KubeCon CloudNativeCon North America 2026

#KubeCon #CloudNativeCon

What the Docs Don't Tell You

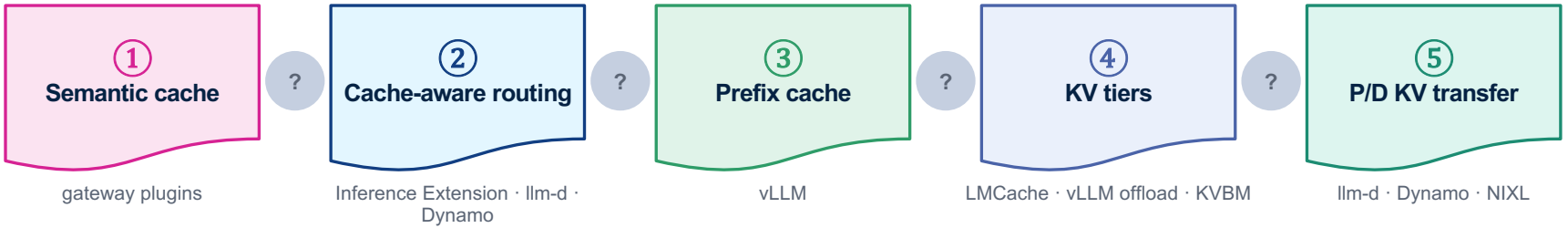
A Telecom's Year Operating Cache-Aware LLM Inference on Kubernetes

YoungJoon Jeong, AWS · Seol Heo, LG Uplus



Every cache has a doc. The stack has none.

Each project documents its own layer. Nobody draws them on one request path.



One request crosses all five. Which one sets your hit rate?

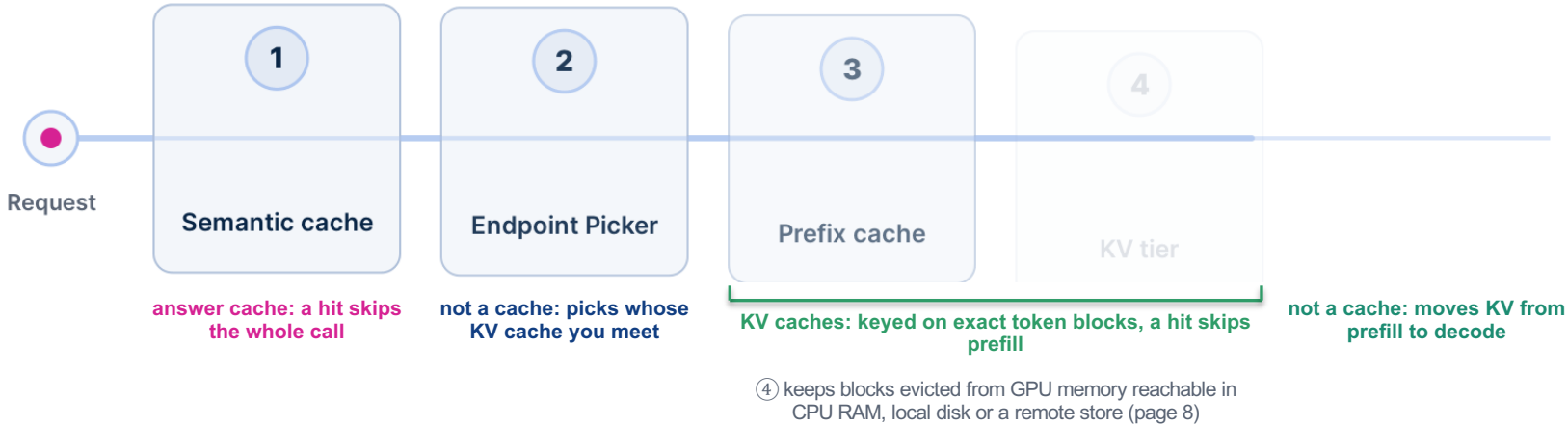
THIS TALK

1. **How the caches fit** — YoungJoon · 9 min
2. **How LG U+ cut TTFT and passed 95%** — Seol · 13 min
3. **The playbook and what's next** — YoungJoon · 5 min

KV hit rate 71.6% → 95.4%
TTFT 5.34 s → 0.39 s
LG U+ measured

One request, five layers that decide its cache hits

Only the first stores answers. The router stores nothing: it picks whose KV cache you meet.



Report the answer-cache hit rate separately: those requests never reach the engine.

Every agent turn resends the whole conversation

So most of each prompt is a prefix the stack has already seen once.



