ACCELERATE **CHICAGO**

Craft software with confidence in 2026

Seizing the Means of Computation with Local AI

Jeremy Kahn — Accelerate Chicago 2026

Why local AI matters

(Audience participation time!)

claude.ai — Major Outage



90 days ago — 99.11 % uptime — Today

Claude Console (platform.claude.com) — Major Outage



90 days ago — 99.56 % uptime — Today

Claude API (api.anthropic.com) — Major Outage



90 days ago — 99.4 % uptime — Today

Claude Code — Major Outage



90 days ago — 99.26 % uptime — Today

Claude Cowork — Major Outage



90 days ago — 99.51 % uptime — Today

We’ve suspended access to Claude Mythos 5 and Claude Fable 5

Subscribe

Monitoring - Learn more here: https://anthropic.com/news/fable-mythos-access

Jun 13, 2026 - 00:50 UTC

Cloud-based AI

- Smarter models

- (Increasing) API costs
- Usage limiting
- Questionable uptime
- External dependency
- **No privacy**

Local AI

- Full privacy
- Infinite tokens
- 100% uptime
- No usage limits
- Full independence
- FREE 🤑

Setting expectations

Don't expect "Claude on your laptop"

Local AI is good for **common knowledge**

Local AI is **not** the best choice for agentic coding

But it will still work!

Local AI is GREAT for **automation**!

This is the killer app!

Use Cloud-based AI for:

- Agentic coding
- Fast inference
- Maximum intelligence

Consider Local AI for:

