Sharded Inference of a 229B-Parameter MoE over the Public Internet at Interactive Speed

Speculative decoding and verifiable execution on scattered consumer GPUs

leyten · c0mpute.ai · July 3, 2026

Abstract. We serve MiniMax-M2.5, a 229B-parameter mixture-of-experts model, split across five consumer RTX 5090s in five European countries. The stages are untrusted and share nothing but the public internet. Single-stream decoding reaches 10–13 tokens/s on interactive reasoning and 70–87 tokens/s on draftable text. Every request returns cryptographic receipts proving each stage did its work, at a measured cost of 0.4% of stage compute. We derive and validate a simple law for speculative decoding over high-latency links: pipelined speculation collapses below a per-token acceptance of $\alpha \approx 0.8$, which no drafter reaches on novel text. The system design follows from that law. We also show that stage time on consumer fleets is bounded by the host CPU, not the GPU. All numbers link to signed receipts in the public repository.

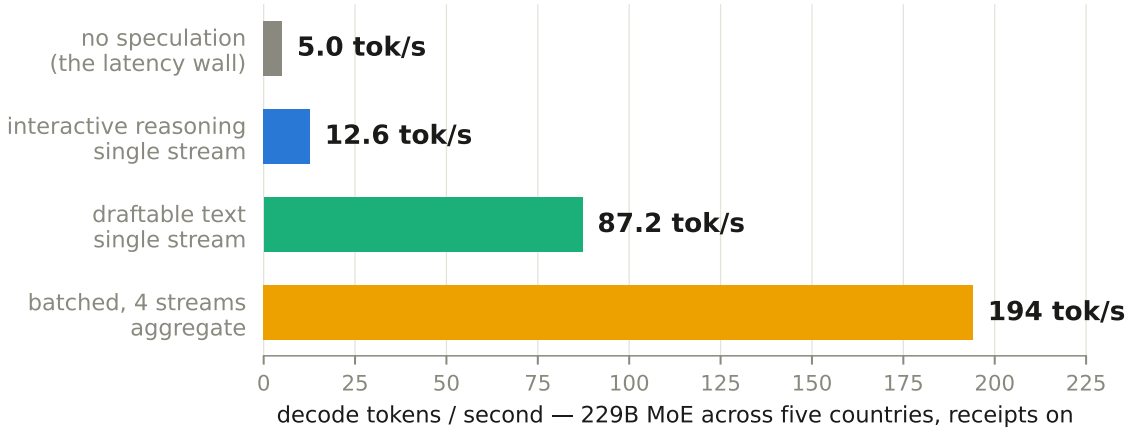

Figure 1: Measured decode throughput by serving mode, one five-stage ring, greedy decoding. Bars show the top measured median per mode; full ranges: baseline 4.8–5.0, interactive 10.7–12.6, draftable 70.7–87.2, batched aggregate 150–194 tokens/s. Verification receipts on (cost: 0.05 ms per 11.7 ms stage span). Receipts: `docs/receipts/m25-paper-*`.

1 Introduction

c0mpute's premise is that idle consumer GPUs can form a permissionless fabric for large-model compute. Decentralized inference is the first test of that premise, and the question that decides it is concrete. Can a frontier-scale model, sharded across GPUs connected only by the public internet, serve a single user at usable speed? And can the user verify they got what they paid for?

This report answers both with a running system. Our engine, **shard**, splits MiniMax-M2.5 (229B total parameters, 10B active, 62 layers, 115 GB in NVFP4) across five RTX 5090s rented from independent hosts in Czechia, Switzerland, Norway and Denmark. Inter-stage round trips are 5–38 ms. No two stages share a network.

Four contributions, each backed by receipts in the repository:

1. **A law for speculative decoding over WAN** (§4). A depth-D speculative pipeline only pays when the previous chunk fully accepts, which happens with probability α^K. That collapses below $\alpha \approx 0.8$, and no current drafter clears 0.8 on novel text. Pipelining therefore cannot fix WAN latency for reasoning

workloads. Our coordinator routes per round instead: pipelined chains on draftable spans, synchronous speculation trees on novel ones.

2. **A measured anatomy of a WAN traversal** (§5.3). Transport takes 55–68%. The compute share is bounded by the host CPU's kernel-launch rate, not the GPU: identical 5090s differ 4× in stage time depending on the CPU and co-tenant load behind them.
3. **Verification that costs nothing** (§5.5). Each stage signs a hash chain over its activations. The coordinator checks signatures and layer coverage against the model's true depth, fail-closed. Measured cost: 0.4% of stage compute.
4. **Fault tolerance, demonstrated live** (§5.6). A coordinator killed mid-decode is replaced on the same warm ring in seconds. No stage restarts, no weights reload.

