

LUCIEN LUSCIEN

HETRA RESEARCH INITIATIVE

HETRA: Distributed Inference and Low-Communication Training on Consumer GPUs

THE PEOPLE ARE THE INFRASTRUCTURE.

Instead of a future where a handful of enormous companies own nearly all the computing infrastructure, why couldn't millions of ordinary people own a meaningful part of it and sell the output back to those same companies?

Lucien Luscien · hello@lluscien.com · lluscien.com

Preprint v1.0 · November 2026 · Not peer reviewed

HETRA: Distributed Inference and Low-Communication Training on Consumer GPUs

Lucien Luscien · HETRA Research Initiative · hello@lluscien.com · Preprint v1.0, November 2026

ABSTRACT

We present HETRA (Heterogeneous Execution, Training & Resource Aggregation), a system for coordinating heterogeneous consumer GPUs to perform distributed language-model inference and collaborative training. On a testbed of four 12 GB consumer GPUs (two RTX 5070, two RTX 3060) in two desktops, with every inter-home link passed through an emulator of residential internet (up to 150 ms round trip and down to 20 Mbps), we report five results. **(1) Exactness.** Layer-split inference of a 27B model across two to four cards produced text identical to the unsplit model in every one of 2,040 graded requests across all link profiles, and a 70B model split across all four cards matched a CPU-only reference on 40/40 short answers. Network delay never changed an output. Departures from exact text arose only from changes in numeric settings (speculation, continuous batching, CPU/GPU placement), which we quantify. **(2) Capacity.** Independent groups of homes scale throughput linearly (2.05× on a LAN, 2.02× at 150 ms). A speed-aware router serves mixed hardware near the sum of its parts (28,800 requests, all correct). **(3) Verification and payment.** In a 9,000-request marketplace run with audits, a planted dishonest node was banned 1.1 s after joining, a node killed mid-run lost no answers, honest nodes had 0 mismatches in 896 audits, and the payout ledger reconciled exactly. **(4) Transport.** A command-level traffic census showed 10–15× more link traffic than the computation requires. Two transport changes cut model-graph traffic 40× and raised 70B throughput 37% at 150 ms. With speculative decoding, one user can chat with the pooled 70B at 8.5 tokens/s with homes 30 ms apart. **(5) Training.** With time-based rounds, work-weighted merging and overlapped communication, four mixed cards on 150 ms / 20 Mbps links reached a pre-registered quality target 1.15× faster than the fastest card alone, winning all three seed-matched pairs. Adding homes followed a fitted saturating power law: an out-of-sample forecast for 32 homes, made before the run, landed within 0.014. A 124M model gained 1.7× as much from added homes as a 30M model did, and a 124M model trained from scratch, mostly over emulated home links, reached a validation loss of 3.48 against 3.23 for GPT-2 small on the same text. All experiments ran in one house with emulated links.

1. Introduction

Demand for AI computing is growing faster than data centers can be permitted, powered and built. In the November 3, 2026 U.S. elections, voters favored limiting data-center development in 40 of 41 local ballot measures (Ballotpedia, 2026). Meanwhile a large installed base of consumer GPUs already has power, cooling and an internet connection, and is idle most hours of the day.

HETRA is built for a household-owned compute network: homes contribute idle hardware, the network sells verified AI work, and owners are paid for the work their machines perform. Whether such a network is worthwhile is first a technical question with three parts:

1. **Correctness.** Can a model too large for any one home be split across homes and still give the answers it would give on one machine? Exact equality also makes verification cheap: checking a node becomes a string comparison.
2. **Capacity and usability.** Does adding homes add capacity, and is a pooled model fast enough to use over home-grade links?
3. **Training.** Can heterogeneous consumer cards train one model together over slow links and reach a given quality *faster* than the best single card among them? If not, pooling is a cost rather than a gain.

Contributions.

1. A reproducible household testbed with one residential-link emulator per home, independent grading, and pre-registered, time-stamped pass/fail criteria (Appendix B). Every run, including 30+ failed or aborted runs, is listed (Appendix C).
2. Evidence that cross-home layer splitting is output-exact on mixed GPU generations under all tested link conditions, and a quantified account of which numeric settings break exactness (§4.1).
3. A characterization of where pooled-inference time and traffic go, the removal of the largest avoidable costs (§4.2–4.4, §4.8), and a pre-registered usability test (§4.9).
4. A working verification-and-payment loop: audits, banning, dropout recovery and an exact ledger (§4.10).
5. A training configuration under which four heterogeneous household cards over 150 ms / 20 Mbps links beat the fastest card to a fixed quality on all pre-registered seeds (§5.3), plus a scaling law validated out of sample (§5.4) and a GPT-2-sized model trained from scratch (§5.6).

2. Related work

Volunteer and rented consumer compute. BOINC (Anderson, 2004) showed large-scale volunteer computing for independent tasks. Commercial marketplaces rent whole consumer GPUs to single customers; they do not pool several homes into one model instance. **Model-parallel inference over the internet.** Petals (Borzunov et al., 2023) serves and fine-tunes 100B+ models by pipelining layers across volunteer GPUs. llama.cpp's RPC backend offloads layers to remote devices but is designed for LANs; we use it as the off-the-shelf baseline. **Training over slow or unreliable links.** SWARM parallelism (Ryabinin et al., 2023) trains with pipeline stages across heterogeneous, unreliable nodes. Federated learning (McMahan et al., 2017) trains across devices that keep their data local. DiLoCo (Douillard et al., 2023) synchronizes separate clusters only every few hundred steps, and INTELLECT-1 (Prime Intellect, 2024) trained a 10B model across continents on data-center GPUs. Gradient compression (Lin et al., 2018) reduces the bandwidth of each exchange. **Latency hiding.** Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) drafts several tokens and verifies them in one pass of the large model.

Work	Hardware	Network	Workload	Distinctive element
BOINC	Volunteer PCs	Internet	Independent tasks	Scale; redundancy-based validation
Petals	Volunteer GPUs	Internet	Inference, fine-tuning	Layer pipelining of 100B+ models
SWARM	Heterogeneous, preemptible	Internet-like	Training	Fault-tolerant pipeline stages
Federated learning	Phones, devices	Mobile	Training	Data stays on device
DiLoCo / INTELLECT-1	Data-center GPUs	Inter-cluster WAN	Training	Rare synchronization
This work	Consumer GPUs, two generations	Emulated residential, per home	Inference and training	Exact-output checks, audits and ledger; time-based rounds for mixed cards

Table 1. Positioning. Descriptions of prior work are summaries of their stated settings.