- Batch jobs
- Automation
- Sensitive information processing

Compute options

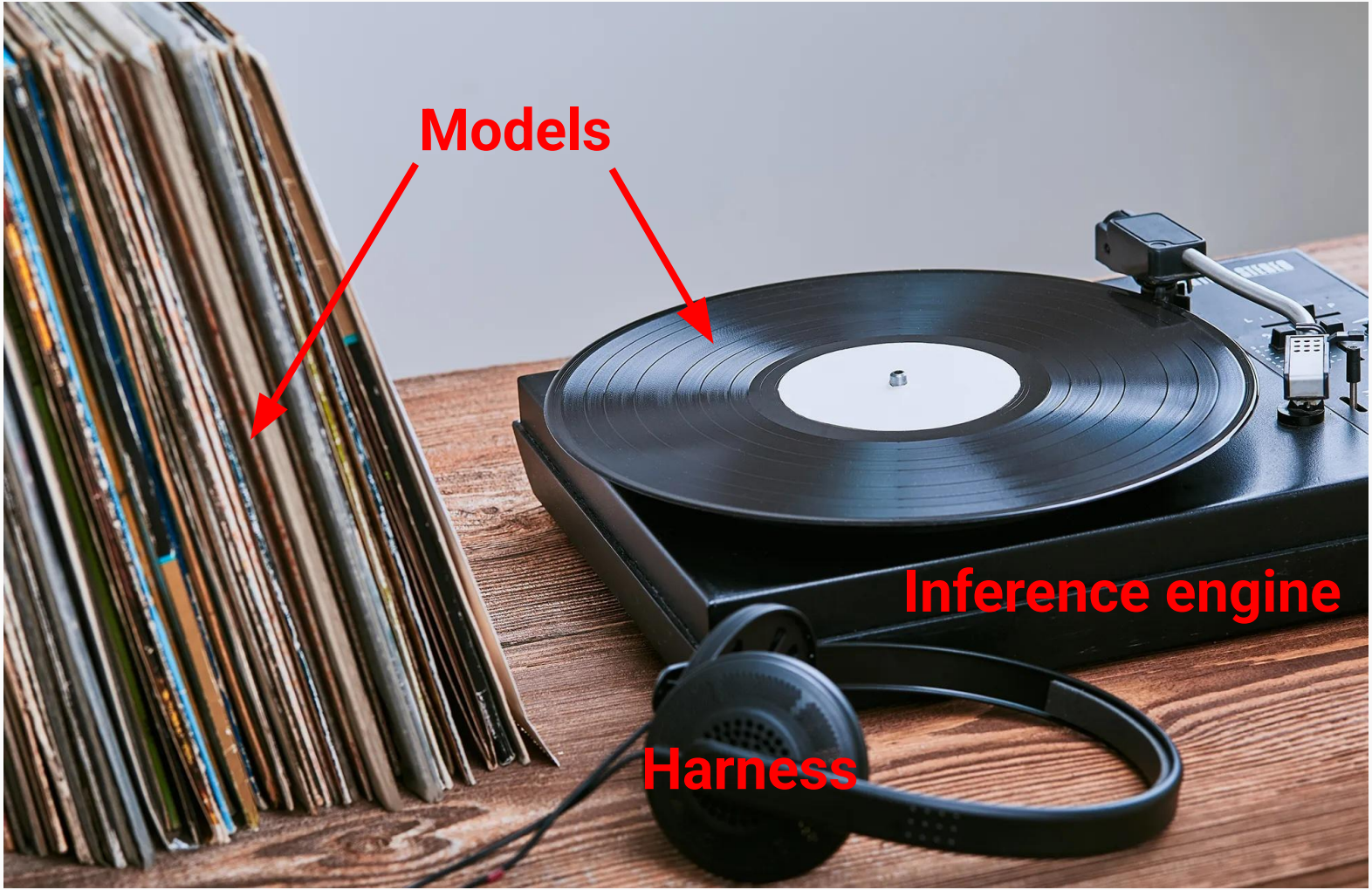
- Nvidia
- AMD
- Apple Silicon
- CPU

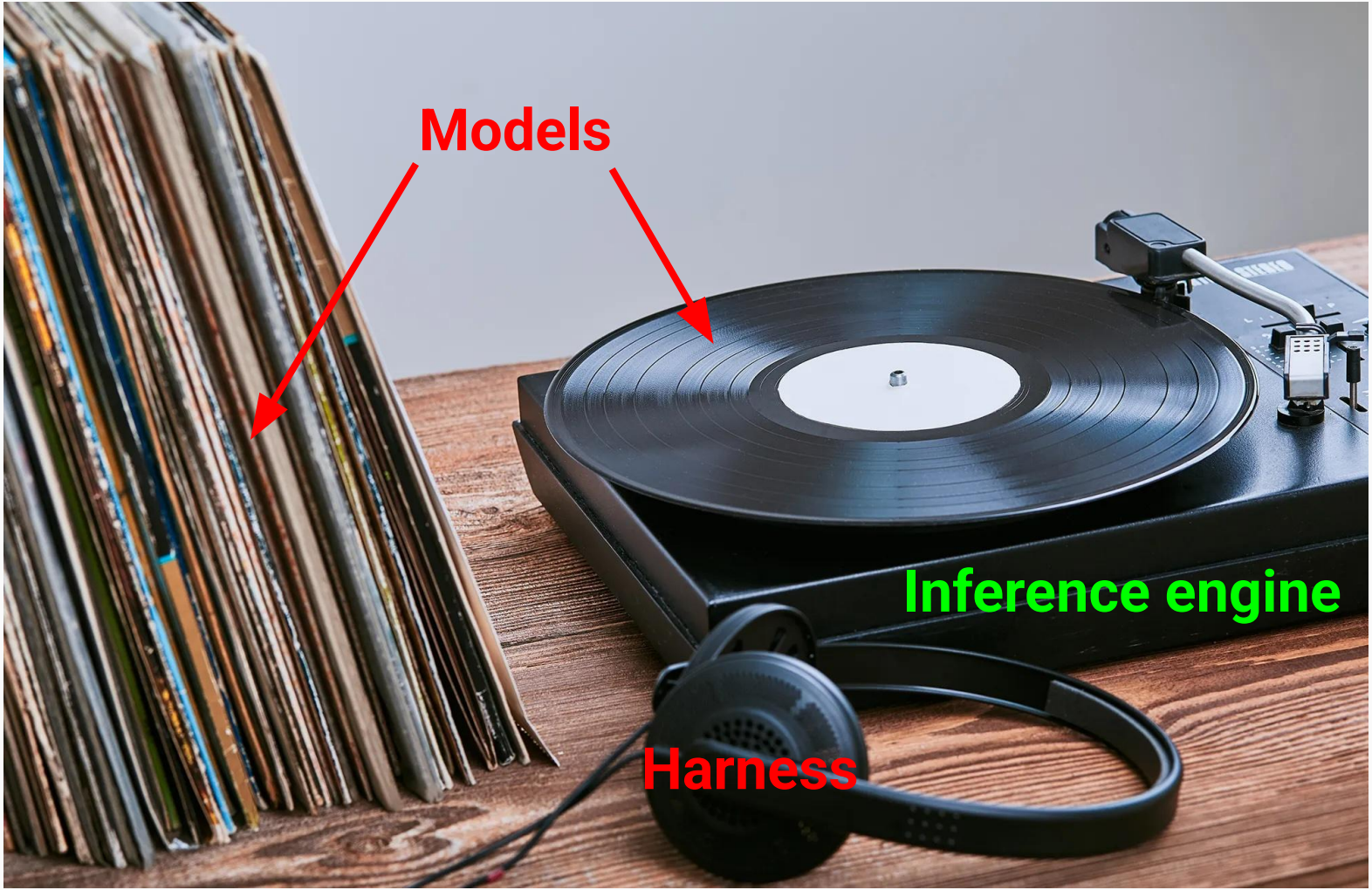
Now is **not** the time to buy hardware.

