

# Kalorad: Technical Overview

Bohdan Havryliuk, Founder & CEO | [bohdan@kalorad.app](mailto:bohdan@kalorad.app) | [kalorad.kaloradteam.workers.dev](https://kalorad.kaloradteam.workers.dev) | [linkedin.com/in/bohdan-havryliuk-370762386](https://linkedin.com/in/bohdan-havryliuk-370762386) | October 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, web and desktop apps, Telegram bot, OpenAI-compatible model server) is built and test-covered. The training stack has completed one full pretraining run on real hardware, and the corpus for the production model is built (below). The production model - a mixture of experts with multi-token prediction, 1.32B active parameters, 28B tokens - needs about 230 GPU-hours, roughly \$500 of compute at current market prices.

## What is built

| Component                             | Implemented                                                                                                                                                                                                                                                                                               | Notes                                                                                                                                    |
|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Model (KaloradLM)</b>              | Decoder-only transformer: grouped-query and multi-head latent attention, SwiGLU, RoPE with YaRN, RMSNorm; mixture of experts with a shared expert; multi-token prediction heads. Two modes on shared weights: Instant (two experts, one pass) and Deliberate (four experts, the middle layers run again). | Presets from 6M (tests) to the production model: 2.8B total, 1.32B active parameters. Attention backend selectable: SDPA, manual, flash. |
| <b>Training loop</b>                  | Reference Muon + AdamW, fp32 master weights with bf16 compute, warmup-stable-decay schedule, sequence-length curriculum (512 to 4096) over one continuous data stream, gradient accumulation and checkpointing, torch.compile, DDP.                                                                       | Peak-memory probe at the longest sequence and with optimizer state before step 0; resume is bit-exact (data position, RNG, mode order).  |
| <b>Data pipeline</b>                  | Streaming ingestion, Gopher-style quality filter plus mojibake, URL-density and repetition penalties, MinHash-LSH deduplication, benchmark decontamination, proportional multi-source scheduler, own 48k SentencePiece tokenizer (Ukrainian, Russian, English, code).                                     | Production corpus built on Google Cloud and verified byte for byte. Vectorised MinHash runs 43x faster than the reference loop.          |
| <b>Serving</b>                        | OpenAI-compatible /v1/chat/completions with streaming and KV cache; constrained decoding over a grammar; typed decisions with calibrated confidence (/v1/decide); deterministic answer selection (self-consistency for maths, test execution for code, cite-or-abstain for documents).                    | A client migrates by changing base_url. Runs on CUDA, ROCm and Apple MPS; an offline installer for closed networks is tested.            |
| <b>User memory</b>                    | Bitemporal store: valid and transaction time per fact, supersede/end/as-of queries, importance decay that retires rather than deletes, explicit GDPR erase; recall ranks by the question.                                                                                                                 | "What did we believe in April" is answerable; a correction keeps its predecessor.                                                        |
| <b>Verticals and self-improvement</b> | Per-domain LoRA adapters with routing and hot swap. Closed run-update-train-optimize-rerun loop with automatic scoring, checkpoint promotion and rollback on regression.                                                                                                                                  | Adapters are wired; training them is on the compute-gated roadmap.                                                                       |
| <b>Tests</b>                          | 94 test modules, 580+ tests: model, training objective, optimizer against the reference, curriculum, attention-backend equivalence, exact resume, data stream, quality filter, dedup speed, memory semantics.                                                                                             | Every bug found on real hardware got a test that fails without the fix.                                                                  |

## What has been measured

| Item                                                          | Result                                                                                                                                                                                                                                                                             |
|---------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>First full pretraining run (Sept 2026, one AMD MI300X)</b> | 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.                                                           |
| <b>Throughput</b>                                             | 88,000 tokens/s for the 683M preset in bf16 during the run (about 24% MFU).                                                                                                                                                                                                        |
| <b>Numerical parity</b>                                       | CPU versus accelerator logits agree to 3.3e-06 in fp32; manual, SDPA and flash attention backends agree, including the cached-decode case where is_causal would silently be wrong.                                                                                                 |
| <b>Production corpus (Oct 2026)</b>                           | 23B tokens from commercially licensed sources only (40% code, 22% Ukrainian, 22% Russian, 16% English), deduplicated and cleaned of benchmark text, plus a 4.9B-token annealing set.                                                                                               |
| <b>Architecture comparison (Oct 2026, one H100)</b>           | 350M-parameter models on 164M tokens of the production corpus: mixture of experts with multi-token prediction 3.175 validation loss against 3.251 for a dense model of the same active size (-0.076, stable along the curve); a looped variant did not help and cost 23% per step. |
| <b>Production preflight (H100 80 GB)</b>                      | Batch 1 x 4096 with optimizer state fits (56 GB); resume reproduces the uninterrupted run exactly.                                                                                                                                                                                 |
| <b>Resulting checkpoint</b>                                   | A base model without instruction tuning; perplexity is research-grade, not product-grade, and I say so. The 683M run proved the pipeline, not the model (see below).                                                                                                               |

## What went wrong during the first 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 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 so every source is represented.
- **Phase-3 OOM after hours of healthy training.** The curriculum's last phase used 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.** Kept an fp32 master of inv\_freq outside the buffers, cast at apply time.
- **The optimizer was not the reference Muon, and master weights were bf16.** 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. Fixed; on a 25M-token sweep the reference Muon at lr 0.01 leads AdamW.

## Stack

Python, PyTorch (CUDA / ROCm / MPS), SentencePiece, numpy, FastAPI; ASP.NET Core for the product backend; Cloudflare Workers for the site. Hardware used so far: AMD MI300X (AMD Developer Cloud) for the first run, NVIDIA H100 for architecture comparisons, Google Cloud for the corpus build, Apple M-series for local inference.

## Roadmap, gated on compute

Production pretraining: mixture of experts with multi-token prediction (2.8B total, 1.32B active parameters) on 28B tokens, about 230 GPU-hours, with the deliberate mode trained in the annealing phase; a short SFT pass for chat format; a published Ukrainian benchmark; eight vertical LoRA adapters measured against the base model. Compute applications filed: EuroHPC AI Factories (10,000 GPU-hours), Google for Startups Cloud, Google TPU Research Cloud, Lambda Research Grant.

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.