3. Testbed and methods

3.1 Hardware and homes

Two desktop computers on one home network. Computer A: AMD Ryzen 9 5900XT, 48 GB RAM, two RTX 5070 (12 GB, Blackwell). Computer B: two RTX 3060 (12 GB, Ampere). Each GPU is one "home", pinned by device UUID. Computer B's homes reach Computer A through an encrypted SSH tunnel, so no home exposes a listening port. Experiments with more than four homes place several homes on one card; they measure learning per round, not wall time.

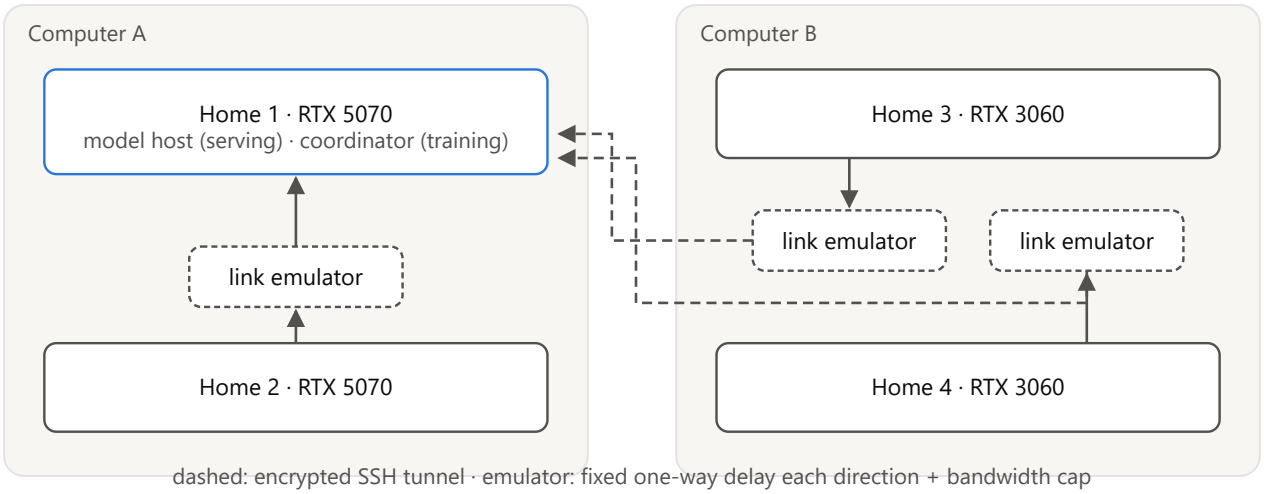


Figure 1. Testbed. Every remote home sits behind its own link emulator. For serving, home 1 hosts the model server and relays each stage (llama.cpp RPC is hub-and-spoke); for training, home 1's computer also runs the coordinator.

3.2 Residential-link emulation

A TCP relay in front of each remote home adds a fixed one-way delay in each direction and caps bandwidth. Its profile is re-read every second, so conditions change between runs without reloading models. Profiles: LAN (unshaped), 20, 30, 50 and 150 ms round trip at 100 Mbps, and 50 ms or 150 ms at 20 Mbps. A test of Nagle's algorithm on the relay's sockets showed no material effect (§4.3), so delay results are not relay artifacts. The emulation has no jitter, packet loss or real home routers.

3.3 Serving stack and grading

llama.cpp release b12207 (official binaries, and our own build verified to give identical answers; Appendix D). Models: a 27.3B hybrid model (Q4_K_M, 15.7 GB), Qwen3-8B (Q4_K_M), Qwen3.6-35B-A3B MoE (20.2 GB) and Llama 3.3 70B Instruct (Q3_K_M, 34.3 GB); model files were checked against their published SHA-256. The accuracy set is 40 fixed questions with known short answers (greedy: temperature 0, top-k 1, seed 7, up to 48 tokens, thinking off); the long-answer set is 10 fixed prompts × 128 tokens. A separate script grades each reply against accepted answers and compares exact text with a one-machine reference. Every request is logged with UTC times and input/output hashes.

3.4 Training algorithm

Our trainer belongs to the DiLoCo family, with three changes for heterogeneous homes: rounds of fixed *wall time* rather than fixed step count, merge weights proportional to the work behind each update, and communication overlapped with training.

Algorithm 1. Time-based, work-weighted low-communication training (per home h; coordinator)

```
Home h:
  θ ← shared weights from coordinator
  loop each round r:
      train on own data slice for T seconds with AdamW → local weights θ_h, s_h steps
      Δ_h ← int8(θ_h − θ)                      # one byte per parameter, per-tensor scales
      push (Δ_h, s_h); keep training while it is in flight      # overlap
      on merged update U: θ ← θ + U; rebase local weights onto θ
Coordinator, each round:
  collect updates until all active homes report or the timeout expires
  w_h ← s_h / Σ_k s_k                          # work-weighted
  U ← int8( η · Σ_h w_h Δ_h )                   # outer step: plain average, η = 1, momentum 0
  broadcast U; log per home: steps, bytes, ‖Δ_h‖, cos(Δ_h, U); validate shared model
```

Data: FineWeb-Edu (Penedo et al., 2024). The 30M experiments use a science subset (documents with ≥4 distinct terms from a fixed 57-term list: 46,246 of 569,000 documents in shard 000; 120M training and 2M held-out tokens; extended to 771M tokens for the scaling ladders). The 124M model uses unfiltered general text (3.75B training tokens; 2.0M held-out tokens from a separate shard, with science-validation documents excluded from training). Validation always uses the same 320 windows of 512 tokens. Reference: OpenAI GPT-2 small (124M), loaded from its published weights and scored on exactly the same windows and tokenizer (3.357 science, 3.229 general). Hyperparameters are in Appendix A.

3.5 Pre-registration and measurement integrity

Evaluation criteria for serving were written before any split-home result: correctness is a pass/fail gate; time-to-solution, throughput and latency are all reported; scale-out and fixed-job speed-up are both measured; small-test failures are retested before any negative conclusion. Each training claim was filed as a dated amendment before its first run, with an explicit pass rule (Appendix B). Other rules applied throughout: only same-session, alternating comparisons are used for engine claims (cross-hour numbers drifted up to 30%; Appendix E); every test port is guarded against collisions; tooling and data are hashed before use; and no run is dropped from the record.

4. Serving results

4.1 Exactness, and what breaks it

Test	Graded requests	Correct	Identical text to unsplit model
27B, 2 homes, LAN, matched server flags	40	40	40
27B, 2 homes, five link profiles (LAN to 150 ms; 20 Mbps)	600	600	600
27B, 2 homes, output layer on host, four profiles	480	480	480
27B, chains of 2, 3 and 4 homes, LAN and 150 ms	720	720	720
27B, two concurrent 2-home chains, LAN and 150 ms	160	160	160
27B, our build vs official build, LAN and 150 ms	40 + 320	360	360
70B, 4 homes, LAN and 150 ms (vs CPU-only reference)	80	80	80

Table 2. Exact-output results with fixed numeric settings (single sequence or fixed slots, no speculation). Text identity holds across Ampere and Blackwell cards, every link profile and both device orders.

