LOXILB INFERENCE GATEWAY

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT KOREA

GPU-AWARE LOAD BALANCING FOR LLM INFERENCE

Beyond Round-Robin

Your load balancer picks a GPU from a 5-tuple. The GPU that already holds the prompt's KV blocks is sitting right there — and it just recomputed them again.

Seokhwan Kong
CO-Founder, NETLOX · creator of LoxiLB

github.com/loxilb-io/loxilb-inference-gateway

IF YOU REMEMBER THREE THINGS

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
KOREA

01

Cache locality beats fairness

For LLM traffic, TTFT is dominated by whether the chosen GPU already holds the prompt's prefix. Round-robin scatters exactly the requests that should stay together — and at saturation, it has no usable floor at all.

02

Mirror the engine, don't guess it

We recompute vLLM's own block hashes and subscribe to its KV-event stream, so routing reflects what the GPU actually holds. The good projects all do this now — it is table stakes, not a moat.

03

It belongs in the data plane

One Go/eBPF binary. No Envoy, no ext-proc chain, no mandatory Kubernetes. Bare metal, VM, or DPU — the same decision, in the hot path.

SAME PROMPT · SAME FLEET · SAME SECOND

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
KOREA

Two identical requests. Two very different bills.

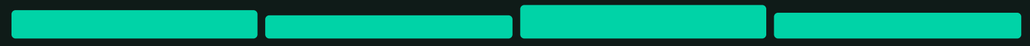
REQUEST A → GPU 2

Prefix already cached

3,000-token preamble → reused from KV cache

Prefill work → the new turn only

Time-to-first-token → fast



REQUEST B → GPU 5

Recomputes the identical tensors

3,000-token preamble → full forward pass, again

Prefill work → everything

Time-to-first-token → inflated

Nothing failed. The load balancer did precisely what it was configured to do — it just had no way to know the difference.

THE VISIBILITY GAP

A 5-tuple cannot describe an LLM request

WHAT THE BALANCER SEES

src 10.2.4.71:51824

dst 10.10.10.254:8080

proto tcp

WHAT ACTUALLY DECIDES COST

"model": "Qwen3-8B"

"messages": [3,000 tokens]

prefix → held by GPU 2

Requests carry a model name

One VIP can front several models. The right pool is named in the JSON body — invisible at L4.

Backends hold warm state

Sending a follow-up turn to the worker that already computed the prefix skips the rebuild entirely. Stateless routing throws that away.

Responses stream for minutes

SSE connections must stay open, dodge idle timeouts, and still be accounted per token.

ONE REQUEST, TWO WORKLOADS

Prefill and decode want different machines

PREFILL (P)

Compute-bound

A full forward pass over every input token in parallel, building the initial KV cache. Latency scales with prompt length. Wants high FLOPS.

this is the part that gets **recomputed**

DECODE (D)

Bandwidth-bound

One token at a time, each pass re-reading the whole KV cache from GPU memory. Throughput scales with memory bandwidth.

this is the part that **streams** to the user

Split them onto separate pools and each scales on its own terms — but now something has to route both legs and move the KV between them.

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT KOREA

THE PROPOSAL

Put the inference decision in the load balancer itself.

Loxilb-inference-gateway is a fork of loxilb that adds inference-aware routing on the same Go/eBPF data path. One binary covers L4 through inference-aware L7 — and every AI feature is opt-in per rule.

Apache-2.0 | no Envoy | no ext-proc chain | no mandatory Kubernetes | vLLM · SGLang

ARCHITECTURE

One binary: control plane, eBPF, and an L7 fullproxy

OpenAI-compatible clients

POST /v1/chat/completions

HTTP · SSE

LOXILB INFERENCE GATEWAY · single Go/eBPF process

Control plane (Go)

REST API :11111 /netlox/v1

KV-cache selector

routing rules & endpoints

P/D coordinator

▼ programs

Data plane

eBPF XDP/TC — L4 fast path

sockmap zero-copy

fullproxy (mode 4) — L7

SSE stream relay

prefill pool

ep_role: 1

decode pool

ep_role: 2

SGLang pool

single-role

◀ — KV-cache events (ZMQ) ———————————————— the feedback loop this talk is about

LIGHTWEIGHT AI TRAFFIC CONTROL PLANE

It has to be a good load balancer first

edge sites & enterprise AI clusters

FOUR SIGNALS YOUR L4 TIER NEVER HAD

Prefix-aware

Continuations follow their own history, no conversation ID required.