2 Related work

Petals [1] is the published reference point for decentralized inference at frontier scale: roughly 1 token/s single-stream at 176B over wide-area volunteer GPUs. Our setting is stricter, since no stage is trusted, and our single-stream results are an order of magnitude faster at larger model scale on cheaper hardware.

Recent industry benchmarks of sharded serving measure two GPUs on a LAN and extrapolate to higher latencies. Their conclusions agree with ours where they overlap: pipeline parallelism survives distance, tensor parallelism does not, and MoE models are cheap to shard per parameter. The difference is that we measure the wide-area case directly, with speculative decoding built rather than proposed, and with a verification layer that prior systems lack.

Speculative decoding [2], [3] and its drafter line, Medusa [4] and EAGLE [5], assume the drafter and verifier share a device or a datacenter interconnect. EAGLE-3 [6] holds the strongest published novel-text acceptance, $\alpha \approx 0.74$. Over WAN, verification costs a full network round trip, which is the regime §4 analyzes. Tree verification (SpecInfer [7]) packs more candidates per round trip; our contribution is the routing law that decides, per round, between pipelined chains and synchronous trees.

Full cryptographic verification of LLM inference remains impractical at this scale. Our receipts take the economic middle ground: signed activation hash chains that make free-riding detectable for the price of a hash and a signature per stage.

3 System

3.1 Model and placement

Each stage holds a contiguous block of 10–13 layers, sized to measured free VRAM (weights plus KV within 30 GB per 32 GB card). Pipeline parallelism is the only parallelism that survives WAN latency: a stage forwards one small activation tensor per round, a few tens of kilobytes at fp8, instead of per-layer all-reduces.

3.2 Transport and topology

Stages talk through per-box libp2p sidecars (Noise-encrypted, NAT-traversing). Each stage dials only its successor; the tail returns results directly to the coordinator. Frames use a compact binary codec, no pickling, with activations and drafter tensors quantized to fp8 on the wire.

Ring composition is measured, not assumed. Before weights are placed, the launcher probes the candidate pool: all-pairs RTT, free VRAM, subnet (to exclude co-location), uplink bandwidth, and single-thread CPU speed under load (a consequence of §5.3). A combinatorial optimizer (`shard/topology.py`) picks the subset, ring order and per-node layer blocks, with the coordinator's stage pinned first so the chosen ring is the ring that launches.

3.3 Coordinator and serving surface

One coordinator process drives prefill and decode, runs the drafters, commits verified tokens, and exposes an OpenAI-compatible endpoint with streaming, tool calling and multi-turn support. Greedy decoding is lossless by construction: the ring's own argmax decides every committed token. Speculation only changes how many candidates each round trip evaluates.

3.4 Receipts

Every stage maintains a signed hash chain over its (input, output) activation pairs and its layer range, under a persistent node key. The coordinator verifies the signatures and checks that the attested blocks tile all 62 layers with no gaps, reading the true depth from the model config, never from the receipts. A stage that skips work fails the request. Receipts for every benchmark below are committed to the repository.

3.5 Fault tolerance

A dead coordinator is replaced mid-session: the tail keeps its warm KV and predecessor link and adopts the next coordinator's return channel. A dead stage triggers a cascade re-handshake in which the surviving stages rebuild links without reloading weights. §5.6 shows measured timelines.

4 Speculative decoding over WAN

4.1 The latency wall

A decode step cannot leave the ring faster than one traversal: T ≈ 300–450 ms on a good five-stage European ring. Autoregressive decoding is therefore capped near 1/T, and we measure that cap directly in §5.2: 4.8–5.0 tokens/s on every workload. Throughput is g/T, where g is committed tokens per traversal. Everything else in this section is about raising g or hiding T.

4.2 The accept-gated pipelining law

Keeping D speculative chunks in flight is the classic answer to latency. For speculative chunks it fails in a quantifiable way. Chunk N+1 is drafted assuming all K tokens of chunk N commit, which happens with probability α^K; any rejection flushes the pipe.

- At $\alpha = 0.97$ (verbatim text, n-gram drafter): $\alpha^8 \approx 0.78$. Pipelining pays. The system reaches 70–87 tokens/s on such spans.
- At $\alpha = 0.74$ (EAGLE-3, the strongest published novel-text drafter): $\alpha^8 \approx 0.09$. Most traversals flush, and depth buys nothing.