Delay never changed an output: for every configuration tested at both LAN and 150 ms, the two outputs were identical. Exact text did change when *numeric settings* changed, which matters for verification design:

Setting change	Effect on text	Correctness
Speculative decoding (model's own MTP heads), ≤3 drafted tokens	9/10 long answers identical to no speculation	unchanged
Speculative decoding, ≤6 drafted tokens	7/10 identical	unchanged
Continuous batching, 4 slots (8B model)	98.8% identical over 28,800 requests ("7" vs "seven")	100%
MoE experts computed on CPU instead of GPU (35B MoE)	short 40/40 identical; long answers diverge after 105–589 chars	40/40
70B on GPUs vs CPU-only	short 40/40 identical; long 4/10 identical (diverge after 205–608 chars)	40/40

Table 3. Sources of non-identical text. In each case verifying a token batch, or computing on a different device, changes floating-point summation order and can flip near-tied token choices.

Implication: exact-match audits are valid when every node in a job uses the same build, batch shape, speculation setting and device placement, or when audit requests run with batch size 1; otherwise a normalized or tolerance-based comparison is needed (as used in §4.10). A methodological warning: with default server settings, the 2-home split first appeared to need 10.3 s per answer. Two caching features were moving the hybrid model's recurrent state across the link on every request; disabling them gave 0.30 s with identical outputs.

4.2 Link delay and output-layer placement

Single-stream generation speed fell with delay (27B, 2 homes: 14.1 tokens/s on LAN, 4.2 at 20 ms, 3.1 at 50 ms, 2.3 at 150 ms, 1.2 at 50 ms / 20 Mbps). Between 50 and 150 ms the marginal cost was ≈1.1 round trips per token, close to the floor for a two-stage split. Byte counters showed ≈3.3 MB returning per short question on the 20 Mbps link, far above the ≈20 KB per token the hand-off requires: the remote home held the output layer, so full-vocabulary logits crossed the link every token. Keeping the final layers on the requesting host:

Profile	Generation tokens/s: before → after	8-in-flight answers/min	Correct / identical
LAN	14.1 → 13.7	244.0 → 224.6	120/120 / 120/120
20 ms, 100 Mbps	4.2 → **7.1**	63.1 → 66.3	120/120 / 120/120
150 ms, 100 Mbps	2.3 → **3.2**	42.0 → 46.5	120/120 / 120/120
50 ms, 20 Mbps	1.2 → **2.7**	17.9 → 18.7	120/120 / 120/120

Table 4. Output layer on the requesting host: 1.4–2.3× faster generation under delay or low bandwidth.

Eight requests in flight gained little over one at a time under delay, because the stock RPC path runs stages in turn: only one home computes at once. Splitting decode batches into micro-batches made it worse (31.8 → 13.6 answers/min at 150 ms), since each micro-batch pays its own round trips and the stock path does not overlap them.

4.3 Where the link traffic goes

We enabled llama.cpp's per-command RPC trace at 150 ms with 8 requests in flight (40/40 correct and identical):

Command	Calls	Host → worker	Worker → host
Step inputs (SET_TENSOR)	972	127.8 MB	—
Full graph description (GRAPH_COMPUTE)	97	73.1 MB (736 KiB each)	—
Cached graph (GRAPH_RECOMPUTE)	12	≈0	—
Stage output (GET_TENSOR)	109	≈0	42.1 MB

Table 5. RPC command census. The graph cache hit only 11% of steps although the graph never changed size, because any change of batch shape creates a new graph identity.

Measured traffic was 10–15× the intrinsic stage hand-off. Three hypotheses were tested one at a time. (i) Nagle's algorithm in the relay: **rejected** (no material change). (ii) The per-step inputs are attention

masks: **rejected**, since input volume was identical at context 8,192 and 2,048 (112.8 MB both). Sizes clustered at multiples of 20.3 KB, one 32-bit vector of the 5,120-wide hidden state, so (iii) the inputs are token embeddings computed on the host: **confirmed**. Placing the embedding table on the first home cut that leg 66% (112.8 → 38.4 MB) and raised LAN throughput 16% (176.5 → 204.8 answers/min), with 40/40 identical. In the stock tool, this tensor override silently disables llama.cpp's pipeline parallelism, so the two optimizations conflict.

4.4 Self-drafting with the model's own prediction heads

Variant (2 homes, 10 × 128 tokens)	LAN tokens/s	150 ms tokens/s	150 ms total	Drafts accepted
No speculation	19.57	3.38	384.3 s	—
MTP heads, ≤3 drafted	34.13 (1.74×)	**7.72 (2.28×)**	**171.5 s**	57.8%
MTP heads, ≤6 drafted	29.41 (1.50×)	7.81 (2.31×)	169.4 s	35.1%

Table 6. Speculative decoding across two homes. On short answers (3–5 tokens) speculation was slower (1.64 vs ≈3.3 tokens/s), and with several parallel slots this hybrid model crashed on draft rollback, so self-drafting was usable only for single-stream long answers in this build.

4.5 Longer chains

Homes in chain	LAN ms/token	150 ms ms/token	150 ms answers/min (8 in flight)	Correct / identical
2	66	305	41.4	240/240
3	79	543	26.1	240/240
4	122	846	16.9	240/240

Table 7. Each added remote stage cost +238 ms and +303 ms per token at 150 ms (≈1.6–2.0 round trips), because the host relays every stage's input and output. In the 4-home chain one RTX 3060 held a smaller share of layers.

4.6 Capacity: replicas and routing

Two concurrent 2-home chains of the 27B model delivered 431.9 answers/min against 210.3 for one on a LAN (**2.05×**) and 87.3 against 43.1 at 150 ms (**2.02×**), with unchanged per-request latency and 160/160 identical. Chain length sets the largest servable model and its latency; chain count sets capacity. With a model that fits one card (Qwen3-8B), each home ran its own server and a pull-based router dispatched requests:

Homes	Answers/min	Scaling	Note
1 (RTX 5070)	1,216	1.00×	
2 (+RTX 5070)	2,416	1.99×	
3 (+RTX 3060)	3,159	2.60×	3060 replicas served ≈0.65–0.75 of a 5070's share
4 (+RTX 3060)	4,101	3.37×	≈ the sum of the hardware
5–6 (+CPU-only homes)	2,818–2,963	2.32–2.44×	stragglers and CPU contention

Table 8. Growth curve (short runs of 4–14 s; preliminary). All 2,440 answers correct.