KV-aware

Exact block hashes from the engine's own event stream — not a guess.

GPU-aware

A warm but saturated engine loses to a cooler one. Affinity is capped.

Session affinity

The cheapest cache hit there is — and it was always a load balancer's job.

THE PRICE OF BEING ALLOWED IN THE PATH AT ALL

Retry

Every tier fails open. A miss re-picks; it never drops the request.

Health check

Unhealthy endpoints leave the pool before the router ever scores them.

K8s Service · bare metal

Service-type LB and kube-proxy replacement — or no Kubernetes at all.

Prometheus

One scrape target for L4 and inference alike. Shipped alerts, four tiers.

Our goal wasn't to build another AI router. It was to build a **production-grade load balancer that truly understands AI workloads**.

THE TIER LADDER

One request. Four decisions. One best GPU.

request → health / CB mask → Tier 0 → Tier 1 → Tier 1.5 → Tier 2 → any-healthy → 503

TIER 0 Session

Have I seen this conversation before?

X-Conversation-Id: absent

A	—
B	—
C	—
D	—

MISS — no session key

↓ falls through

TIER 1 Trie

Does the prompt look like something I routed before?

our own radix trie of past routes

A	12%
B	8%
C	5%
D	0%

MISS — best 12% < 20% threshold

↓ and the trie can go stale on eviction

TIER 1.5 KV-exact

Which GPU actually holds these blocks, right now?

live ZMQ block inventory

A	3
B	41
C	7
D	0

✓ ROUTE TO B — 41 blocks already resident

bounded by the load cap

TIER 2 Min-load

Forget the cache. Who is least busy?

active_conns + queued

A	5
B	6
C	4
D	2

→ would pick D — the one cold GPU

the fallback, and the trap

The coldest GPU is often the *least loaded* — precisely because it is cold. So least-connections doesn't merely miss the cache: it is biased toward missing it.

TIER 1 · THE INTEGRATION QUESTION EVERYONE ASKS

Three radix trees — and why we sync none of them

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT KOREA

IN THE GATEWAY

Our Tier-1 prefix trie

“Where did *I* route this prefix recently?”

```
root
 ├ "You are a…"  → B
 ├ "Summarize…"  → C
 ├ "def parse…"  → A
```

What I sent, and when. Self-referential — assumes past routing implies present cache. Blind to eviction.

INSIDE EACH WORKER

RadixAttention

“What is *actually* in my GPU cache right now?”

GPU B · 41/64 pages 

"You are a…" ✓ held

"Summarize…" ✗ evicted

Ground truth — but server-local. No router can query it per request at line rate.

IN THE COMPETING ROUTER

Router-side cache_aware tree

“What do I *think* each worker has?”

```
root
 ├ "You are a…"  → B ✓
 ├ "Summarize…"  → C ✓
 ├ "def parse…"  → A ✓
```

Still believes C is warm. Optimistic drift — never saw the 14:05 eviction, or /flush_cache, or restarts.

SO WE SYNC NOTHING — THE HASH ALREADY IS THE TREE

hash(k) = SHA256(parent_digest ‖ tokens) — a hash at depth *k* encodes the whole prefix path to the root. An inventory of chained hashes *is* the radix tree, flattened into its node-paths.

Events beat snapshots on the axis that decides outcomes: **eviction truth.** A simulated tree only ever learns about arrivals.