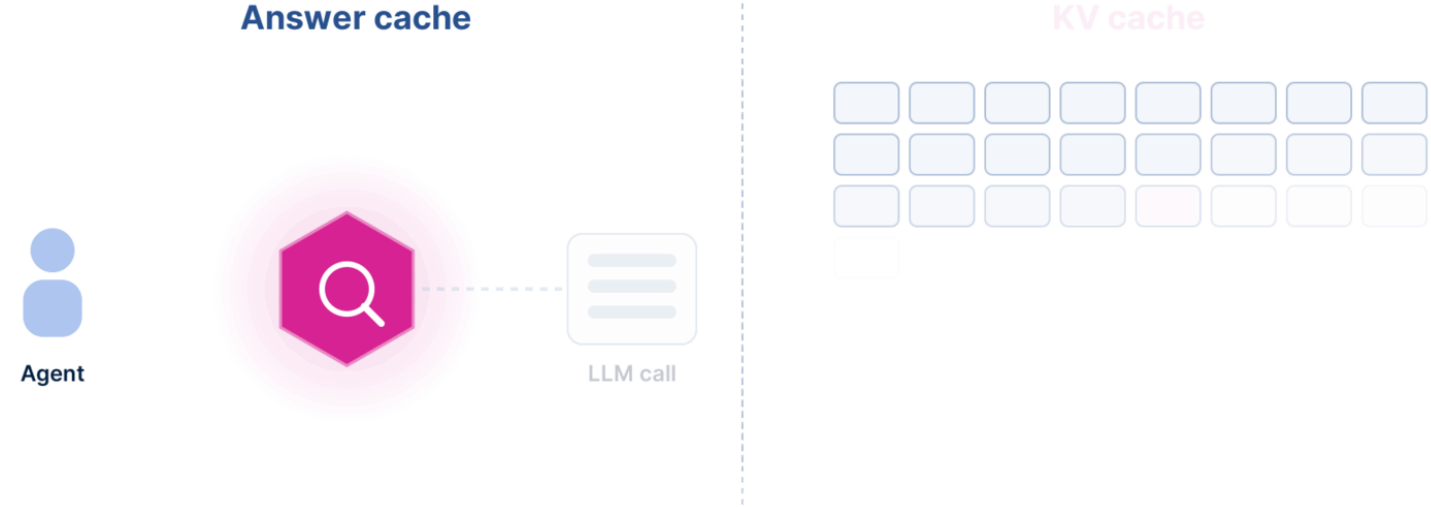
Call n resends calls 1 … n−1 and adds one new turn
most agent harnesses resend it, tool calls and outputs included

Only the new turn needs prefill
if the repeated part is still cached, and cached where the request lands

That repetition is the opportunity. How much of it repeats is the ceiling (page 10).

Two caches: answers by meaning, KV by exact tokens

An answer hit skips the model call. A KV hit skips prefill for the blocks it already holds.

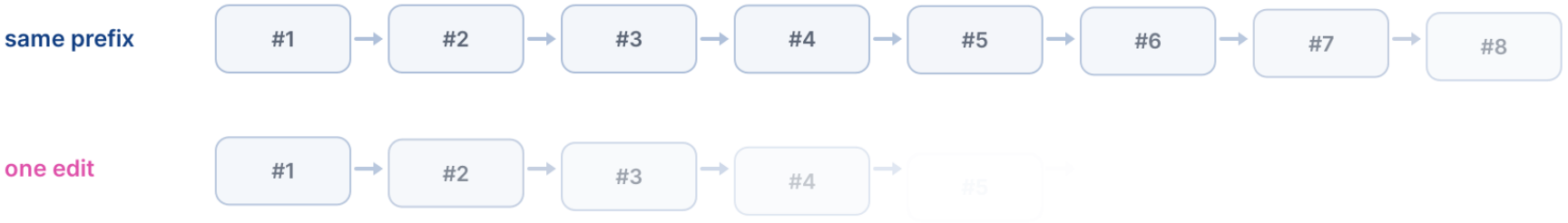


lives in the gateway · embed → match → return
a near-miss in meaning can return a wrong answer

lives in the engine · GPU first, then the CPU tier
exact match only: the same answer, computed faster

A prefix hit is a hash-chain match

vLLM caches full blocks, and each block's hash includes its parent's.



1

system

```
hash(block n) = H( hash(block n−1), tokens of block n, extras )
```

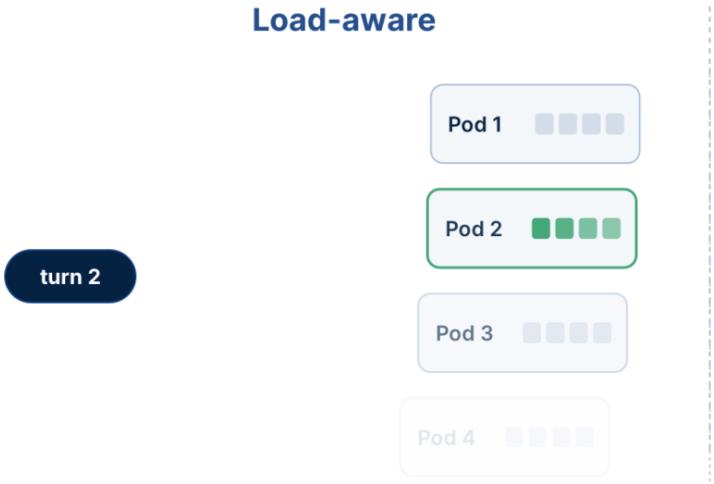
■ Only full blocks are cached

≠ Another LoRA or cache salt: a separate cache

Prompt layout sets the ceiling: stable content first, append-only after.