A speed-aware router (exponentially weighted service times, tail protection for slow replicas, hedged duplicates for late requests) was compared with the naive router over 18 runs of 1,600 requests (28,800 total, all correct). With four GPUs the two were identical (4,073 vs 4,073 answers/min; no hedges fired). Adding a CPU-only home on a separate machine cost −1.2% with the speed-aware router against −3.9% naive, with 3.4× lower run-to-run variance. A CPU-only home co-located with GPU homes cost 17% even with the better router, traced to host CPU contention, which gives the placement rule *one node per machine*. On this model CPU-only homes served ≈1% of requests each.

4.7 Beyond one card's memory

System RAM. A 20.2 GB mixture-of-experts model ran on *one* 12 GB card by holding expert weights in system RAM: 2.1 GB of VRAM with all experts in RAM (35.4 tokens/s on long answers), or 11.4 GB with the first 20 layers' experts in RAM (56.8 tokens/s, 36% of two-GPU speed). Short answers were 40/40 identical to the two-GPU reference. **70B across four cards.** Llama 3.3 70B (34.3 GB), larger than either computer's GPU memory, split across all four cards (≈10 GB each):

Configuration	Correct (40)	Mean s/answer	Long answers tokens/s	Identical to CPU reference
4 GPUs, LAN	40	0.82	6.11	short 40/40; long 4/10
4 GPUs, 150 ms	40	5.07	1.25	short 40/40
CPU + RAM only (one machine)	40	12.46	1.12	—

Table 9. The two GPU runs matched each other on all 10 long answers.

With one request at a time at 150 ms, the stock transport was barely faster than the CPU. With 16 parallel slots, total throughput rose only from 10.6 to 14.2 answers/min: each short question is ≈60 prompt tokens, and prompt processing is relayed to each home in turn.

4.8 A transport modification

We changed two things in llama.cpp's RPC transport, each behind a flag: (1) a new graph is sent as an XOR delta against the previous graph on the connection plus zero-run encoding (lossless); (2) activations between homes travel as 16-bit floats (lossy). A CPU-only check on an 8B model gave 40/40 byte-identical answers with the delta and 38/40 with f16 (the differences were a capital letter and a trailing period).

70B, 4 homes, 150 ms, 100 Mbps	Stock	Modified	Change
One request at a time, answers/min	10.6	14.5	+37%
16 in flight, answers/min (three repeats: 17.1, 19.7, 19.4)	14.2	19.4	+37%
Graph bytes on the wire (437 graphs)	113.2 MB	2.9 MB	40× less

Table 10. All 160 answers correct; 159/160 identical to the reference. The first repeat includes one-time size-query cache misses.

4.9 Usability: can one person chat with a pooled 70B?

Pre-registered target: one user, long answers, at least 5 tokens/s and a first token within 2 s, at two distances, 3 repeats each. Configuration: modified transport, a 1B draft model on the user's card (up to 8 drafted tokens, 30.5% accepted).

Distance per link	Run	Tokens/s	First token, mean (max)
30 ms (same metro)	1 (cold)	5.28	2.98 s (5.58)
30 ms	2	**8.57**	**1.43 s** (1.61)
30 ms	3	**8.46**	**1.46 s** (1.62)
150 ms (cross-country)	1–3	3.61, 3.56, 3.61	2.43, 2.47, 2.38 s

Table 11. Target met at 30 ms on the warm runs; not met at 150 ms (2.9× the stock rate of 1.25 tokens/s).

A per-command profile at 150 ms accounts for each ≈1.0 s pass: three sequential host↔home round trips (≈0.45 s of pure delay plus ≈12 ms per 144 KiB hand-off each way), the RTX 3060 stages (≈3.5 ms per layer vs ≈1.1 ms on the RTX 5070) and ≈0.25 s of host-side work. No hidden round trips remain. A 3B draft raised acceptance from 47% to 51% on two prompts (warm-up only). Direct home-to-home hand-off, which removes one round trip per pass, is implemented but not yet measured.

4.10 Verification and payment in a running network

We built a coordinator with runtime join and leave, speed-aware pull routing, re-queueing of failed requests (a home is marked offline after 3 consecutive failures), audits of 10% of answers on a trusted home with normalized text comparison, banning when a home has ≥8 audits with >25% mismatches (its pending earnings forfeited and all its answers re-done), and a ledger in which credit moves from pending to verified after a 15 s audit window. A scripted run used per-home Qwen3-8B servers: two homes start, two more join, a fake "GPU" that returns fabricated answers in 50 ms joins, then one honest home's server is killed.

Measure (run 2, 9,000 requests, 194 s)	Result
Final answers present in the log and correct	9,000 / 9,000
Dishonest node	banned after 8 audits (7 mismatched), 1.1 s after joining; all 81 of its answers re-done
Honest nodes, audit mismatches	0 in 896 audits
Node killed mid-run	offline after 3 failures; 6 in-flight requests re-routed; none lost
Ledger	9,000 × $0.0004 = $3.6000 verified; $0 pending; $0.0324 forfeited (placeholder rate)

Table 12. Marketplace run. Run 1 exposed a logging defect (75 answers delivered but not flushed to the log before exit); it is reported as 8,925/8,925 verified plus 75 unverifiable, and run 2 used per-record flushing. An independent script recomputes all claims from the raw files (11/11 checks pass).

5. Training results

5.1 Slow links cost time, not learning

Four homes trained the 30M model with fixed rounds of H = 200 local steps, once on a LAN and once with each home on its own 150 ms / 20 Mbps link.

Merge round	2	4	6	8	10
Validation loss, LAN	6.6604	5.9511	5.6268	5.3927	5.2475
Validation loss, 150 ms / 20 Mbps	6.6608	5.9500	5.6262	5.3918	5.2464

Table 13. Learning per round is identical to within 0.001 (int8 rounding). Only time per round changed: ≈29 s on the LAN vs ≈59 s over the slow links (≈30 MB int8 update each way at 20 Mbps, ≈12 s per direction).

The levers for residential links are therefore more useful work per round and smaller updates, not a different learning rule. In ten minutes neither run reached the single card's quality: the RTX 5070s finished each round in ≈15 s and idled ≈15 s waiting for the RTX 3060s.