The crossover sits near $\alpha \approx 0.8$ (Figure 2). We validated the law twice: a Monte-Carlo of the production pipeline, calibrated at both measured operating points, reproduces our novel-text and verbatim throughputs from α alone; and the live A/B in §5.2 shows depth is worthless on reasoning cells and decisive on draftable ones. The corollary is blunt. No engineering pipelines novel-text reasoning through a high-latency ring. Only raising α (drafter training) or cutting T (transport) moves that number.

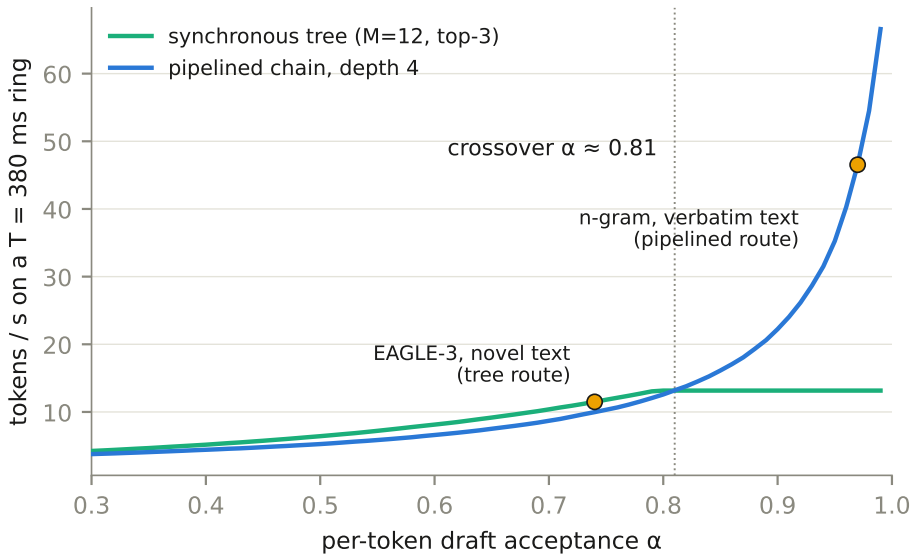

Figure 2: Seeded Monte-Carlo of the production pipeline (flush on divergence, depth 4) against the synchronous tree round, calibrated to both measured operating points. Below the crossover, keeping chunks in flight buys nothing. Reasoning workloads live below it.

4.3 The router

The law dictates a router, not a single strategy. Each round, a cheap n-gram drafter says whether the immediate continuation is draftable (quoting, copying, code echoes, boilerplate).

- **Draftable:** K-token chains as plain verification frames, up to depth D in flight, flash attention on every stage, minimal payload. Divergence discards in-flight chunks and re-anchors.
- **Novel:** an EAGLE-3 head grows a best-first token tree; the ring verifies the whole tree in one forward pass under an ancestor-only mask, and the longest accepted path plus one correction token commits. Trees raise g per round trip exactly where pipelining cannot.

The modes interleave freely under a small KV bookkeeping contract (a tree round leaves its committed path's KV rows dirty; the next frame re-feeds them). A CPU-only harness replays a teacher-forced oracle ring over real sockets and asserts token-exact losslessness and KV integrity across divergences and mode switches: 64 tests, no GPU needed.

5 Evaluation

5.1 Methodology

All arms run on one warm ring, interleaved cell by cell with arm order rotated per repetition, so WAN and co-tenant drift (we measured 1.32× across two hours on the same ring) hits every arm equally. Greedy decoding, reasoning on, receipts on unless stated. Cells cover reasoning, chat, code editing, retrieval quoting, tool calling, and document QA at 8k and 30k context. Full per-job metrics are in the repository; we report medians over repetitions with min–max ranges.

