

Kafila: Serving Large Language Models on a Trusted Set of Heterogeneous Commodity Machines

PHD CANDIDATE

Murtaza Rangwala

mrangwala@student.unimelb.edu.au

qCLOUDS Lab • School of Computing and Information Systems • The University of Melbourne

SUPERVISORS

Prof. Richard O. Sinnott

Prof. Rajkumar Buyya



Kafila lets a small group run one large language model across the laptops and desktops they already own, by giving each machine the share of the model it can actually serve.

5.2×

less waiting on the slowest machine than splitting the model evenly across devices

3.0×

less waiting than splitting by how much memory each device has

75–87%

of the group's hardware kept busy, where existing splits leave more than half idle

3.2×

more tokens per second than an even split when four people use the session at once

1 • The problem: a bounded session

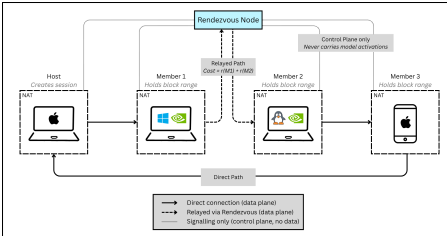
A research group or a circle of friends owns several consumer computers. None is large enough to run a capable LLM alone; together they may be. The motivating workload is a model over a person's own documents, messages and health records: content people are least willing to send elsewhere.

Existing systems (Petals, SWARM, Parallax) pool capacity across an *open swarm*: many peers hold each block, so a slow peer is routed around. A group admitting only trusted machines gives up exactly that redundancy.

A bounded session must use every device it admits. Its pipeline advances at the pace of whichever device received a share it cannot serve quickly, so the division has to be right *before* serving begins.

2 • Session model

A **session** is a bounded set of devices, assembled on demand to serve one model, dissolved when its last member leaves. Membership is defined by relationship, not proximity: one household, or a lab spread over two continents.



A four-member session. Data-plane edges run over QUIC, encrypted end-to-end with TLS 1.3: direct where hole punching succeeds, relayed through the rendezvous as ciphertext where it fails. The rendezvous holds no weights and does no inference.

- One member holds the **head** (embedding, unembedding, sampler); the planner chooses which, not whoever opened the session.
- Prompts are accepted at every member; the laptop that contributes a shard is also where its owner types.
- The session code is the whole access-control model: a bearer token shared with a household, a lab, or collaborators.

CONTRIBUTIONS

- A **session protocol** that forms a pipeline entirely from behind NAT, with no router configuration.
- A **planner** that divides the model exactly, for a fixed ring order, from each device's measured bandwidth, capacity and reachability.
- An **evaluation** on three real fleets, from a shared LAN to two continents, on models up to 32B parameters.

3 • Protocol: a ring behind NAT

No participant is ever required to accept an unsolicited inbound connection or to configure a router. Each node is classified once against the rendezvous and STUN (O(m) probes); every pair's feasibility is predicted from RFC 4787 behaviour classes (arithmetic, no packets); only the edges the plan uses are verified.

A direct edge costs one round trip; a relayed edge costs both members' round trips to the rendezvous, paid on every token, so the planner prices both in one unit.

NAT CLASS PAIRS, EMULATED • RULE VS. TRAVERSAL

Initiator	Responder	Rule	Achieved
permissive	any	direct	direct
ordinary	permissive	direct	direct
ordinary	ordinary	direct	direct
ordinary	restrictive	relay	direct
restrictive	permissive	direct	direct
restrictive	ordinary	relay	direct
restrictive	restrictive	relay	relay

Seven of nine pairs predicted correctly; both errors forecast a relay where traversal succeeded directly, never the reverse. On real networks: US Central punched every edge direct (median 1.29 s); intercontinental relayed four of five edges.