5.2 Heterogeneity, delayed updates and the outer step

Time-based rounds (30 s; the 5070s then ran ≈530 steps per round, the 3060s ≈230) with work-weighted merging and overlap, using DiLoCo's outer optimizer (Nesterov, lr 0.7, μ 0.9), **diverged**: validation 6.43 → 8.25 → 11.92 within 132 s while the shared-weight norm grew 276 → 898. We attribute this to momentum acting on one-round-delayed, ≈2.7× larger updates. Two 4-minute diagnostics isolated the cause:

Diagnostic (seed 1, 150 ms / 20 Mbps)	Outer step	Overlap	Validation at ≈260 s
F1	plain average (lr 1.0, μ 0)	on	**4.955** (single card ≈5.02)
F2	Nesterov 0.7 / 0.9	off	5.41

Table 14. F1 was stable and passed the single-card curve at ≈228 s. Its settings were frozen as Milestone M2 and pre-registered on new seeds, since they had been chosen using seed 1.

5.3 Milestone M: four homes reach the target before the fastest card

Target: validation loss 4.676, the single RTX 5070's value at its 8-minute evaluation, fixed before any multi-home run. Pass rule: all three new seeds reach it in less than the single card's 452.7 s.

Seed	Single RTX 5070, same seed	Four homes, 150 ms / 20 Mbps each	Speed-up
2	406.8 s	355.2 s	1.15×
3	406.7 s	356.0 s	1.14×
4	418.4 s	357.9 s	1.17×
Mean ± s.d.	410.6 ± 6.7 s	356.4 ± 1.4 s	**1.15×**

Table 15. All three seeds passed the pre-registered rule against the seed-1 baseline (1.27×); the stricter seed-matched comparison is shown. At the end of the 10-minute runs the four homes were also ≈0.1 lower in loss (seed 3: 4.41 vs 4.54 at ≈611 s).

The margin is modest: the pool has ≈3.1× the raw compute of the single card. Most of the gap is the straggler and staleness cost of mixed hardware, which is where further gains lie.

5.4 Adding homes follows a predictable curve

We trained with N = 1, 2, 4, 8 and 16 homes for exactly 20 merge rounds (H = 100, batch 8 × 512, LAN, equal weights) and fitted a saturating power law $L(N) = L_\infty + A \cdot N^{-\alpha}$. The pre-registered check: a fit on N ≤ 8 must predict N = 16 within ±0.03.

Homes N	1	2	4	8	16	32
30M, validation after 20 rounds	6.0096	5.8294	5.6755	5.6295	5.5659	5.5518
Prediction made before the run	—	—	—	—	5.5756 (−0.0097)	5.5377 (+0.0141)
124M, validation after 20 rounds	6.126	5.820	5.596	5.483	—	—
124M, seconds per round	≈22	≈37	≈46	≈68	—	—

Table 16. Both predictions fell inside ±0.03. Five-point fit (30M): L_∞ = 5.485 ± 0.065, A = 0.527 ± 0.060, α = 0.67 ± 0.26 (jackknife s.e.); six-point refit L_∞ = 5.503, α = 0.710.

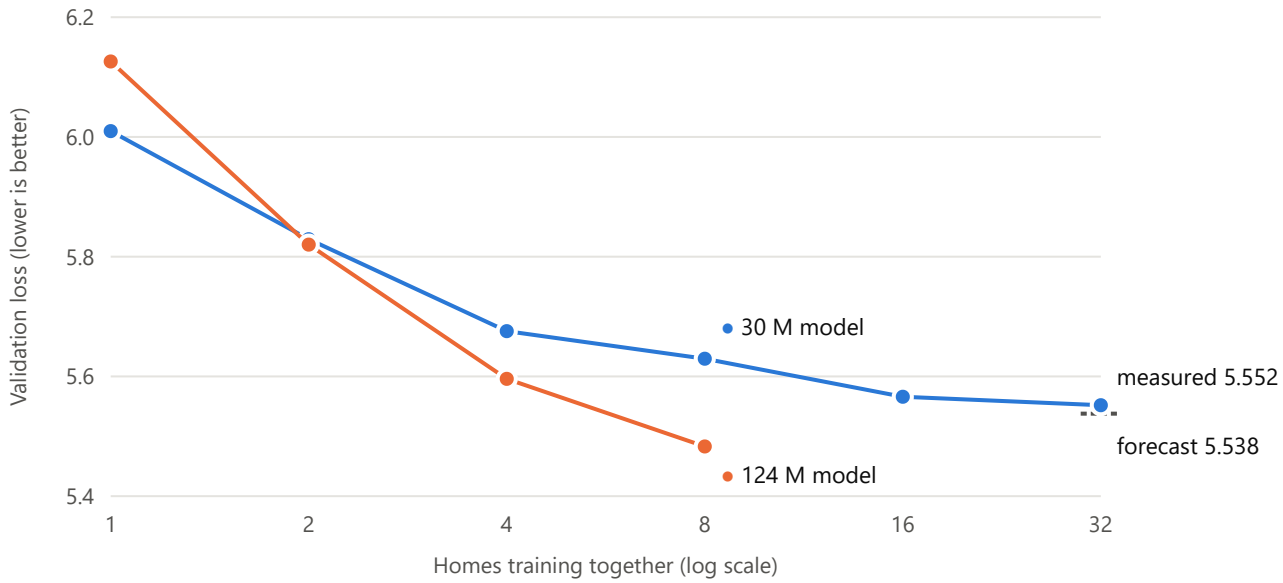


Figure 2. Validation loss after 20 merge rounds vs number of homes. Blue: 30M model; the N = 32 point was forecast (dashed tick) before it was run. Orange: 124M model. Homes beyond four share cards, so this measures learning per round, not wall time.

N ≥ 16 used a batched worker that hosts several homes in one process per card (one CUDA context per card). Equivalence to separate processes was tested before use: maximum difference $6 \cdot 10^{-8}$ on CPU, and its N = 4 and N = 8 curves matched the per-process runs within 0.004 on GPU.

5.5 Bigger models gain more from homes

Repeating the ladder with a GPT-2-small-shaped model (124,082,688 parameters) tested whether the 30M plateau was a property of the model. Pre-registered rule: the 124M gain from 1 to 8 homes must

exceed the 30M gain (0.380) by more than 0.03. Measured: **0.644** (pass by 0.23). The fitted floor moved from 5.503 to 5.245, and 124M with 8 homes (5.483) beat 30M with 32 homes (5.552). The plateau belonged to the small model, not to the network.

5.6 A GPT-2-sized model trained from scratch

Pre-registered goal: a 124M model trained only by these cards, each over its own emulated 150 ms / 20 Mbps link, reaching the GPT-2-small score (3.229) or reporting its curve after 12 h. A 10-minute pilot chose the classic recipe (5.118) over a Muon-based speed recipe (5.203). Final configuration: compiled model, batch 16, 90-second rounds, overlap, work-weighted plain averaging, int8 updates (124 MB per home per round), three homes (both RTX 5070s and one RTX 3060; the fourth card was not used for this run, a deviation from the four homes pre-registered).