The best hardware for local AI is what you already have.



https://www.wired.com/story/where-to-shop-for-vinyl-records-online/


Models
Inference engine
Harness

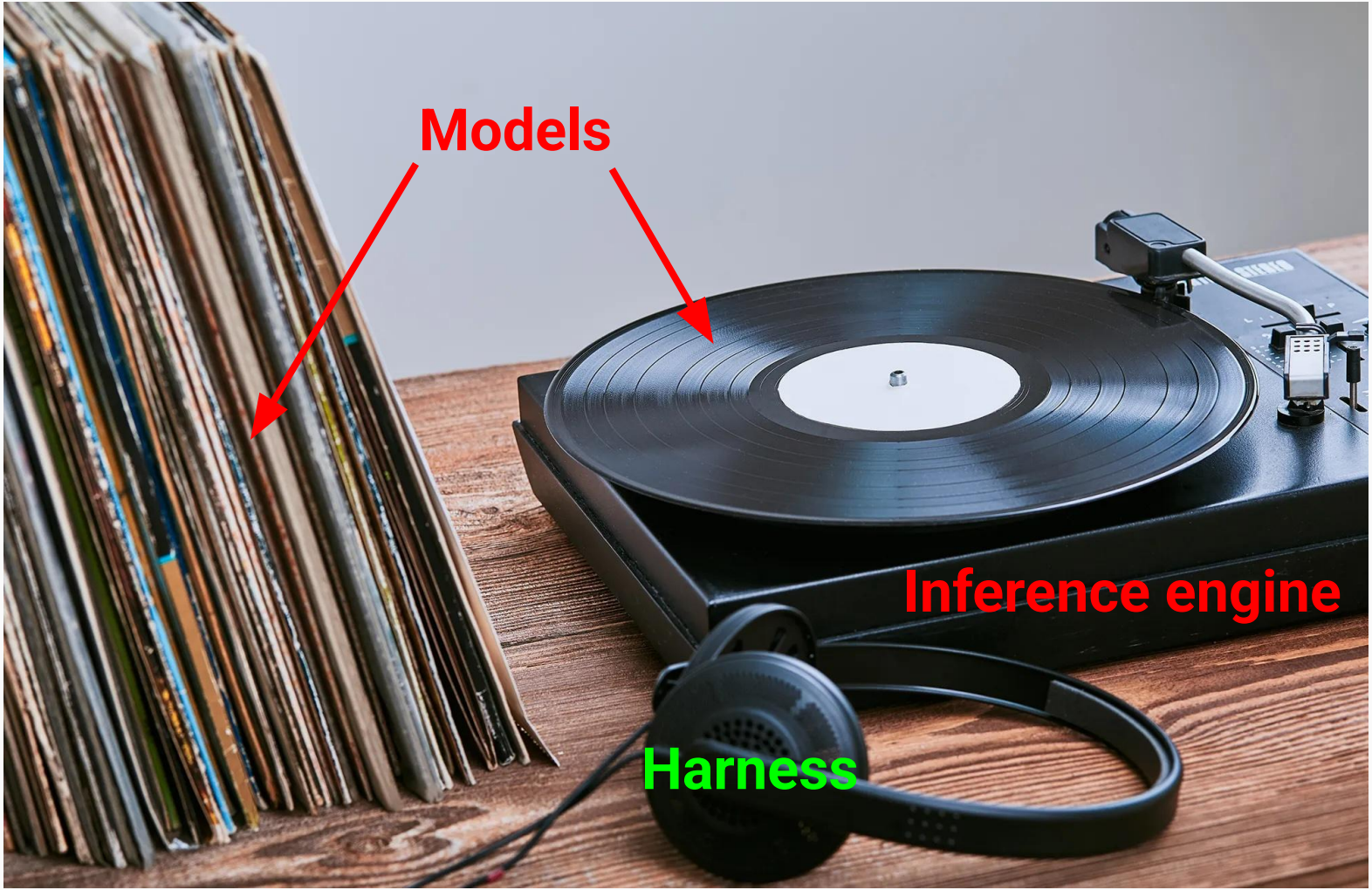

Models
Inference engine
Harness

Inference engines