TIER 1.5 · THE INVENTORY PLANE

We move hashes, not tensors

ON THE WIRE

~8 bytes

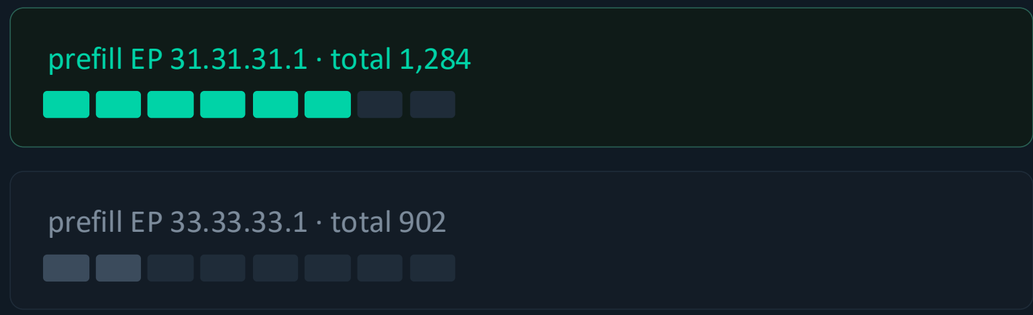
per block — a 64-bit hash key, plus add/evict deltas. That is the entire sync payload.

WHAT NEVER MOVES

GB of KV

The tensors stay in GPU memory. We only need to know *where* they are.

PER-ENDPOINT BLOCK INVENTORY



Inventory updates track evictions within one message cycle
so a score reflects current GPU memory , not what we hoped was still there.

TIER 1.5 · THE REQUEST SIDE

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT KOREA

The prompt *is* the routing key

1

Terminate

Fullproxy (mode 4) reads the full HTTP body. You cannot hash what you cannot see.

2

Tokenize

In-process, with the model's own HuggingFace tokenizer.json staged on the host.

3

Block

Group token IDs into fixed-size blocks — kvBlockSize: 16 , matching the engine.

4

Hash

Canonical CBOR per block, chained through the parent hash — vLLM's exact recipe.

5

Score

Overlap count against each endpoint's inventory — then bound it by load.

No explicit conversation ID is required. The prompt content is the key — which means it works for shared system prompts, RAG documents, and few-shot preambles, not just chat sessions.

PER-FRAMEWORK KV ANALYSIS

Same tokens in. Different bytes out.

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT KOREA

	① the prompt IDENTICAL	② what gets hashed DIFFERS	③ digest DIFFERS	④ which 8 bytes → wire DIFFERS
vLLM	t0 · t1 · t2 · … · t15	CBOR array parent_hash [t0 … t15] null	SHA-256 or XXH3-128	LAST 8 bytes, BE u64 · unsigned
SGLang	t0 · t1 · t2 · … · t15	raw concat — no envelope parent 32B · t0 · t1 · … each token = 4-byte LE	SHA-256 only	FIRST 8 bytes — the inverse int64 · signed, often –

block 0's parent

vLLM NONE_HASH ← PYTHONHASHSEED

SGLang nothing — bare tokens

granularity knob

vLLM --block-size · 16

SGLang --page-size · model-dependent

topology

vLLM P/D roles · kvExactMode 1

SGLang one flat pool · kvExactMode 3

The one thing that *didn't* change: the ZMQ wire format is byte-identical — the decoder needed zero changes. Framework-agnostic is earned per engine, with committed parity vectors.

BOUNDING THE AFFINITY

Argmax alone would melt one GPU

Fifty clients sharing one hot system prompt all score highest on the same endpoint. Pure overlap-argmax sends every one of them there while its siblings idle.

HARD MODE — DEFAULT

$$cap_i = \lceil (1+\varepsilon) \cdot totalLoad \cdot capacity_i / totalCap \rceil$$

Argmax overlap *among under-cap endpoints*. Default ε = 0.75 — an endpoint keeps its affinity traffic while it carries at most 1.75× its capacity-fair share.

