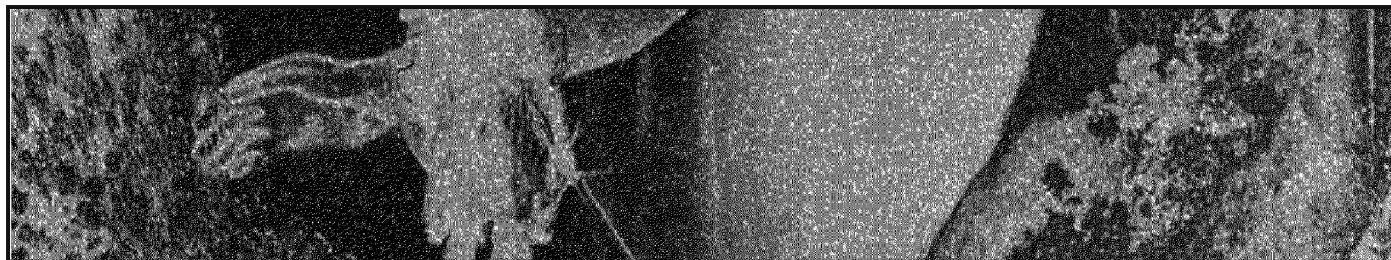


Own Your LLM



SPEAKER

Heili Aavola, Junior Research Fellow in Machine Learning, Institute of Computer Science, University of Tartu

PROJECT HERMES

Built with Martin Leissoo. Supervisors Mari-Liis Allikivi and Ardi Tampuu. Hermes Agent by Nous Research.

WHERE AND WHEN

Digit 2026, Friday 9 October, 14:30. Workshop 1: Own Your LLM, Lessons from Building a Local AI Agent.

01 / RULES OF THUMB

Rules of thumb

Four pieces of arithmetic decide whether a model runs on your hardware. All sizes are approximate.

Weights. GB of weights = parameters in billions $\times$ bytes per parameter. FP16 is 2 bytes, INT8 or FP8 is 1, 4-bit is about 0.55 once the scale metadata is counted. A 35B model at 4-bit is about 19 GB. A 320B model at 4-bit is about 176 GB.

Model names. 35B-A3B means 35 billion parameters stored and about 3 billion active per token. Memory cares about the first number. Speed cares about the second.

KV cache. Bytes per token = $2 \times \text{layers} \times \text{KV heads} \times \text{head dimension} \times \text{bytes per value}$. Llama 3 70B with an FP16 cache is about 320 KB per token, so a 100,000-token conversation holds about 32 GB of cache on top of the weights. An 8-bit cache halves that.

Decode ceiling. Tokens per second $\leq$ memory bandwidth $\div$ bytes read per token. 3B active parameters at 4-bit is about 1.7 GB per token. At 273 GB/s (DGX Spark) the ceiling is about 160 tokens/s. At 819 GB/s (M3 Ultra) it is about 480. Real throughput is lower.

Prefill wait. Time to first token $\approx$ tokens already in context $\div$ prefill speed. 100,000 tokens at 500 tokens/s is 200 s. At 3,000 tokens/s it is 33 s. Prefill is compute-bound, decode is bandwidth-bound. Big bandwidth and little compute (older Macs) means fast typing and a long wait on long chats. More compute and little bandwidth (DGX Spark) means the reverse.

Budget. Weights + cache at your longest expected context + every other model you run beside it (speech-to-text, text-to-speech, embeddings) + headroom. Fitting the weights on an empty box is not the same as fitting in use.

Hardware

Memory decides what fits. Bandwidth decides decode speed. Compute decides prefill speed. No single machine wins all three. Figures are representative published specifications; check the current generation before buying.

Machine	Memory for the model	Bandwidth	Compute	Fits at 4-bit, roughly
Gaming GPU, RTX 4090	24 GB GDDR6X	about 1,000 GB/s	High	Dense up to about 27B
Gaming GPU, RTX 5090	32 GB GDDR7	about 1,800 GB/s	High	Dense up to about 35B
Apple M4 Max laptop	up to 128 GB unified	546 GB/s	Moderate	MoE up to about 200B
Apple M3 Ultra desktop	up to 512 GB unified	819 GB/s	Moderate	MoE up to about 800B
NVIDIA DGX Spark	128 GB unified	273 GB/s	High, 1 PFLOP FP4 sparse	MoE up to about 200B
2 × DGX Spark over ConnectX-7	256 GB unified	273 GB/s each	High	GLM-5.3-Flash, 320B

Which model on which memory. Discrete GPU: a dense model that fits is fast, a large MoE rarely fits. Unified memory: a large MoE is the sweet spot, because each token reads only the active experts, so the low bandwidth costs little. A dense model on unified memory fits and crawls.

