

An \$8 chip, twice as fast, same story

October 12, 2026 · by Mark Bartlett

[Download this post as PDF](#)

InsiderLLM Weekly issue 25 – October 12, 2026

An \$8 microcontroller now writes its children's stories more than twice as fast as it did out of the box, and not one byte of the output changed. That's below. Also this week: four new articles, and an unpatched remote-code hole in LMCache that vLLM users should check.

New This Week

- [LLM on a Microcontroller: ESP32-S3 at 22.6 tok/s](#). The main piece, below.
- [Strata vs llama.cpp on an RTX 3090: 78 tok/s vs 18 on the Full Flash-Next](#). A 24 GB card and 64 GB of RAM run the full 177B Qwen3.8-Flash-Next, and Strata, an engine built for that one model, decodes it about four times faster than llama.cpp and reads a 32K prompt in seconds instead of minutes.
- [Does RAM Speed Matter for Local LLMs? 2133 vs 3000, Measured](#). Faster RAM does nothing for a model that fits on the GPU, but with MoE experts offloaded to the CPU it's one of the cheapest speedups you can buy: 13 to 29% faster going from DDR4-2133 to 3000 on one box.
- [Qwen 3.8 35B-A3B Is a Distill: Tested Against Qwen 3.6](#). The Qwen 3.8 35B-A3B you found is a community fine-tune of Qwen 3.6 35B-A3B, not an Alibaba release. On my 3090 it scored within one item of its base model at the same speed, then ignored the instruction and chatted on 6 of 47 messages.

Updated:

- [Best Hardware for Running OpenClaw – Mac Mini vs VPS vs Your Old PC](#) (2026-10-09): the \$500 tier is now the Optiplex-plus-RTX 3060 build alone, about \$375-475, because Apple no longer sells the \$599 Mac mini M4 it used to recommend.

Quick Hits

- [LMCache has an unpatched remote-code hole](#). CVE-2026-105192 (CVSS 9.8) lets one message to its multiprocess ZMQ port, default 5555, run code, as root in the official images,

and JFrog found it from 0.3.9 to 0.5.5 with no fixed release as of October 12. If you run vLLM with LMCache, keep `--host` off routable addresses.

- **Mistral Large 4.** A 1.05T-parameter MoE with 52B active, in public preview on Mistral's API since October 6, with the weights promised "by the end of the month". Even at 2 bits per weight that's about 260 GB, so it's open weights you won't run at home. (ours: [open weights you can't run](#))
- **DeepSeek V4.1-Flash runs off an SSD, slowly.** Colibri streams the 510 GB checkpoint from disk with no conversion, at 0.21 to 0.96 tok/s by its own measurements, and the open llama.cpp port ran its 246 GiB Q2_K at 2.8 tok/s on a 128 GB DGX Spark. It's not a download for a 32 or 64 GB box. ([PR #28696](#); ours: [SSD-streaming engines](#))
- **llama.cpp v0.6.0.** It adds GLM-5.3-Flash, a new `llama_batch_ext` batch API and MTP speculative decoding for Qwen4Exp, which the notes put at about 1.5x decode on a DGX Spark. The session and state file formats changed version, so expect sessions saved on older builds not to load.
- **Anthropic will scan open-source projects for free.** Maintainers apply to OSS Scanner by pull request, and the reports arrive with no human review, though Anthropic's pentesters found 85 of 97 sampled high and critical findings met its disclosure bar. If you maintain a local-AI server people expose to a network, it's worth the PR. ([the oss-scanner repo](#))
- **Ecosia drops Mistral for open-weight models, Qwen, GLM and Kimi among them.** CEO Christian Kroll told Politico, as TechNode reports, that moving to models hosted by the German platform Melious roughly halved costs and improved quality. A production search engine just picked the same model families you run at home.

Twice the Speed on an \$8 Chip, Same Story

A Better Stack video sent a lot of people off to flash slvDev's esp32-ai onto an ESP32-S3: a 28.9M-parameter story model on a chip that costs about \$8. Unmodified, it matched upstream's own profile to within 0.1 ms per token. With the display code switched off, since I had no screen attached, it ran at 10.34 tok/s. That was my baseline.

Then I made five changes: SIMD for the output head, an int4 head, SIMD for the rest of the model, the logits moved into fast internal RAM, and an int4 core. The board finished at 22.59 tok/s, 2.18x the baseline, and every build wrote the same 200-token story, byte for byte.

The chip spends most of each token waiting on its external PSRAM, which reads at about 62 MB/s, and the output head alone reads 2.4 MB of weights per token. Halve the bytes and you roughly halve the time. My first experiment, a CPU clock sweep, said the opposite, and all three

of its predictions failed: slow scalar math was hiding the memory floor. Once SIMD took the math away, the floor showed. If you take one thing from the fork, take the head work, which accounts for 35 of the 51 ms saved per token.

The patches are in my fork, one branch per change, and I've offered them upstream in issue #26. The model still only writes children's stories, and it still has a soft spot for a girl named Lily.

📖 **Full breakdown** [here](#). 📖 **The fork is** [here](#). 📖 **Upstream issue #26 is** [here](#).

That's the week. If there's an ESP32-S3 N16R8 in a drawer, flash upstream first, then the fork, and compare.

Next: the same model chatting on the ESP32-P4, and what it costs in energy.

New here, reading this on the web? [Subscribe](#) and the next one lands in your inbox.

— Mark, InsiderLLM

Got an ESP32-S3 or another microcontroller running a model? Measured it? Reply, or hit me at hello@insiderllm.com. I read everything.

A weekly email with every new guide and measured benchmark.

[Subscribe — free, no spam](#)

Source: <https://insiderllm.com/blog/newsletter-2026-10-12/>

Free guides for running AI locally