Higher ε → more affinity-preserving.
Lower ε → spills sooner.
When the winner is over cap and selection moves off it, that is a spill.

soft — continuous cost blend

$$cost_i = uncached_i \cdot 1000 + (\lambda \cdot load_i)/capacity_i$$

No hard cutoff. λ defaults to 32. At zero load it reduces exactly to overlap-argmax.

adaptive — load-keyed ε

$$\varepsilon\_eff(L) = clamp(175 + 125 \cdot (L-16)/10,\ 175,\ 300)$$

No static ε wins across all load. Tight ε is better at moderate rate, loose at saturation — so scale it with the load the selector itself observes.

LOXILB_KV_LB_MODE = off | hard | soft | adaptive | adaptive-soft

THE WHOLE CONFIGURATION

One POST. No CRDs, no operator, no sidecar.

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
KOREA

```
{ "serviceArguments": {
      "externalIP": "10.10.10.254", "port": 2020, "protocol": "tcp",
      "mode": 4, // fullproxy — the AI prerequisite
      "pd_disagg_mode": true, // split prefill / decode
      "kvExactMode": 1, // Tier 1.5 on (vLLM, ZMQ)
      "kvZmqPort": 5557,
      "kvHashAlgo": "sha256_cbor", // must match the engine
      "kvBlockSize": 16, "kvWarmupSec": 30 },
      "endpoints": [
            { "endpointIP": "31.31.31.1", "targetPort": 8000, "ep_role": 1 }, // prefill
            { "endpointIP": "32.32.32.1", "targetPort": 8000, "ep_role": 2 } // decode
      ] }
}
```

The gateway runs on a CPU host — it needs no GPU of its own. Bare metal, VM.

THE FLEET WE MEASURED ON

A real, deliberately uneven GPU fleet

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT KOREA

NAVER Cloud P/D testbed Qwen2.5-7B-Instruct · MML 32768

NODE	ROLE	GPU	CAPACITY
prefill-1	prefill	L4 · 24G	2,356 tok/s
prefill-2	prefill	L4 · 24G	2,326 tok/s
prefill-3	prefill	L4 · 24G	2,347 tok/s
prefill-4	prefill	L40S · 48G	8,253 tok/s
decode-1	decode	L40S · 48G	NIXL mesh

One prefill node is **3.548×** the others. A flat connection count would starve it — which is precisely why the bounded-load cap is capacity-weighted.

PINNED SUBSTRATE

vllm/vllm-openai:v0.17.0 --block-size 16 --prefix-caching-hash-algo sha256_cbor PYTHONHASHSEED=0 ZMQ KV events :5557

WORKLOAD

Mooncake tool-agent trace, replayed open-loop Poisson at three offered rates — light, moderate, saturated.

SLO

TTFT ≤ 10,000 ms TPOT ≤ 245 ms

Derived from our own P/D unloaded baseline — not borrowed from a direct-endpoint number.

MEASURED · 4-ARM × 3-RATE GRID · POOLED N PER ARM

Affinity routing's value rises with load

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT KOREA

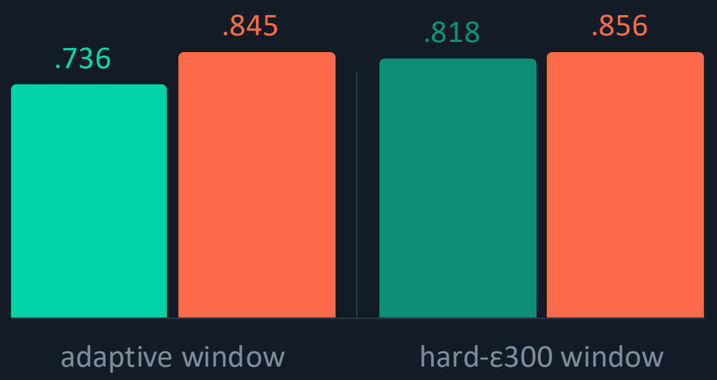
goodput@SLO — higher is better TTFT 10 s / TPOT 245 ms

loxilb adaptive loxilb hard ε300 vllm-router

⚠ bars pair only *within* a campaign window — the fleet is non-stationary, so vllm-router was re-measured in each window