Elapsed	0.25 h	1 h	2 h	3 h	5 h	7 h	best (7.3 h)	9.3 h
Validation loss (general)	4.734	3.991	3.785	3.716	3.641	3.617	**3.614**	3.658

Table 17. The 124M run over emulated home links. The scheduled learning-rate decay (last 20% of the 12 h cap) had not begun when the run was stopped at 9.3 h.

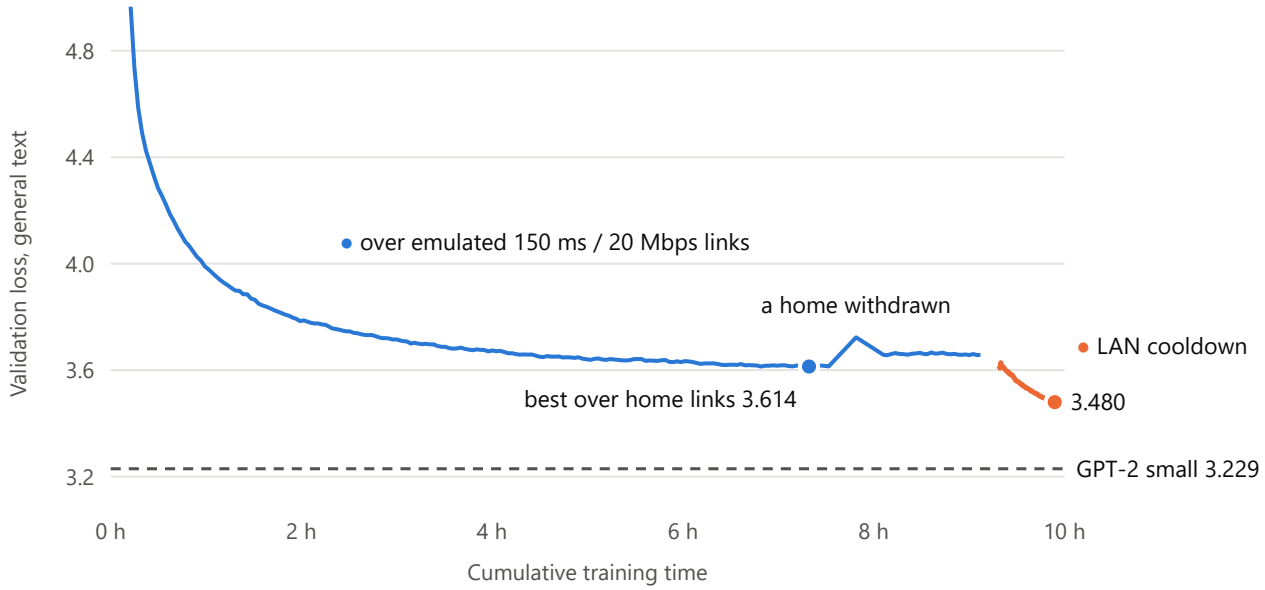


Figure 3. At 7.4 h one home was withdrawn mid-run; the coordinator waited out two 900 s round timeouts and the stale merge cost a temporary jump. The final 35 minutes (orange) ran on the LAN and are reported separately.

A 35-minute LAN cooldown from the best checkpoint (three homes, 30 s rounds, fresh AdamW, linear decay to zero) reached **3.480**, 0.25 from GPT-2 small; the pre-registered target was **not met**. Two earlier configurations failed instructively. 300-second rounds plateaued near 4.36: each RTX 5070 ran ≈3,100 local steps per round, and with overlap each update was a full round stale. The speed recipe stalled when merged: homes trained well alone (training loss 5.36), but Muon moved weights far per round (update norm ≈770 per ≈450 steps vs ≈340 per ≈3,000 for AdamW; cosine to the merge 0.26–0.76), so averaging discarded much of the progress.

6. Getting more from each consumer card (preliminary)

Consumer GeForce cards execute 16-bit matrix math with 32-bit accumulation, the mode used in training, at roughly half the rate of 16-bit accumulation. Measured with cuBLAS at 8192^3:

RTX 5070, inputs / accumulation	TFLOPS	Relative
FP16 / FP16	107.6	1.00×
FP16 / FP32	65.9	0.59×
BF16 / FP32	69.1	0.59×
TF32	34.5	0.30×
FP8 (e4m3) / FP32	138.7	1.19×
FP32	24.2	0.22×

Table 18. The FP32-accumulate penalty was 1.68× on the RTX 5070 and 1.95× on the RTX 3060 (44.5 vs 26.7 TFLOPS).

Our own Triton kernel hits the same limit (67.0 TFLOPS with 32-bit accumulation). *Slice promotion*, which accumulates in 16 bits within 64-wide slices of the inner dimension and promotes each slice into a 32-bit total, reached 97.1 TFLOPS (+45%) with relative error $3.9 \cdot 10^{-4}$, against $2.1 \cdot 10^{-4}$ for full 32-bit accumulation and $2.0 \cdot 10^{-3}$ for 16 bits throughout. In 15-minute head-to-head training runs of the 124M model (same seed, data and batch order):

Method (one RTX 5070 each)	Steps in 15 min	Final validation
Standard 16-bit mixed precision	5,833	4.531
Slice promotion in every linear layer	6,470 (+11%)	**4.465**
BF16 baseline	5,953	4.510
FP8 training (torchao)	6,637 (1.20× steady-state)	**4.477**

Table 19. Single runs. In the first pair the cards sat in PCIe slots of different speed. FP8 was slightly worse per step (≈0.02) but better at equal time. Both need repeats on matched slots and longer runs.

7. What did not work

- **Stock batching does not pipeline across homes** (§4.2), and decode micro-batching made it 2.3× worse.
- **Cross-country single-user chat missed its target** (3.6 tokens/s against 5; §4.9).
- **Self-drafting with several parallel slots crashed** this hybrid model on draft rollback (§4.4).
- **CPU-only homes reduced throughput** when co-located with GPU homes, and contributed ≈1% each on an 8B model (§4.6).
- **Outer momentum diverged** under delayed, larger updates (§5.2).
- **Very long rounds plateaued** (300 s rounds, §5.6): staleness, not bandwidth, was the limit.

- **Muon stalled when merged** (§5.6). Combining fast optimizers with low-communication merging is open.
- **Removing a home abruptly is expensive.** The coordinator waited out two round timeouts while other homes blocked on in-flight updates; a network needs a graceful exit and short timeouts.

8. Discussion

Exactness makes verification cheap, with conditions. Under fixed numeric settings, honest homes produce identical text, so an audit is a string comparison and a dishonest node is detectable within a handful of audits (§4.10). Speculation, batching and device placement each change text slightly without changing correctness. A network must therefore pin these settings per job, or audit with batch size 1, or fall back to normalized or tolerance checks.

