This Week in Local AI — I Built DFlash and Audited Lightning

May 3, 2026 · by Mark Bartlett

Download this post as PDF

InsiderLLM Weekly issue 7 – May 3, 2026

Spent three days building DFlash from source to bench it on my own RTX 3090. Spent another day running a 5-minute audit on my own stack after PyPI's `lightning` package got hit by malware. Both pieces produced firsthand data nobody else had — and on one of them, a piece of news the README didn't tell you. Long week.

DFlash on a Real RTX 3090: I Built It and Tested It

Built DFlash from source on Miu (RTX 3090, 24GB) and ran the full `bench_llm.py` suite against both Qwens with their matching drafts. Mean speedups: **2.59x for Qwen 3.5-27B Q4_K_M, 2.56x for Qwen 3.6-27B Q4_K_M**. Per-bench: 3.5 hits 2.76x on HumanEval, 2.48x on GSM8K, 2.53x on Math500. 3.6 hits 2.81x / 2.25x / 2.61x.

The honest read on the gap. The Luce DFlash README headlines 3.43x for Qwen 3.5 HumanEval — I got 2.76x. Three reasons that's plausible: `bench_llm.py` defaults aren't necessarily what produced 3.43x, single-3090 memory pressure on the tree-verify stage, and DDTree variance at temp=1.0. Not a complaint about the project — just calibration for readers about to pull the repo and run the same script.

The flip side. On Qwen 3.6 I got **2.56x mean against the README's 1.98x**. The 3.6 draft was "still training" when the README was written on April 26; three days later it had matured. Acceptance length on HumanEval climbed from 5.94 in the README to 7.48 on my run. Higher AL flows directly into higher speedup with the same DDTree budget. The gap is closing as the draft trains.

Two days later, when r/LocalLLaMA filled with "I can't replicate" posts, I wrote a troubleshooting piece pointing at the same mismatch from the other end. The original DFlash piece also got a PFlash refresh on May 1 — 10x prefill speedup at 128K context, which changes the long-prompt math.

📖 **The full bench, both Qwens, is here.**

Lightning 2.6.x: First Documented Claude Code Hook Attack

PyPI shipped poisoned versions of the `lightning` package on April 30 — 2.6.2 and 2.6.3, both carrying a 14.8 MB obfuscated JavaScript payload. Standard Shai-Hulud loadout: scoops GitHub tokens, AWS/Azure/GCP credentials, and Actions secrets. What's new is what it does after the smash-and-grab. It plants persistence in three places, including a **SessionStart hook in `.claude/settings.json`** that re-runs the malware every time you open Claude Code in an infected repo.

That's the first documented case of malware abusing Claude Code's hook system in the wild. Hooks fire automatically. No user action, no consent gesture. `git pull` an infected branch, open Claude Code in that working tree, the malware runs. It also drops a `.vscode/setup.mjs` parallel dropper and a `.github/workflows/Formatter.yml` that exfiltrates secrets via `${{ toJSON(secrets) }}` on every push.

I audited my own stack on April 30 and was clean. The article walks through the five commands I actually used: `pip list` for the package, `find` for `_runtime` directories, `find` for IOC files, `cat ~/.claude/settings.json` for unexpected hooks, `find` for the `Formatter.yml` workflow. Five minutes start to finish. One false-positive tip: a `lightning` directory inside `torch/` is internal to PyTorch, not the malicious top-level package.

If you've installed `lightning` at any point this year, run the audit before you open Claude Code today.

📖 **The full audit with all five commands is here.**

What It Takes to Run Qwen 3.6-35B MoE Locally

The 35B-A3B MoE landed mid-April but the practical "what your hardware actually needs" piece took a week to lock down. Honest hardware envelope, llama.cpp settings that work, and where this fits versus the 27B dense for OpenCode and Claude Code workflows.

📖 **The hardware-and-setup guide is here.**

Quick Hits

- **DeepSeek V4-Pro got a NIST CAISI evaluation** — on par with GPT-5 across the suite. Useful baseline if you're choosing between local V4 and frontier APIs.

- **gemma-4-31B-it-DFlash dropped May 1.** DFlash expanding past Qwen is the more interesting story than any single benchmark — block-diffusion drafting wasn't supposed to be a Qwen-only thing.
- **Mistral Medium 3.5 went live April 29** — dense 128B, 256k context, modified MIT license with a revenue carve-out for large companies. We refreshed the Mistral guide the same day.
- **NVFP4 native support landed in llama.cpp b8967** for Blackwell, with b9002 just shipped on top of it. Real Blackwell speedup numbers (around +60% on prefill) are starting to surface on r/LocalLLaMA.
- **Unsloth fixed a Mistral 3.5 implementation bug** announced May 1. If you tried Mistral 3.5 GGUFs early, redownload from the Unsloth repo.
- **Stack versions:** llama.cpp at b9002, vLLM at v0.20.1.
- **Watch list:** Lightning 2.6.4+ release as the all-clear signal for Python ML devs. Pin `lightning<2.6.2` in any requirements.txt you control until then.

That's the week. Reply if there's something specific you want me to cover next.

— Mark, InsiderLLM

Running local AI on weird hardware? Built something novel with it? Drop us a line at hello@insiderllm.com.

Source: https://insiderllm.com/blog/newsletter-2026-05-03/