0.5 req/s

light

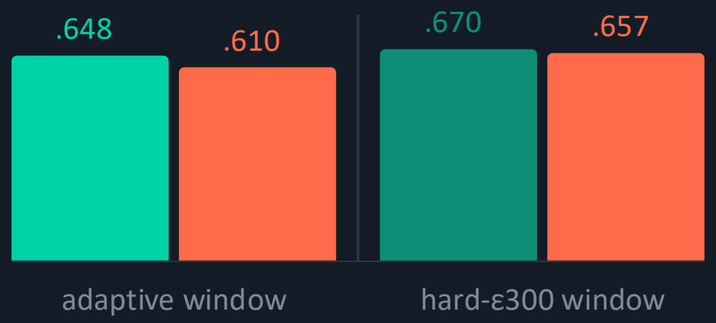


We lose in both windows. Pinning work an idle fleet could spread costs you. Affinity does not pay unsaturated.

RR floor .831 — better than both

1.0 req/s

moderate

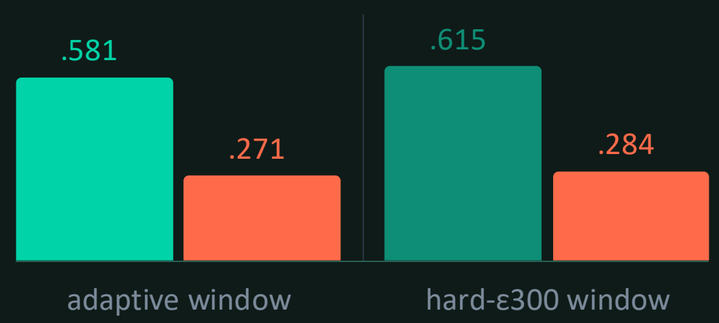


TOST parity, both windows. Statistically *equivalent* to vllm-router — an equivalence test, not a win and not overlapping error bars.

Δ +0.038 and +0.013 · RR floor .665

2.0 req/s

saturated



Win in both windows. The RR floor was conn-storm-reaped — pure balance gives no usable floor at overload. That is itself the point.

~2.14× and ~2.17× · CI excludes zero

Both modes agree at every rate — so the saturation win is a property of KV-exact routing itself , not of one tuned knob. Cache locality is only scarce when the fleet is busy. That is exactly when you care.

WHERE THIS IS GOING

KV routing is one feature. Here's the thesis.

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT KOREA

an inference-aware load balancer — not an LLM appliance

01

EDGE CLOUD

Lightweight enough to live at the edge

One CPU-host binary. eBPF XDP/TC and kernel sockmap on the L4 path, inference-aware L7 in the same process.

→ No GPU needed for the gateway itself

→ No Kubernetes anywhere in the requirement

→ Bare metal · VM · telco MEC · BlueField DPU

The envelope a gateway + EPP stack cannot reach.

02

AI-CONTROLLER

Route on anything you can measure

Cache overlap and queue depth are two signals. A frozen gRPC contract lets an external controller stream full state-of-the-world snapshots and bias selection.

→ Returns only a weight 0–100 and a lifecycle state, per P/D role

→ Snapshots are structurally forbidden from carrying load fields — a reflection test fails the build

→ Controller dies → weights decay to neutral, routing carries on

It can bias the ladder. It can never break the hot path.

03

FRAMEWORK AGNOSTIC

Not one engine, and not only LLM

vLLM and SGLang today via their native KV-event contracts — which genuinely differ. The inventory plane is the porting surface for the next engine.

→ TGI and others: same contract, same surface

→ Bounded load, affinity, health masking are not LLM concepts

→ Embeddings · rerankers · ASR · vision · classical ML serving

Accelerator-aware traffic, whatever is behind it.