5.2 Single-stream results

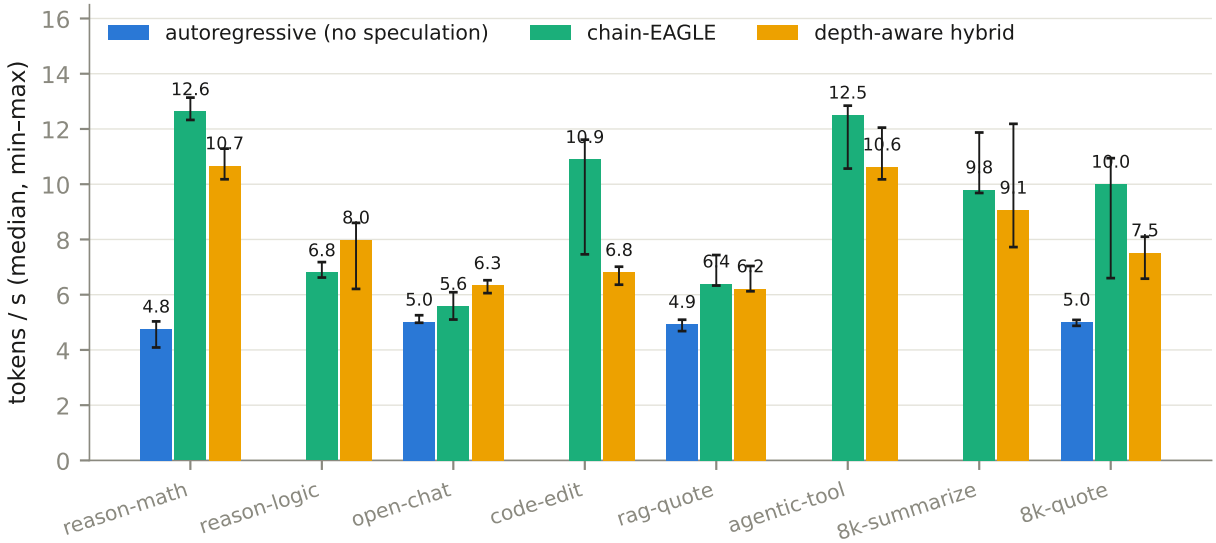

Figure 3: Three arms, interleaved on one ring. Medians over 2–3 repetitions, whiskers min–max. All 64 jobs receipt-verified.

The autoregressive arm measures the latency wall: 4.8–5.0 tokens/s on every cell, exactly $g = 1.00$ by construction. Speculation lifts every cell above it. Interactive cells reach 10.7–12.6 tokens/s median (reason-math 12.6, tool calling 12.5, logic 8.0). Draftable spans change regime entirely: 70.7–87.2 tokens/s single-stream, 14–17× the wall, on the $\alpha \approx 0.97$ branch of the law. A request that reasons first and then copies lands in between, at 16.3.

Two details match the law's fine print. The hybrid and chain arms behave identically on novel cells (same g to two decimals; the router sends the same tree rounds). And the tree-versus-chain preference depends on ring speed: a tree round carries a fixed surcharge, so on fast windows ($T \approx 250$ ms) lighter chain rounds win some novel cells despite lower g. A T-aware router is the obvious refinement.

5.3 Where a traversal goes

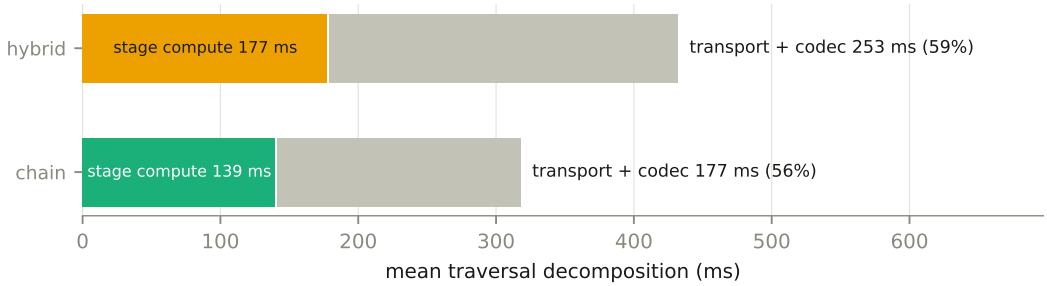

Figure 4: Mean traversal decomposition from per-stage timing stamps.

Per-stage stamps split each traversal into stage spans and a transport remainder. Transport (wire, relay, codec) takes 55–68% on reasoning cells. Against a 105 ms pure-RTT floor, some 65–185 ms is addressable overhead, which is the next engineering target.

The compute term held the surprise: stage time on consumer fleets is bounded by the host CPU, not the GPU. All five GPUs benchmark identically (1523–1527 GB/s, 218–227 TFLOPS, measured in situ). Yet the same 13-layer block takes 11.5 ms behind an idle desktop CPU and 35–50 ms behind an old or co-tenant-loaded server CPU (single-thread probe 0.09 s vs 0.28–0.47 s; one candidate box ran at load average 272). A block forward is hundreds of small kernel launches, and launch cost is single-thread CPU speed. Two consequences: node selection must probe CPU and load (our planner now does), and CUDA-graph capture,

dismissed as a 1.05× lever when measured on a fast CPU, recovers 2–4× exactly on the boxes a permissionless network actually gets.

5.4 Context and batching

Document QA holds usable speed at real context: 9.8–11.2 tokens/s at 8k, 5.0–6.6 at 30k, where the tree round's attention over the full context becomes the bottleneck. Prefill, not decode, is the long-context cost: 22–45 s for a 30k document.

