

Kalorad: Technical Overview

Bohdan Havryliuk, Founder & CEO | kaloradteam@gmail.com | kalorad.kaloradteam.workers.dev | linkedin.com/in/bohdan-havryliuk-370762386 | September 2026

What it is

Kalorad is a sovereign AI platform for organisations and professionals that runs on its own language model, KaloradLM. No third-party AI or inference API sits anywhere in the stack: the architecture, the training loop, the data pipeline and the serving layer are written in-house. For regulated and public-sector customers it means that nothing a user writes leaves the operator's infrastructure, which is the product itself rather than a feature.

Status: pre-launch. The platform (ASP.NET Core backend, billing, Telegram bot, OpenAI-compatible model server) is built and test-covered. The training stack has completed one full pretraining run on real hardware (below). The production-size 1.24B model needs about 163 GPU-hours, roughly \$300 of compute at current market prices.

What is built

Component	Implemented	Notes
Model (KaloradLM)	Dense decoder-only; grouped-query attention and multi-head latent attention (MLA); SwiGLU; RoPE with YaRN scaling; RMSNorm; mixture-of-experts presets with a shared expert (up to 11.6B total).	Presets from 6M (tests) through 683M and 1.24B. Attention backend selectable: SDPA, manual, flash.
Training loop	Reference Muon + AdamW, fp32 master weights with bf16 compute, warmup-stable-decay schedule, sequence-length curriculum (512 to 4096), gradient accumulation and checkpointing, torch.compile, DDP; resume continues the corpus where it stopped.	Peak-memory probe at the longest sequence before step 0, so an OOM shows up in minute one rather than hour six.
Data pipeline	Streaming from HuggingFace, Gopher-style quality filter plus mojibake, URL-density and repetition penalties, MinHash-LSH deduplication, proportional multi-source scheduler, SentencePiece tokenizer (Ukrainian, Russian, English, code).	Vectorised MinHash runs 43x faster than the reference loop; signatures verified identical.
Serving	OpenAI-compatible /v1/chat/completions with streaming and KV cache; constrained decoding over a grammar; deterministic answer selection (self-consistency for maths, test execution for code, cite-or-abstain for documents).	Runs on CUDA, ROCm and Apple MPS. A client migrates by changing base_url; verified end to end from VS Code Copilot Chat.
User memory	Bitemporal store: valid time and transaction time per fact, supersede/end/as-of queries, importance decay that retires rather than deletes, explicit GDPR erase.	"What did we believe in April" is answerable; a correction keeps its predecessor.
Verticals and self-improvement	Per-domain LoRA adapters with routing and hot swap. Closed Run-Update-Train-Optimise-Rerun loop with automatic scoring, checkpoint promotion and rollback on regression.	Adapters are wired; training them is on the compute-gated roadmap.
Tests	29 test modules covering the model, training objective, optimizer against the reference, curriculum, attention-backend equivalence, RoPE dtype, shard splitting, corpus mixing, quality filter, dedup speed, memory semantics.	Every bug found during the GPU run got a test that fails without the fix.

What has been measured (September 2026, one AMD MI300X, ROCm)

Item	Result
Full pretraining run	683M parameters (hidden 1536, 24 layers, GQA 24/4 heads, SwiGLU 4096), 13.1B tokens, 16,750 optimizer steps, 43 GPU-hours, \$86. Model size chosen by Chinchilla compute-optimal sizing for the credit budget available, i.e. for quality per FLOP rather than headline size.

Throughput	88,000 tokens/s for the 683M preset in bf16 during the run (roughly 24% MFU). 42,748 tokens/s for the 1.24B preset with SDPA and torch.compile (batch 32 x 2048).
Numerical parity	CPU versus accelerator logits agree to 3.3e-06 in fp32; manual, SDPA and flash attention backends agree on the same inputs, including the cached-decode case where is_causal would silently be wrong.
torch.compile on ROCm	1.32x speedup and 37% lower peak memory on the 1.24B preset; max-autotune crashes on CUDA graphs, default mode is the one that ships. SDPA on top: a further 1.84x.
Corpus build on the node	12.5B tokens built in-place at 0.8 to 1.4M tokens/s with 12 to 16 worker processes; HuggingFace reads at 15.8M tokens/s versus 33k at home, so the pipeline moved to the GPU node.
Resulting checkpoint	A base model without instruction tuning; perplexity is research-grade, not product-grade, and I say so. Only the last 3,000 steps ran on the corrected training objective (see below), and the optimizer was later found to be non-reference (see below), so the run is a proof of the pipeline rather than of the model.

What went wrong during the run, and how I found it

The run doubled as an audit of my own defaults. Each item below is now a test.

- **Training objective was off by one.** The loss compared position t with token $t+2$. Found at step 13,750 by a test that feeds a known sequence and checks which token the logits favour; fixed to $t+1$.
- **Data loader duplicated shards 4x across workers.** Each worker read the whole file list. Fixed at step 3,000 with stride-based sharding and a test that counts unique sequences per epoch.
- **Validation split was 100% code.** A tail split landed on the code block of the shard list. Replaced with a strided split with a half-offset so every source is represented.
- **Phase-3 OOM after hours of healthy training.** The curriculum's last phase used the config's 4096 while the sweep had measured 2048. Fixed with a peak-memory probe at the true maximum length before step 0.
- **RoPE crashed under bf16 on ROCm** (Float versus BFloat16 in the cache). Kept an fp32 master of `inv_freq` outside the buffers and cast at apply time.
- **The optimizer was not the reference Muon, and master weights were bf16.** Found in review after the run: the update was scaled to the gradient norm instead of the spectral norm, weight decay sat inside the gradient, and in bf16 96% of weights did not move per step at $\text{lr } 2\text{e-}3$ - the decay phase trained almost nothing. Fixed; on a 25M-token sweep the reference Muon at $\text{lr } 0.01$ now leads AdamW.

Stack

Python, PyTorch (CUDA / ROCm / MPS), SentencePiece, HuggingFace datasets, numpy, FastAPI server; ASP.NET Core for the product backend; Cloudflare Workers for the site. Hardware used so far: AMD MI300X (AMD Developer Cloud), Apple M-series for local inference.

Roadmap, gated on compute

1.24B pretraining over 25B trilingual tokens with the corrected objective and optimizer (about 163 GPU-hours), a short SFT pass for chat format, a published Ukrainian benchmark, eight vertical LoRA adapters measured against the base model, and a 3.16B mixture-of-experts preset with 1.84B active parameters. Applications filed: Scaleway Founders Program, Lambda Research Grant, USF Startup EDGE (pre-registration).

Code is in a private repository. I am glad to walk through the architecture, the training loop and a live request from VS Code on a call.