THE QUESTION YOU'RE ALL HOLDING

The routing has converged. The footprint hasn't.

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
KOREA

	llm-d	NVIDIA Dynamo	vllm-router	loxilb
Precise KV-cache routing	Yes — KV-events indexer	Yes — KV-aware router	Approximate	Yes — exact block hash
P/D disaggregation	Yes	Yes	—	Yes — NIXL
Where the decision runs	EPP pod, via Envoy ext-proc	Dynamo router service	Router process	In the LB data plane
Runs without Kubernetes	No	Effectively no	Yes	Yes — metal · VM · DPU
Also your L4 load balancer	No	No	No	Yes — XDP/TC, svc-LB
Tiers you operate	Gateway + EPP + CRDs	Runtime + coordination svcs	One process	None — one binary

Anyone claiming they alone read the engine's real cache state is selling you something. Three of us read the same ZMQ events. The difference is how many tiers you run to get there.

WHEN NOT TO USE US

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT KOREA

They make your serving stack inference-aware. We make your load balancer inference-aware.

USE llm-d OR DYNAMO WHEN

Kubernetes is your substrate

You already run Gateway API Inference Extension, and adding an EPP is a config change, not a new tier.

You want scheduling *and* autoscaling — a planner that scales replicas, not just routes to them.

You need multi-node model parallelism, or their tiered KV story.

You value the ecosystem gravity of Red Hat, Google, IBM, NVIDIA behind it.

Those are good reasons. We are not trying to win that deployment.

USE LOXILB WHEN

A load balancer is already in the path

Your GPUs sit on bare metal, VMs, a telco edge, or a DPU — where the others are not options at all.

You'd rather your *existing* L4 tier got smarter than add a second control plane above it.

Classic service-type LB, kube-proxy replacement, or 5G traffic shares the same node and the same binary.

Box count going *down* matters more to you than scheduling depth.

We are the traffic layer. We don't schedule pods and we don't autoscale.

With llm-d you still need a gateway and a VIP underneath it. That tier is the one we replace — which is why our box count goes down where theirs goes up.

STATUS

What ships today — and what doesn't yet

AVAILABLE TODAY

Model-name routing with wildcard pools

KV-cache-aware routing — vLLM & SGLang

P/D disaggregation over NIXL

CHWBL & GPU-aware selection

OpenAI-compatible SSE + token accounting

MCP gateway, session-sticky

mTLS for AI backends, OPA L4 policy

CLI · TUI · web UI · Prometheus + Grafana

IN PROGRESS

Data-plane auth enforcement

API keys and tenant rate limits are control-plane CRUD today — managed, but not yet rejecting 401/403/429 in the request path.

Locality metrics gaps

LMCache hit/match rates and per-endpoint KV block-utilization are not exported yet — do not build panels on them.

EXPERIMENTAL

Adaptive routing controller

TTFT-aware, capacity-weighted prefill selection for heterogeneous fleets. Off by default.

LMCache tiering

A CPU KV tier beneath the GPU cache. No official compat matrix — gate it.

FOR THE PEOPLE WHO CARRY THE PAGER

Routing is a third of it. This is the rest.

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT KOREA

CLI

Scriptable, swagger-traceable

Every command and flag traces to the gateway's swagger spec

Full AI surface: model routing, P/D, KV-exact, CHWBL, API keys

restore plans by default — --commit applies it

JSON / wide output for automation

loxicmd-inference-gateway

TUI

GitOps in a terminal

11 live panels; logs tailed *through the API* — no SSH to the box

Declarative YAML for 38 resource kinds: validate · diff · apply · rollback

Auto-snapshot before every apply — rollback is always safe

NDJSON audit trail, token-redacted, rotated

loxitui-inference-gateway

WEB UI

Multi-operator, multi-tenant

RBAC: admin / operator / viewer, with capability-gated actions

JWT with server-side session revocation

AI Gateway pages, not just classic L4/L7

한국어 · English · 日本語

loxilb-ui

METRICS

The silent failures, made loud

A deliberately small dashboard: **12 panels, 3 rows** — a page, not a museum

Row 1: KV hit rate · subscriber uptime · in-flight · 5xx

Per-stage Tier-1.5 latency — tokenize · hash · cgo · scan

~50 series; 4 alerts ship, more documented to add

Prometheus + Grafana

Remember the fail-open design: a broken feed costs you nothing but silence. The alert that catches it ships as Critical — kv_subscriber_connected == 0 for 2 min. A hit-rate-low alert is documented as the next one to add. A silent failure mode is only acceptable if it is loudly observable.