Distance should be managed, not fought. Each additional remote stage costs one to two round trips per token. Grouping consecutive stages of a model regionally (tens of milliseconds apart) and adding capacity by replicating groups (§4.6) uses the network where it is strong.

For training, heterogeneity is the first-order problem. Fixed step counts left the faster cards idle half of each round and erased the pool's raw advantage. Time-based rounds, work weighting and overlap turned the same hardware from 0.3 behind the single card into ahead of it. Real networks will be far more heterogeneous than two card types, so these mechanisms are necessary rather than optional.

Records for paying homes. Every pushed update, its size, step count, norm and agreement with the merge is recorded per round, and the inference ledger reconciles to the cent. These are the records a network needs to pay for measured contribution and to weight or exclude bad updates (agreement-weighted merging gave a poisoned home weight 0.00 in offline tests).

9. Limitations

All results come from two computers in one house. Residential links were emulated with fixed delay and bandwidth caps, with no jitter, packet loss, NAT traversal, real routers or real availability patterns. There are four physical cards; experiments with more homes share cards. Models are 8B–70B for serving and 30M–124M for training. Many results have one to three repeats, and the 4–14 s growth-curve runs are preliminary. The marketplace used a fake dishonest node with fabricated answers, not an adaptive adversary. Power, household economics and buyer demand were not measured.

10. Next

1. Real homes in different houses and cities, over their own internet, repeating the serving and training tests.
2. Measure direct home-to-home hand-off for cross-country serving; rollback-safe speculation for parallel requests.

3. Close the GPT-2 gap entirely over home links; a fast optimizer that survives merging; graceful home exit.
4. A model too large for any single card to train; agreement-weighted merging and low-rank updates in live runs.
5. A first paying workload, priced below cloud.

Data and records

Every run, including failed and aborted ones, has its commands, software versions, code and data hashes, per-request and per-update logs, and the time-stamped pre-registration of each test. Two frozen benchmark packages (the marketplace run and the 70B run) include a manifest of file hashes and a script that recomputes every claim from the raw files. These records are kept for audit and will be published with the code.

References

- Anderson, D. P. (2004). BOINC: A system for public-resource computing and storage. *5th IEEE/ACM Int'l Workshop on Grid Computing*.
- Ballotpedia (2026, Nov 6). Voters favored limiting data center development in 40 of 41 local ballot measures on Nov. 3.
- Borzunov, A., et al. (2023). Petals: Collaborative inference and fine-tuning of large models. *ACL 2023, System Demonstrations*.
- Chen, C., et al. (2023). Accelerating large language model decoding with speculative sampling. *arXiv:2302.01318*.
- Douillard, A., et al. (2023). DiLoCo: Distributed low-communication training of language models. *arXiv:2311.08105*.
- Leviathan, Y., Kalman, M., & Matias, Y. (2023). Fast inference from transformers via speculative decoding. *ICML 2023*.
- Lin, Y., Han, S., Mao, H., Wang, Y., & Dally, W. J. (2018). Deep gradient compression: Reducing the communication bandwidth for distributed training. *ICLR 2018*.
- McMahan, B., Moore, E., Ramage, D., Hampson, S., & Agüera y Arcas, B. (2017). Communication-efficient learning of deep networks from decentralized data. *AISTATS 2017*.
- Penedo, G., et al. (2024). The FineWeb datasets. Hugging Face.
- Prime Intellect (2024). INTELLECT-1 technical report.
- Radford, A., et al. (2019). Language models are unsupervised multitask learners. OpenAI.
- Ryabinin, M., et al. (2023). SWARM parallelism: Training large models can be surprisingly communication-efficient. *ICML 2023*.
- ggml-org/llama.cpp, release b12207 (2026).

Appendix A. Hyperparameters and settings

Model	Layers	Width	Heads	Context	Embeddings	Parameters
30M GPT	6	384	6	512	tied	30,160,896
124M GPT (GPT-2-small shape)	12	768	12	512	tied	124,082,688

Table A1. Models (GPT-2 BPE tokenizer).

Setting	Value
Local optimizer	AdamW, lr $6{\cdot}10^{-4}$, 200-step warm-up, β = (0.9, 0.95), weight decay 0.1, gradient clip 1.0, bf16 autocast
Update codec	int8 with per-tensor scales, both directions
Fixed-step runs (§5.1)	4 homes, H = 200, batch 16 × 512, equal weights, Nesterov outer step lr 0.7, μ 0.9, 10 min, seed 1
Milestone M2 (§5.3)	4 homes, 150 ms / 20 Mbps each, batch 16 × 512, 30 s rounds, overlap, work-weighted, plain average (lr 1.0, μ 0), 10 min, seeds 2–4
Single-card baseline	RTX 5070, same model, batch, schedule and data (all 120M tokens)
Scaling ladders (§5.4–5.5)	H = 100, exactly 20 rounds, batch 8 × 512, LAN, seed 1, equal weights, 1/N data slice per home; round timeout 600 s
124M run (§5.6)	batch 16 × 512, compiled, 90 s rounds, overlap, work-weighted plain average, int8; 12 h cap with linear decay over the last 20%; round timeout 900 s
124M cooldown	3 homes, LAN, 30 s rounds, fresh AdamW, 100-step warm-up, linear decay to 0 over 35 min
Serving flags (compared runs)	-ngl 99, --ctx-checkpoints 0, --no-cache-prompt, --cache-ram 0; layer split by -ts
Decoding	temperature 0, top-k 1, seed 7, 48 tokens (short) or 128 (long), thinking off
Usability test (§4.9)	70B Q3_K_M, split 11/12/6/9, context 2048, one slot, draft Llama-3.2-1B Q8, ≤8 drafted tokens, 5 prompts × 96 tokens
Speed-aware router (§4.6)	EWMA α 0.2; slow replicas (EWMA > 2.5× fastest) stop pulling near the end; hedge after max(0.5 s, 3× fastest EWMA)
Marketplace (§4.10)	10% audits; ban at ≥8 audits and >25% mismatch; 15 s audit window; offline after 3 consecutive failures

Table A2. Training and serving settings.

Appendix B. Pre-registration

All times UTC. Each item was filed before the first run it governs; amendments never changed a rule after results were seen. Where a configuration was chosen using an earlier run, it was re-tested on new seeds.

