

Réponse sous 24h

Devis gratuit →