- Llama.cpp ⭐
  - Format: GGUF
  - Use Vulkan variants for speed 🏃
- vLLM
- mlx-lm (⚠️ Apple only)
  - Format: MLX

Inference UIs

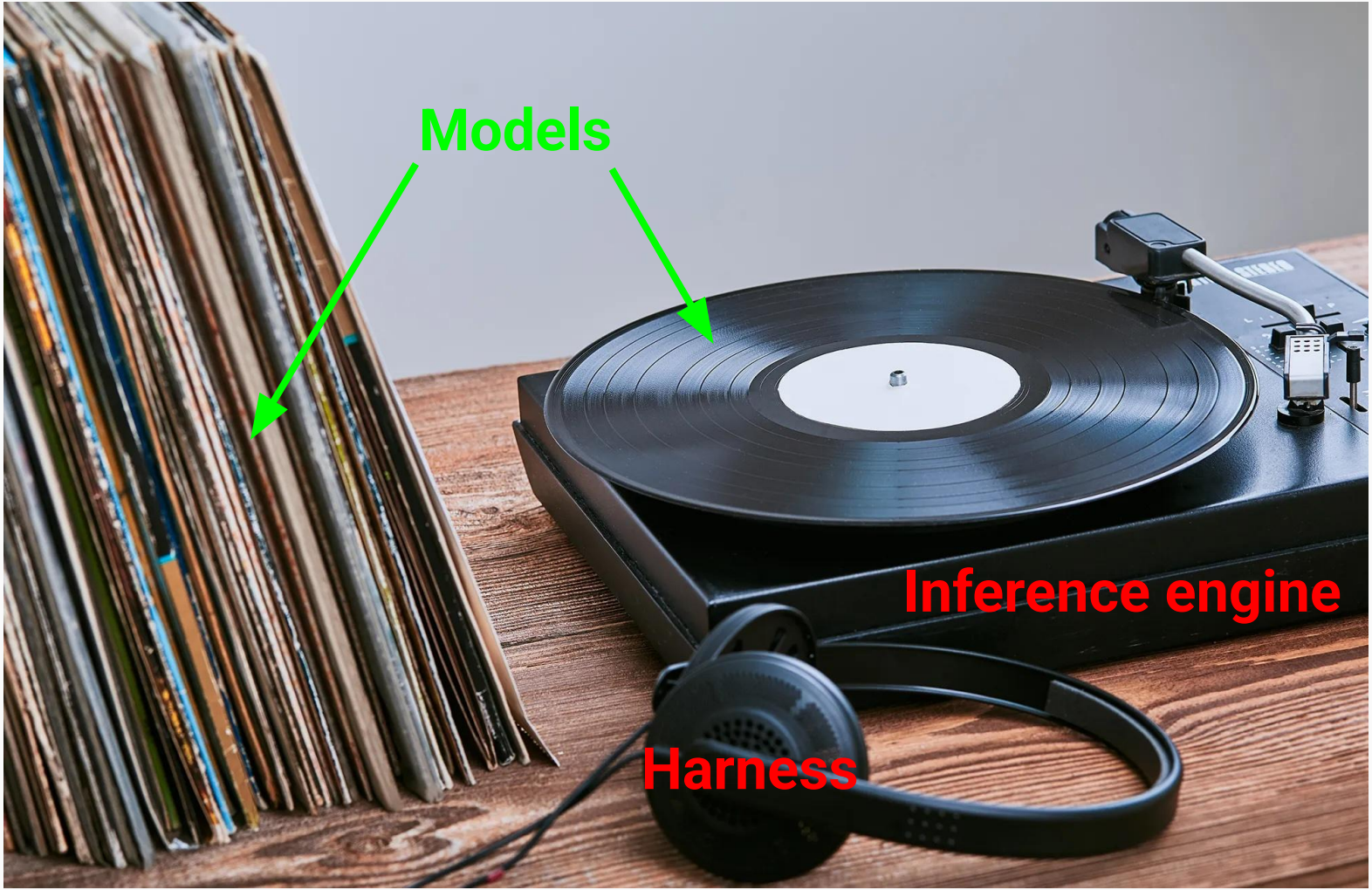
- Native apps
  - **LM Studio**
  - Jan
- Web front ends
  - OpenWebUI


