LOCAL RAG PROJECT / INFERENCE EXPERIMENT

Local AI Cluster Benchmark

A Mac M1 baseline compared with Windows and Android workers using llama.cpp RPC.

22.37 tokens/s

Mac + Windows

9.0% higher reported throughput than Mac alone. The request finished 0.33 s sooner.

Three configurations, one workload

Configuration	Split	First token	Total time	Reported tok/s
Mac only	Baseline	0.83 s	5.46 s	20.53
Mac + Windows	**1:1**	**0.88 s**	**5.13 s**	**22.37**
Mac + Windows + Realme	2:2:1	0.80 s	5.12 s	21.96

Reported generation throughput



The third device improved first-token time in this run

The Realme configuration reached 0.80 s to first token, 0.03 s ahead of Mac-only. It generated 0.41 fewer tokens/s than Mac + Windows, and the phone’s RPC endpoint became unreachable after the test.

Method and next steps

What this first run measured and how to validate the result on a real RAG workload.

Test environment

Mac M1 coordinator, Intel Core i7 Windows worker and optional Realme Android worker via Termux. The Mac and Windows nodes used a direct Ethernet link reported at approximately 93 Mbps. The project was developed with Cursor. Qdrant and Ollama are part of the local RAG project. This suite timed the llama.cpp RPC inference path.

Workload and metrics

Llama 3.1 8B Instruct (GGUF Q4_K_M), 512-token context, four threads and one request at a time. All modes streamed the same storage replication prompt at temperature 0 with a 48-token output cap. First-token time measures the delay to the first streamed content. Total time measures wall time to stream completion.

How to read the result

Mac + Windows is the current demo profile based on this session. The source reports decode throughput separately from total request time and does not include raw token counts or decode timestamps to verify the rate calculation. One reported result per mode leaves run-to-run variation unknown. The suite did not measure Qdrant retrieval, Ollama inference, or answer quality.

What I would test next

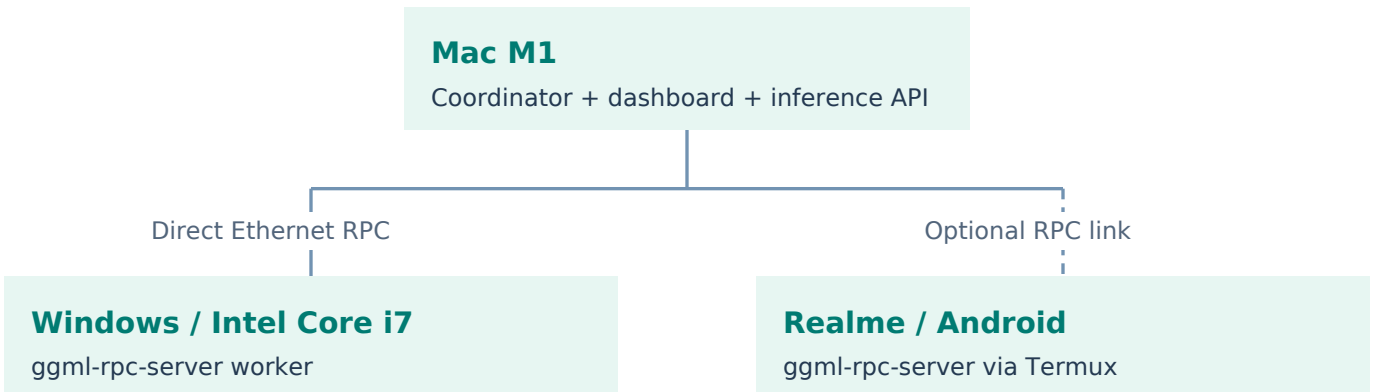
Repeat. Run warm and cold trials, log token counts and decode timestamps, and report p50 and p95 latency.
Profile. Track resource use, network traffic and worker uptime across repeated requests.
Measure the full pipeline. Time embedding, Qdrant retrieval and Ollama generation separately. Check answer grounding on a fixed set of document questions.

Source / Operator benchmark comparison, 25 November 2026

TECHNICAL APPENDIX / A1

Test environment and topology

Node roles, network paths and runtime configuration for the inference benchmark.



Mac only uses the coordinator alone. **Mac + Windows** enables the Ethernet worker. **Mac + Windows + Realme** enables both workers.

Node addresses and interfaces

Host / link	Network	Ports / notes
Mac (M1)	Private network	Dashboard and inference API
Windows (i7)	Private network	RPC worker
Realme Android	Private network	RPC worker via Termux
Mac–Win Ethernet	Private network	100 Mbps class · ~93 Mbps measured

Software stack

Model Meta-Llama-3.1-8B-Instruct-Q4_K_M.gguf
Runtime Dockerized llama.cpp coordinator and Windows worker. The optional Android worker runs in Termux.
llama.cpp commit 8d60873dfb8619f2b5ed31adfbab07cccb40db5c
RPC coordinator flags -ngl 99, --rpc, --tensor-split, --fit off, --no-kv-offload, --no-warmup
Settings CTX_SIZE=512, LLAMA_THREADS=4, parallel=1
Workers Windows uses compose.worker.yaml on the private Ethernet link. Android is optional and enabled with PHONE_RPC_ENABLED=1.

Methodology and workload

Request settings, timing definitions and conditions shared across the three modes.

Benchmark procedure

Start each mode from the dashboard and wait for **Ready**. Run the built-in streamed chat benchmark against the coordinator API. Monitor the phone·s RPC port during and after the test.

Coordinator chat completions endpoint

Benchmark request

Prompt “Explain the difference between synchronous and asynchronous storage replication in 80 words.”
Settings stream=true, temperature=0, max_tokens=48 (BENCH_MAX_TOKENS), include_usage enabled.

Timing definitions

First token Time to the first streamed content delta.
Total request time Wall time until the [DONE] event.
Reported tokens/s Completion tokens divided by the decode window, as described by the benchmark. Raw token counts and decode timestamps were unavailable for verification.

Comparison conditions

All modes used the same GGUF file, prompt, output cap and context size. The Mac-only baseline used no RPC worker. Mac + Windows used a 1:1 tensor split. Mac + Windows + Realme used 2:2:1. These are configured split weights and do not represent measured compute throughput (FLOPs).

The three-device setup recorded 0.80 s to first token. The phone’s RPC endpoint became unreachable after the run.