Batching amortizes the WAN cost a single stream must eat. Four concurrent streams aggregate 150–194 tokens/s against 71–87 single-stream, at verified per-stream coherence; an earlier six-stage ring sustained 155 tokens/s at 16.4k context. One constraint surfaced: batch KV competes with stage size. Our 13-layer tail capped batched context near 12k where the six-stage ring's lighter tail reached 16k. Stage sizing and batch capacity trade off, and the topology planner can optimize for either.

5.5 The cost of verification

Receipts ride every request above (64/64 bench jobs verified, fail-closed). On drift-free idle-CPU stages the span is 11.72 ms with receipts and 11.67 ms without: +0.05 ms, about 0.4%. The larger end-to-end difference between the on and off phases (9–16%) is fully explained by measured WAN drift between the runs; the span data bounds the true cost two orders of magnitude below it. Verification, at this granularity, is free.

5.6 Fault tolerance, live

One continuous timeline from the timestamped demo. t = 0: a baseline job completes at 14.7 tokens/s. t = 14 s: the coordinator is killed with SIGKILL, four chunks in flight. t = 34 s: a new coordinator connects to the same ring; the tail keeps its warm KV, adopts the new return channel, and drops the dead job's frames. t = 48 s: the new coordinator has completed a full job, prefill included. No re-warm, no stage restart, no weight reload. t = 60 s: a receipts-enabled job passes signature and full 62-layer coverage checks on the recovered ring.

6 Limitations

- **The single-stream ceiling is physics, and we operate near it.** The law bounds serial novel-text decoding; on this ring class the ceiling is roughly 12–14 tokens/s and we measure 10–13. Remaining levers are engineering worth tens of percent, plus one research bet: drafter training toward $\alpha \approx 0.8$, which would also unlock pipelining. Batched serving does not share the ceiling.
- **Regional rings.** Results are intra-European (5–38 ms legs). Transcontinental rings multiply T; the fabric should compose regional rings instead.
- **Right-sized models.** A 229B model needs five hops. A 30–70B model needs one or two, with proportionally higher single-stream speed on the same fabric.
- **Trust scope.** Receipts prove layer coverage and bind activations to stages. They do not yet bind to wall-clock or prevent replay. Freshness binding and randomized spot-recomputation are the next layers.

7 Conclusion

A 229B mixture-of-experts model serves a single user at interactive speed from five consumer GPUs in four countries, and every request arrives with proof that each stage did its share. The design came from measurement: a law that says when speculation can hide the network and when it cannot, a timing split that shows the bottleneck on consumer fleets is the CPU next to the GPU, and a benchmark discipline that publishes the receipt behind every number.

The same measurements set the roadmap. Drafter training toward the $\alpha \approx 0.8$ crossover is the one lever that raises the reasoning ceiling, and crossing it would unlock pipelined speculation as a second gain. The verification layer extends next to freshness binding and randomized spot-recomputation, closing the replay gaps listed above. CPU-aware node selection and CUDA-graph capture recover the stage time that slow

hosts currently waste. And because the engine is model-agnostic, the fabric that carries a 229B model in five hops carries a 30–70B model in one or two, with single-stream speeds to match.

The supply side of decentralized AI already exists: hundreds of millions of consumer GPUs sit idle at any hour. What this report shows is that using them honestly, at frontier scale, over the network we already have, is a tractable engineering problem. We intend to keep publishing the engineering, and the receipts with it.

Reproducibility every number links to a receipt under `docs/receipts/`; harnesses are `research/m25_paper_bench.py` and `research/m25_usability_report.py`; the CPU-only correctness harness is `tests/`.

Bibliography

[1] A. Borzunov *et al.*, "Petals: Collaborative Inference and Fine-tuning of Large Models," 2022.

[2] Y. Leviathan, M. Kalman, and Y. Matias, "Fast Inference from Transformers via Speculative Decoding," in *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[3] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper, "Accelerating Large Language Model Decoding with Speculative Sampling," 2023.

[4] T. Cai *et al.*, "Medusa: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads," 2024.

[5] Y. Li, F. Wei, C. Zhang, and H. Zhang, "EAGLE: Speculative Sampling Requires Rethinking Feature Uncertainty," in *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[6] Y. Li, F. Wei, C. Zhang, and H. Zhang, "EAGLE-3: Scaling up Inference Acceleration of Large Language Models via Training-Time Test," 2025.

[7] X. Miao *et al.*, "SpecInfer: Accelerating Large Language Model Serving with Tree-based Speculative Inference and Verification," in *Proceedings of ASPLOS 2024*, 2024.