4 • Planner

Decoding at batch size one is memory-bandwidth-bound. Each stage's cost is *derived*, not *fitted*, from a short probe taken before any weight is fetched:

$$c_i = o_i + \frac{n_i \beta_{\text{block}}}{b_i} + \mathbb{1}[i = \text{head}] \frac{\beta_{\text{head}}}{b_i}$$

b_i : measured memory bandwidth • n_i : blocks assigned • o_i : per-token overhead • β_{block} , β_{head} : byte sizes from the checkpoint manifest

- Ring order** from measured reachability: a closed-form rule over NAT classes, replaced by exhaustive search over priced edges once round trips are known; ties broken on ring separation.
- Block placement** for a fixed order is chain-on-chain partition (Pinar et al.), solved exactly by **PARTITION**: every stage cost is affine in its block count, so the bottleneck is one of at most $k \cdot m$ candidates fixed ($i + n \cdot \text{per}(i)$, for k blocks and m devices; binary search over them with a greedy left-to-right feasibility pass runs in $O(km \log km)$).
- Head placement** is solved jointly: m rotations, one exact solve each; the head lowers a device's block limit as well as raising its cost. The anchor keeps the head unless a rotation improves by more than $\mu = 5\%$, chosen against overhead noise.
- Admission** degrades through three tiers (full cost model → arithmetic throughput → working set) and refuses a session no division fits, naming the undersized device rather than reporting only that the session cannot run.

The cost model is fitted to nothing, yet predicts per-stage compute to a median of **18%** across 165 stages, so the residual imbalance is that prediction gap, not a limit of the method.

5 • Evaluation on three real fleets

Ollama fork on MLX (Metal and CUDA); Qwen3-8b/14b/32b. Baselines: **Uniform** (GPipe's equal layer count) and **Mem-prop.** (exo's memory-proportional share). Bandwidth spans 11×, memory 4×, and on every fleet speed and memory rank differently.

LAN

4 devices, Melbourne
12% network overhead

US Central

5 devices, 4 cities
38–62% overhead

Intercontinental

5 devices, RO • CZ • AU
84–94% overhead

BOTTLENECK STAGE, MS (LOWER IS BETTER) • UTILISATION (HIGHER IS BETTER)

One scale; per-device times, independent of network.

LAN • Qwen3-8b

Kafila	11.3 • 86%
Mem-prop.	26.0 • 48%
Uniform	36.2 • 41%

US Central • Qwen3-8b

Kafila	5.2 • 81%
Mem-prop.	5.6 • 77%
Uniform	5.6 • 81%

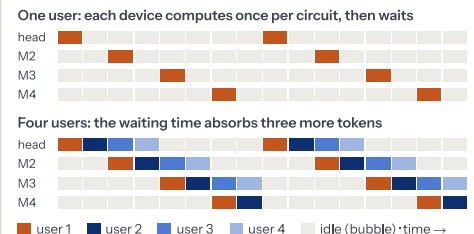
Intercontinental • Qwen3-32b

Kafila	18.1 • 85%
Mem-prop.	54.6 • 41%
Uniform	infeasible

Where a fleet holds a laptop, each heuristic buries it. US Central's five GPUs are closer in speed, so the divisions nearly agree. No uniform split of Qwen3-32b fits the 18 GB member; the planner names it and serves the model.

6 • Filling the pipeline bubble

A token passes through every member before the next begins, so at any instant one device computes and the rest wait. Concurrent users fill that gap: each device works on one person's token while the ring carries another's.



Schematic: one unit of compute per member, one per hop. Four users fit into otherwise idle time, each served at almost the rate of one.

On the LAN, four users raise Kafila's throughput **4.0×** against 2.5× (memory-proportional) and 2.0× (uniform). Imbalanced splits stall at their overloaded stage; only a balanced division can fill the bubble, so a 1.56× lead widens to 3.2×

Conclusion

A trusted set of heterogeneous commodity machines can serve an LLM none of them could run alone. Every admitted session formed without router configuration. What a session gives up in redundancy it returns as membership, capability and connectivity known before serving begins.

Future work

- The network term bounds what remains to recover.
- Membership that changes mid-session: cache recovery around a replan.
- Mobile members: how often to replan, not how.
- Sharing reserved cache across members.



Paper, code and data
kafila.co/research/kafila