Routing decides which replica's cache you get

With N replicas and load-only routing, the next turn lands on the previous turn's replica about 1 time in N.





Load-aware
fails when sessions are long

Approximate prefix-aware
llm-d default: routes where the prefix went before; drifts after evictions and restarts

Precise prefix
costs an event stream and an index

All three still weigh load. Too little affinity copies one session onto several pods (Cliff 1, page 15); too much turns the warm pod into a hot spot.

llm-d Router plugins: approx-prefix-cache-producer · prefix-cache-affinity-filter · token-load-scorer

When GPU memory fills, a CPU tier keeps the blocks

An evicted block moves to DRAM. A later turn loads it back: a slower hit instead of a miss.



Survival
the share of repeated tokens still cached when the request arrives (page 10)

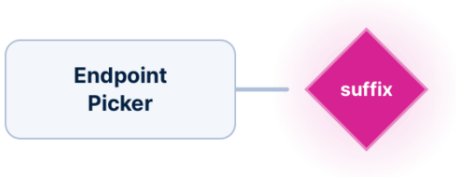
How
vLLM OffloadingConnector or LMCache copy blocks to CPU RAM; the llm-d tiered prefix cache guide wires it up

Let the router see it
with self_describing_kv_events, CPU-tier blocks reach the router's index too

At LG U+, a 240 GB DRAM tier took the KV hit rate from 87.7% to 95.4% (page 18).

Split prefill from decode, and let the cache decide

The router picks a decode replica first, then asks how much of the prompt that replica does not hold.



P **Prefill**
Compute-bound. Prefix affinity keeps groups warm.

D **Decode**
Latency-bound. Decode TP ≥ prefill TP with NIXL.

↔ **KV transfer**
RDMA strongly recommended. TCP is for development.

llm-d: disagg-profile-handler + prefix-based-pd-decider

Higher hit rate → shorter uncached suffix → fewer turns need the prefill pool.

Hit rate = ceiling × placement × survival

Three factors with three owners. To reach 95%, they must average above 98% (98.3%).

01

0.32%

Ceiling, LG U+ measured

×

02

About 0.0%

Placement times survival, calculated

=

Ceiling
prompt layout · app and agent teams

Placement × survival
routing (inference gateway) and KV tiers (serving platform)

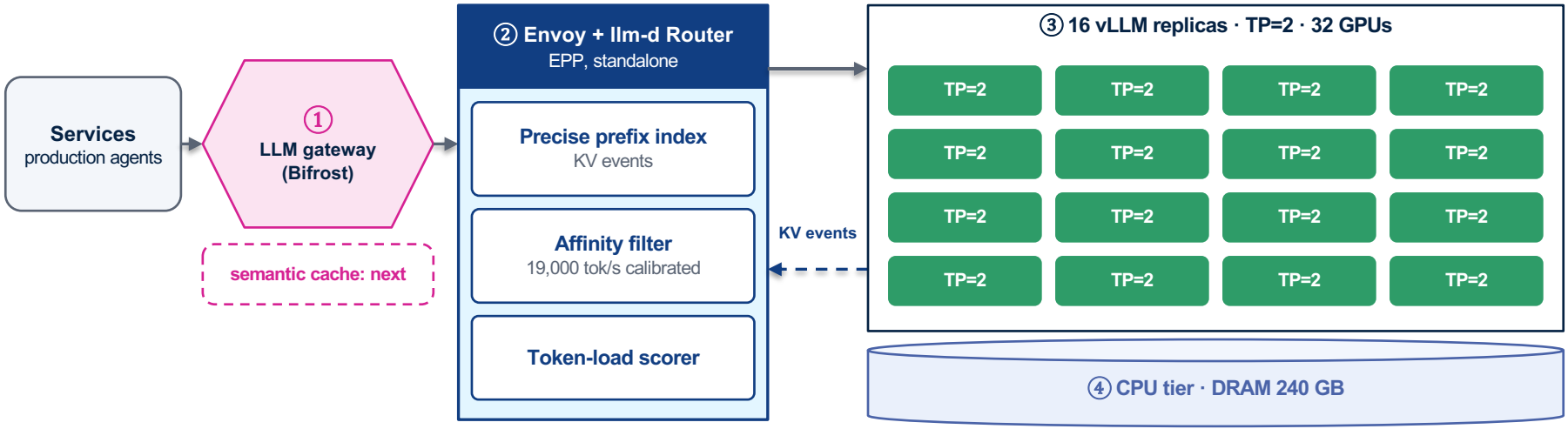
KV hit rate
the product of all three

Missed 4.6 pts: 3.04 from the prompt, 1.56 from routing and capacity (calculated)

TBU · Confirm ceiling and 95.4% were measured on the same traffic
Seol · due 11/05

The stack behind the numbers

One gateway, one router, 16 vLLM replicas and a CPU tier: the setup behind Act 2.



