MoE expert streaming explained: how a 180B model can run on a 32 GB Mac

23 November 2026 · 7 min read · blogs.iictp.org/moe-expert-streaming-transformers-run-180b-model-32gb-explained

Short answer: expert streaming keeps only the “hot” experts of a mixture-of-experts (MoE) model in fast memory and reads the rest from RAM or SSD on demand. A Hugging Face community post on 18 November 2026 reports that a Transformers implementation reached about 14 tokens per second on a 32 GB Mac with a 180B-parameter model, within roughly 6% of the Strata engine. It matters because MoE experts are over 90% of the weights, yet each token uses only a few.

TL;DR The short version

- Expert streaming keeps only the 'hot' experts of a mixture-of-experts model in fast memory and reads the rest from RAM or SSD on demand.
- A Hugging Face community post reports about 14 tokens per second for a 180B-parameter model on a 32 GB Mac with a Transformers implementation.
- Results are one author's on specific hardware; if your model fits in RAM, llama.cpp with experts on the CPU may be enough.

Read online, listen to it, or change the text size: https://blogs.iictp.org/moe-expert-streaming-transformers-run-180b-model-32gb-explained

What is MoE expert streaming?

In an MoE model a router sends each token to a small number of experts, for example 8 or 10, so most weights sit idle at any moment. The post’s author says experts are 90%+ of an MoE model’s weights. Streaming loads only the routed experts into a bounded cache and leaves the rest on slower storage. This is a personal experiment, not a vendor release.

How other engines handle it

- **llama.cpp:** memory-maps the GGUF and can keep experts on the CPU, computing them there. It works well when experts fit in RAM, but the page cache thrashes when they do not.
- **vLLM:** `--cpu-offload-gb` puts weights in pinned RAM read over PCIe on every use, with no cache and no disk tier.
- **Transformers with Accelerate:** copies a whole layer’s expert bank to the GPU per forward pass, with no routing awareness, and on a Mac “offload to CPU” frees nothing because memory is unified.

The key idea in the Transformers version

The author turned the cache into a slot pool so the existing grouped matrix-multiply kernel (which runs all experts of a layer in one call) can run on the cached experts unchanged. A Python loop that ran one expert at a time was slow, while the slot pool restored speed. Metal kernels are compiled at runtime and published on the Hub; there is no C++ extension, and much of the code was built with AI help, per the author.

Numbers reported by the source

- On one 24 GB A10G with Qwen3.6-35B-A3B, decode speeds in tokens per second were: Strata Q8_0 82 to 88; llama.cpp Q8_0 with fit on 63 to 64; vLLM FP8 with expert-only offload 20.1; Transformers with Accelerate 0.16.
- On a 32 GB Mac, the author reports Qwen3.8-Flash-Next at about 14 tokens per second with this work, which he describes as close to Strata (within about 6% on token generation) and still too slow for his own coding-agent use.
- The generic implementation is proposed as a Transformers pull request, and benchmarks are in a Hugging Face Space.

Limitations and open questions

- One author’s hardware and settings; results on other machines will differ.
- The remaining gap is limited by per-layer synchronisation that PyTorch does not expose, per the author.
- The pull request was feedback-stage at the time of the post.

How to try or apply it

1. Check if your model fits in RAM: if so, llama.cpp with experts on the CPU may be enough.
2. If not, try an expert-streaming engine or the Transformers pull request.
3. Measure tokens per second for your own prompts, not only decode peak.
4. Watch SSD wear and thermal behaviour on laptops.

What it means for local-AI builders

Memory capacity stops being the only limit, and storage bandwidth becomes the bottleneck. See local LLM vs cloud AI and hardware for local AI.

Frequently asked questions

- **What is a mixture-of-experts model?** A model with many expert sub-networks where a router picks a few per token.
- **Is this fast enough for coding agents?** The author says about 14 tokens per second is too slow for his own use.
- **Do I need a Mac?** No. The comparison table was run on a single NVIDIA A10G.

Related reading

- llama.cpp v1.0.0 explained
- LLM quantisation explained: GGUF and AWQ
- Hardware for local AI
- Local LLM Practitioner

Sources and currency

Sources were read on 23 November 2026. Products, rules and standards change, so check the linked originals before you rely on any detail.

> Related credential: Local LLM Practitioner. Explore the exam facts and syllabus.

Sources and further reading

1. tarekziade. 180B Parameters, 32GB of RAM: Pushing MoE Inference to the Limit with Transformers. Hugging Face community blog, 18 November 2026. huggingface.co/blog/moe-expert-streaming.

Published by IICTP, an independent technical credential body, on 23 November 2026. The current version of this post is at https://blogs.iictp.org/moe-expert-streaming-transformers-run-180b-model-32gb-explained. More guides: blogs.iictp.org.