TEN MINUTES, IF YOU ALREADY RUN VLLM

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT KOREA

Try it tonight

1 — RUN THE GATEWAY

```
docker run -u root --cap-add SYS_ADMIN --privileged -dit \ -v /dev/log:/dev/log -v /opt/loxilb/config:/etc/loxilb --name loxilb \ ghcr.io/loxilb-io/loxilb-inference-gateway:latest
```

2 — STAGE THE TOKENIZER (slug = model name, / → __)

```
wget -O /etc/loxilb/tokenizers/Qwen__Qwen3-0.6B/tokenizer.json \ https://huggingface.co/Qwen/Qwen3-0.6B/raw/main/tokenizer.json
```

3 — POST THE RULE, THEN WATCH IT FIRE

```
curl -s http://<vip>:11111/netlox/v1/config/loadbalancer -d @rule.json curl -s "http://<vip>:11111/netlox/v1/config/ai/kv/inventory?service_id=0&ep_idx=0" # total climbing → the subscriber is ingesting. Send the prompt twice.
```

0% hits on the second request? It is one of the three parity legs — seed, hash algo, block size. That is the first thing the troubleshooting page checks.

WHAT WE WANT FROM THIS ROOM

Three things we can't do alone

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT KOREA

01

Break the ladder

Run it on a fleet that isn't ours — heterogeneous GPUs, weird prompt mixes, long multi-turn — and tell us where a tier makes the wrong call.

→ open an issue with your trace

02

Extend a surface

vLLM and SGLang are wired; TGI is the same surface. Or write a controller — the gRPC contract is frozen and open, and it cannot break the data path.

→ inventory plane, or loxilb.aictrl.v1

03

Challenge our constants

ε = 0.75, λ = 32, the adaptive anchors at L=16 and L=26 — calibrated on *our* testbed. Those should be community-validated, not vendor-asserted.

→ run the methodology, publish the delta

APACHE-2.0 · ISSUES AND PRS WELCOME

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT KOREA

Come break it, then tell us how.