DGX Spark in one line. GB10 Grace Blackwell, 20 Arm cores, 128 GB LPDDR5x at 273 GB/s, 1 PFLOP FP4 sparse, 4 TB NVMe, 240 W, ConnectX-7 at 200 Gb/s to pair two units, \$3,999 list. Source: [NVIDIA DGX Spark hardware overview](#).

Serving software and harnesses

Every server below exposes an OpenAI-compatible endpoint. The harness does not know which one is behind it, so you can swap models and servers without touching the agent or the UI.

Tool	What it is	Verdict from the project
llama.cpp	GGUF models on CPU, GPU and Apple silicon	The most flexible. Start here on a laptop.
vLLM	Production server with continuous batching and paged attention	Served the first two generations. Built for GPUs and many concurrent users.
TensorFold	EXL3 checkpoints, tensor parallel across machines	Current server, across two Sparks. See the checkpoint link in the stack section.
Ollama	One-line installs, a wrapper around llama.cpp	Easiest first run. Slow defaults. Try with it, do not serve with it.
LM Studio	Desktop app over llama.cpp, and MLX on a Mac	Good on a laptop when you want a window rather than a terminal.

Quantisation formats follow servers. GGUF for llama.cpp. AWQ, GPTQ and FP8 for vLLM. EXL3 for TensorFold. Download the format your server reads. Benchmark the quantised checkpoint you will run, not the paper. Small models lose the most at 4-bit.

Harnesses that drive a local model.

- [Hermes Agent](#), Nous Research. Tools, memory, skills and a gateway API. The one Project Hermes runs.
- OpenCode. A coding agent in the terminal that also fronts hosted models, so you can compare.
- Pi. Bare bones, you configure everything. OpenClaw was built on it. Good for learning what a harness is.

Reaching the box from anywhere. Install Tailscale on both machines and sign in with the same account. SSH to the box by its tailnet name. Serve the model on a port. Point any client at `http://box:8001/v1`. Tailscale Serve can expose a UI to the tailnet only, which is how the testers reached the Command Center.

The Project Hermes stack

Three generations in five months. The harness (Hermes Agent), the UI (Hermes Command Center, Next.js and Bun), Docker and Tailscale did not change between them. Parameter counts say nothing about speed or answer quality.

When	Model	Total / active	Server	Hardware
May 2026	Qwen3.6-35B-A3B , INT4	35B / 3B	vLLM	1 × DGX Spark
September 2026	DeepSeek-V4-Flash-Vision-Exp (V4-Flash family)	284B / 13B	vLLM	2 × DGX Spark
8 October 2026	GLM-5.3-Flash , EXL3 4bpw checkpoint for TensorFold	320B / 18B	TensorFold v0.6.0	2 × DGX Spark

Current configuration. TensorFold v0.6.0 in Docker, tensor parallelism across two ranks, one memory pool per host. Server context limit 1,048,576 tokens, Hermes configured for 1,000,000. EXL3 4-bit routed experts, 4-bit dense components, FP8 conversation cache. Speculative decoding with the [DFlash2 draft model](#).

Not entirely offline. Search and extraction use Tavily. Text-to-speech uses Edge. A hosted fallback, MiniMax-M2.5 through OpenCode Zen, takes over when the local model is down. Know which calls leave the building and decide which you can live without.

What the testers said. Round one, seven testers, 29 May to 2 June 2026, on the Qwen generation: 3 of 7 named speed as the biggest problem, 2 of 7 named output quality, 5 of 7 rated the agent similar to the frontier tools they use, 6 of 7 would use it again. Round two, one tester, September 2026, on the DeepSeek generation: much better than frontier tools, nothing to improve. One data point, recorded as such.