Models
Inference engine
Harness

Agent harnesses

- OpenCode
- Kilo Code
- Codex
- Pi

The model landscape


Models
Inference engine
Harness



https://huggingface.co

American models

- Gemma (Google) 🥇
- GPT-OSS (OpenAI) 🥇
- Nemotron (NVIDIA) 🥈
- Llama (Meta) 🪦

Chinese models

- Qwen 🥇💻
- GLM 🥇
- Kimi 🥇
- Deepseek 🥇
- Minimax 🥇



Jargon alert



Minding our B's and Q's

Parameters

- Definition: (Roughly) How many weights a model has
- M = million, B = billion (3B = 3 billion), T = trillion
- Memory implications: At Q4 quantization, 1B of parameters consumes ~1GB of memory at low context
- More parameters == **Smarter** but **slower**
- Fewer parameters == **Dumber** but **faster**

Context window

- How much short-term memory an LLM has
- Measured in tokens
- More context == Higher quality output, but more memory usage
- Default settings are typically very low (4k, or 4096 tokens)
- My recommended baseline settings:
  - **32k** for chat
  - **64k** for agentic workflows (ideally 128k+)

LLM Architectures

- Dense
  - **All** parameters activated per token generated
  - Smarter but slower relative to memory usage
- Mixture of Experts (MoE)
  - **Subset** of parameters activated per token generated
  - Dumber but faster relative to memory usage

Quantization

"Quantization is the process of reducing the precision of a digital signal, typically from a higher-precision format to a lower-precision format"

- https://www.ibm.com/think/topics/quantization

Quantization: A visual explanation

Original Uncompressed



Low Quantization\n(High Quality: 95%)



High Quantization\n(Low Quality: 5%)



Quantization

- Important quantization (quant) levels:
  - Q8 (8-bit integer quantization) — *highest quality/size you'd ever need locally*
  - ⭐ Q4 (4-bit integer quantization) — **sweet spot** mix of quality/speed/memory (especially Q4_K_M)

Quantization



Choosing the right model

- Optimize for your use case
- Rightsize for your hardware
- **Experiment, experiment, experiment** with models and quants

Understanding Performance

Sometimes LLM "performance" is used to refer to **processing speed**, sometimes **output quality**.

We'll use it for the former here.

Understanding Performance

- Tokens per second ("tk/s")
  - Prompt processing ("pp")
  - Token generation ("tg")
- **KV caching**: Speeds up token generation
- **Prompt caching**: Speeds up conversation threads and agentic workflows



This is all… a lot



Where should I begin?

A recommended local AI starter kit

- App: LM Studio
  - lmstudio.ai
- Model: Gemma 4 E4B
  - Quantization: Q4_K_M

Demo time!

(Finally!)

Demo: Local chat bot

- Model: Gemma 4 E4B
- Harness: LM Studio

Demo: Local Cursor alternative

- Model: Gemma 4 E4B
- Harness: VS Code + Kilo Code

Demo: Local Claude Code alternative

- Model: Gemma 4 E4B
- Harness: OpenCode

The local AI community (where to learn more)

- r/localLLaMA
- AI model YouTuber: Bijan Bowen
- AI hardware YouTuber: Alex Ziskind

Gaps that remain

- Intelligence gap between local and SOTA models
- **Hardware cost**
- Approachability

The future of local AI

- More capable models
- Mainstream acceptance and usage
  - Probably once Apple ships their version 🙃
- Affordable hardware

Thanks for listening!

Questions?

github.com/jeremyckahn

PLEASE RATE MY SESSION