GATEWAY SOURCE

github.com/loxilb-io/loxilb-inference-gateway

DOCS

loxilb-io.github.io/loxilbdocs-inference-gateway

OPERATOR TOOLING

loxicmd-inference-gateway · loxitui-inference-gateway · loxilb-ui

SLACK

loxilb.io/members

Seokhwan Kong

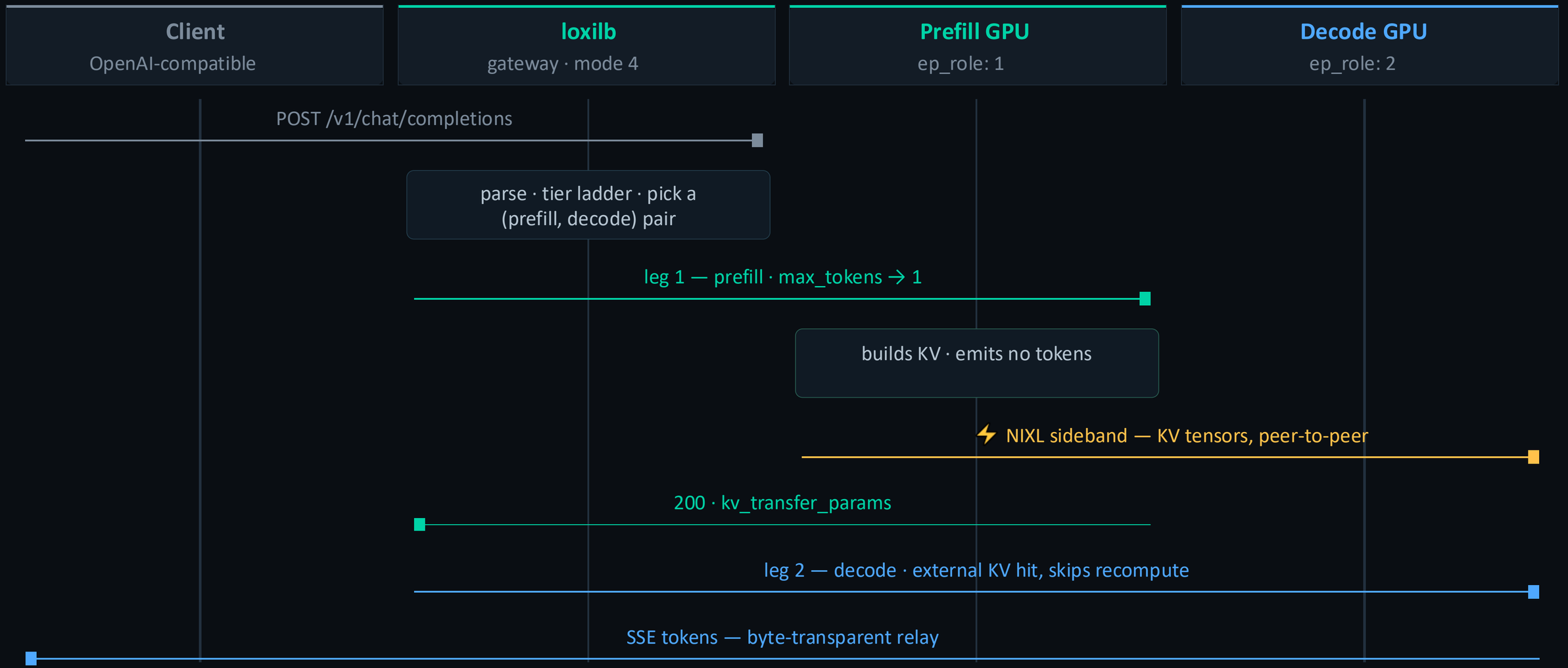
CO-FOUNDER, NETLOX

DEMO

P/D DISAGGREGATION

Two HTTP legs — and the KV takes a different road

time flows down gigabytes never touch the gateway



The tensors bypass us entirely — we orchestrate two HTTP legs. That is what makes this viable in a load balancer rather than a serving framework.

Gotcha: decode's conn counter stays 0:0 — it is gateway-originated. Read decode from vLLM's metrics.

WHY YOU SHOULD BELIEVE THE LAST SLIDE

Five gates stand between a run and a number

$$\text{good} \iff \text{TTFT} \le \text{SLO} \text{ AND } \text{TPOT} \le \text{SLO}$$

A **conjunction** — an OR would score a request that blew its TPOT as "good" because its TTFT was fine. Failed requests are **misses in the denominator**, never discarded.

PHASE 0
Health
A real 200 through the VIP — not an open port

PHASE 1
Calibrate
Drive one endpoint directly. Never the VIP

PHASE 2
Probe
Peak÷mean proves the mode mechanically engaged

PHASE 3
Replay
Same trace, same load, one variable per arm

PHASE 4
Analyze
BCa bootstrap, TOST for parity — overlapping CIs are not parity

PHASE 5
Verify
N ≥ 5 and CV ≤ 10%, else ADD_REPLICATES

P/D vs aggregated. We run disaggregated, vllm-router aggregated — the light-load gap mixes disagg overhead with routing quality.

Paired windows only. The fleet is non-stationary. Never compare our number from one window to theirs from another.

State the SLO and the rate. A routing policy that wins at one offered rate can lose at another. A goodput without its rate is not portable.