Should you self-host? Hard requirement on privacy, retention or offline use: yes. Then match the model to the memory you have. If a 3B to 20B-active open model is good enough for the task, measured on ten real tasks rather than assumed, run it locally. If not, run hybrid: local by default, hosted fallback for the hard ones.

Where this knowledge lives

Most of what is on this sheet was learned from other people running the same hardware and the same models. These are the places to read, and to ask when something breaks.

If you have a DGX Spark, start here.

- [DGX Spark developer forum](#): other people optimising for the same GB10 chip. Container recipes, driver and CUDA issues, aarch64 builds, two-Spark setups. Search before you post, your problem has usually been hit already.
- [DGX Spark playbooks](#), source on [GitHub](#): NVIDIA's step-by-step guides. Directly relevant to this talk: Run Hermes Agent with Local Models, Connect Two Sparks, NCCL for Two Sparks, vLLM for Inference, Run models with llama.cpp on DGX Spark, Speculative Decoding, NVFP4 Quantization, Set up Tailscale on Your Spark.

Communities for local models in general.

- [r/LocalLLaMA](#): the main one. New open-weight releases, quantised checkpoints, tokens-per-second reports per hardware, serving tips. If a model runs badly on your machine, someone has posted about it here.
- [r/LocalLLM](#) and [r/ollama](#): beginner-friendly, laptop and single-GPU questions.
- [r/MachineLearning](#): papers and research discussion rather than hardware.

- GitHub issues and discussions on [llama.cpp](#), [vLLM](#) and [Hermes Agent](#): the fastest route to a fix for a serving or harness bug, and where new hardware support lands first.
- Discord servers run by Nous Research, Hugging Face and the llama.cpp community: live help, release chatter.

Models and quantised checkpoints.

- [Hugging Face](#): the model hub. Filter by task, size and format (GGUF, AWQ, EXL3). Read the model card and the Discussions tab of a checkpoint before downloading: that is where quality reports for a specific quantisation appear. Well-known publishers of quantised files include bartowski and unsloth.
- The checkpoints behind this talk are linked in the stack section above.

Benchmarks, and how to read them.

- [LMArena](#): human preference votes between models, open and closed.
- [Artificial Analysis](#): quality, speed and price side by side, including open models on hosted providers.
- [LiveBench](#): contamination-resistant questions refreshed over time.
- [Aider polyglot leaderboard](#) and [SWE-bench](#): coding and agentic tasks, the closest proxies for what a harness will do.
- Leaderboards score the full-precision model on someone else's hardware. Your 4-bit checkpoint on your bandwidth is a different system. Measure it: llama-bench ships with llama.cpp, vLLM has vllm bench, and ten of your own real tasks beat any of them.

06 / LINKS AND CONTACT

Links and contact

- Hermes Agent: <https://github.com/NousResearch/hermes-agent>
- GLM-5.3-Flash base model: <https://huggingface.co/zai-org/GLM-5.3-Flash>
- GLM-5.3-Flash EXL3 4bpw for TensorFold: <https://huggingface.co/Mia-AiLab/GLM-5.3-Flash-EXL3-4bpw-TensorFold>
- DFlash2 draft model: <https://huggingface.co/incoai/GLM-5.3-Flash-DFlash2>
- DeepSeek-V4-Flash: <https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash>
- Qwen3.6-35B-A3B: <https://huggingface.co/Qwen/Qwen3.6-35B-A3B>
- DeepSeek-V3 technical report, the reference for MoE sizing: <https://arxiv.org/abs/2412.19437>
- NVIDIA DGX Spark hardware overview: <https://docs.nvidia.com/dgx/dgx-spark/hardware.html>
- llama.cpp: <https://github.com/ggml-org/llama.cpp>
- vLLM: <https://github.com/vllm-project/vllm>
- Tailscale: <https://tailscale.com>

Heili Aavola, Junior Research Fellow in Machine Learning, Institute of Computer Science, University of Tartu. heili.aavola@ut.ee

Martin Leissoo, co-author of Project Hermes. martin.leissoo@ut.ee

Supervisors: Mari-Liis Allikivi and Ardi Tampuu. Developed during the [Project in Applied Artificial Intelligence \(LTAT.02.039\)](#) course.

Workshop: Own Your LLM, Lessons from Building a Local AI Agent. Digit 2026, Friday 9 October, 14:30.