Filed	Item	Rule	Outcome
11-08 16:45	Serving criteria	Correctness is a gate; judge time-to-solution, throughput and latency together; test scale-out and fixed-job speed-up; realistic baseline; retest small failures before any negative conclusion	applied throughout
11-08 16:55	Measurement plan	Per-request logs, independent grading, hashed outputs, every run listed, limits stated per result	applied
11-09 07:15	Usability target U1	≥5 tokens/s and first token ≤2 s at 30 ms and 150 ms, 3 repeats	30 ms met; 150 ms not met
11-09 05:20	Amendment 1	Training records: per-home contributions, checkpoints, curves, bytes, single-card control	applied
11-09 08:55	Amendment 2	Target = single card's loss at its evaluation nearest 480 s	4.676 at 452.7 s
11-09 09:28	Amendment 3	Ladder N = 1–16; fit on N ≤ 8 must predict N = 16 within ±0.03; then out-of-sample N = 32	pass (−0.0097); pass (+0.0141)
11-09 09:45	Amendment 4	M1a/M1b: beat 452.7 s with momentum outer step	M1a diverged; M1b not run
11-09 10:32	Amendment 5	M2 (F1 settings) must beat 452.7 s on seeds 2, 3 and 4	pass on all three
11-09 12:00	Amendment 6	Screen line-search and optimistic merging (≥5% faster)	not run
11-09 14:00	Amendment 7	124M gain N = 1→8 must exceed 0.380 + 0.03	pass (0.644)
11-09 15:05	Amendment 8	124M over home links to ≤3.229, or report at 12 h	not met (3.614; 3.480 after LAN cooldown)

Table B1. Pre-registered criteria and outcomes.

Appendix C. Run register

Every run in the record, in order. "Used" marks runs whose numbers appear in this paper.

Run	Purpose	Outcome
A1, A6	27B one-machine reference (default, matched flags)	40/40; used (A6)
A2–A5	27B two-home split; diagnosis of 10.3 s/answer	caching artifact found; A5 used
A7/B	Five link profiles, single and 8-in-flight	600/600 identical; used
H1	Output layer on host	1.4–2.3× faster under delay; used
S	Self-drafting with MTP heads	2.28× at 150 ms; used
H3	Nagle in the relay	hypothesis rejected
H2-trace 1–3	RPC census	failed (readiness check; port collisions); no data
H2-trace 4	RPC census	used
M	Are step inputs attention masks?	rejected; embeddings identified
E 1 / E 2	Embeddings on first home	flag rejected / −66% traffic, +16%; used
BEST-OTS	H1 + E + MTP, 8 slots	crashed on draft rollback; partial
BEST-OTS-NOMTP	Off-the-shelf baseline	41.9/min at 150 ms; baseline
P, P2, P3	Our build vs official	identical answers; matched speed
U	Decode micro-batching	rejected (2.3× worse)
C	Chains of 2–4 homes	720/720 identical; used
R 1 / R 2	Two concurrent chains	aborted by a GPU-memory watchdog during loading / 2.05×, 2.02×; used
G, G2	Growth curve; speed-aware router	used (G preliminary)
X1	35B MoE on one card with RAM	used
Marketplace 1 / 2	Join, cheat, dropout, ledger	logging defect / clean pass; used
X2	70B across four cards	used
X3	70B, 16 slots, stock transport	2 of 3 sweeps (stopped early); used
W0	Transport change, CPU check	40/40 and 38/40 identical
X4 B / X4 C	Graph delta only / delta + f16	failed to start / +37%; used
U1 1 / U1 2	Usability	draft out of memory / used (spec switches found compiled out; arms relabelled as repeats)
P1, U2	Pass profile; 3B draft	used; warm-up only
Single s1–s4	Single-card baselines	used
Runs 2, 3	Fixed-step, LAN and 150 ms	used
M1a	Momentum outer step	diverged
F1, F2	Diagnostics	used
M2 s2–s4	Milestone M	pass; used
Ladder 30M	N = 1–32	11 aborted attempts (missing data file, link drop, memory, process collisions, OOM, slow data path); 6 clean rungs used
Ladder 124M	N = 1–8	1 aborted (coordinator not started); 4 rungs used
124M pilot	Classic vs speed recipe	classic chosen

124M attempts	s1, s1b, s1c, speed s1, speed s1b, c16 s1	stopped (GPU memory limit); 300 s plateau; stopped to switch recipe; GPU memory limit; Muon stall; compile-cache crash
124M c16 s1b, cooldown	Final run	3.614 / 3.480; used
Arithmetic benches, A/Bs	Formats, slice promotion, FP8	used (preliminary)

Table C1. Run register (condensed; the full log lists commands, hashes and timestamps per run).

Appendix D. Software and data provenance

Item	Identity
llama.cpp	release b12207 (official CUDA 13.5 and 12.4 binaries); our MSVC build of the same tag, answer-identical to official
Training stack	Python 3.11.3, PyTorch 2.11.0 (CUDA 12.8), NumPy 2.4.6, triton-windows 3.7.1
Llama 3.3 70B Instruct Q3_K_M	34,267,499,488 bytes; SHA-256 50221eec… (matches published listing)
Qwen3-8B Q4_K_M	5,027,783,488 bytes; SHA-256 d98cdcbd… (matches published listing)
Llama 3.2 3B Instruct Q4_K_M (draft)	2,019,377,696 bytes; SHA-256 6c1a2b41… (matches published listing)
GPT-2 small reference weights	548,105,171 bytes; SHA-256 248dfc39… (matches published listing); loaded as tensors, no pickle
Science training data	FineWeb-Edu sample/10BT shards 000–004, filtered; 770,975,535 tokens; SHA-256 dc7ae239…
General training data	Same shards, unfiltered; 3,748,955,676 tokens; SHA-256 a7d14912…

Table D1. Software and data. Full hashes are in the records.

Appendix E. Measurement incidents

- **Contaminated GPU (≈50 min).** A stopped smoke test left a training process running on one card. Two usability attempts overlapped it; none of their numbers are used. Rule adopted: after stopping a job, find and stop child processes by command line before the next test.
- **Port collisions.** Three early census attempts failed because their port was already in use by other software or the readiness check fired too early; no data from them is used. Rule adopted: test ports in a reserved range, with a guard that aborts if the port is already listening.
- **Environment drift.** The same official configuration measured 42–46 answers/min early in a session and 32–33 two hours later. Rule adopted: only same-session, alternating comparisons for engine claims.
- **Compiled-out switches.** Per-request speculation settings are disabled in this llama.cpp build, so three planned usability arms were identical; they were relabelled as repeats.
- **Logging defect.** Marketplace run 1 lost its last 75 log records at exit; reported as unverifiable, fixed, rerun.
- **Memory limits.** An accidental 16-home start exceeded available memory and was aborted; later, one training home was withdrawn mid-run (§5.6). All affected runs are marked aborted or reported with the event.
- **Memory watchdog.** A GPU-memory watchdog stopped one replication attempt during model loading, as designed.