TTFT = queue wait + prefill · caching cuts prefill, slots cut the queue

TBU · Confirm GPU type, and whether 240 GB is per pod or per node
Seol · due 11/05

KubeCon CloudNativeCon North America 2026

Act 2

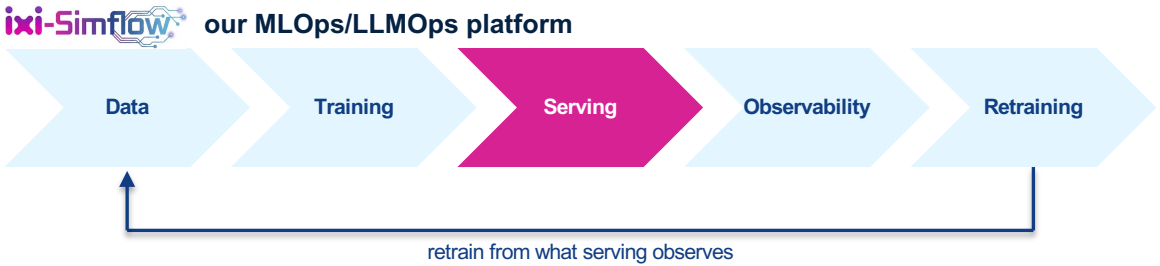
What production taught us



Seol Heo · LG Uplus · 13 min

Why we chase TTFT

Model developers own answer quality. The platform team owns latency: time to first token.



Answer quality
model developers

Latency (TTFT)
platform team · this talk

Telecom customers expect low latency. A nationwide AI service raised the bar.

Day one: 16 replicas behind a load balancer

The LLM gateway (Bifrost) load-balanced 16 vLLM replicas directly. Fine until users surged.



fine at first → users surged → TTFT rose

TTFT = queue wait + prefill

segment widths are illustrative



prefix cache → shorter prefill
cache hits, pages 15 to 18

execution slots → shorter queue wait
max-num-seqs, page 19

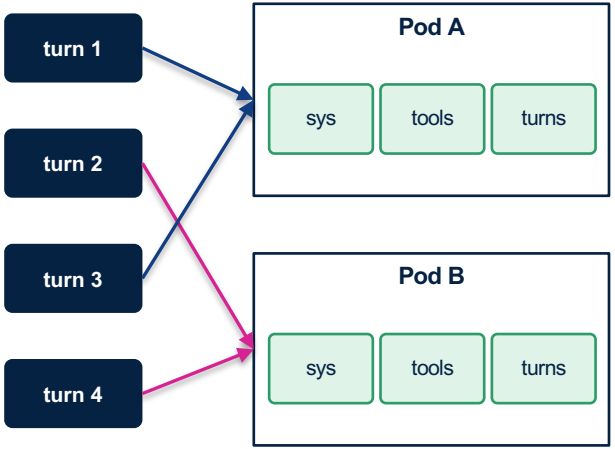
prefix cache + execution slots → Shorter TTFT
12.5 s → 0.39 s

Cliff 1: when HBM overflows

The llm-d Router came first, still balancing on load: one session landed on two pods.

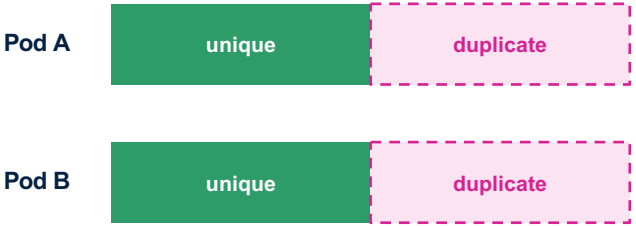
71.6%
KV hit rate

12.5 s
TTFT p95



llm-d Router · load-aware by KV-cache utilization

HBM per pod



The same prefix blocks sit on both pods.

Each session cached twice: half the HBM

Cliff 2: when the prefix breaks

A config change and a prompt change broke prefix matching. The ceiling caps what any routing can reach.

Block size changed → hit rate fell sharply

The hybrid model's engine block changed from its automatic 784 tokens, and every baseline moved.

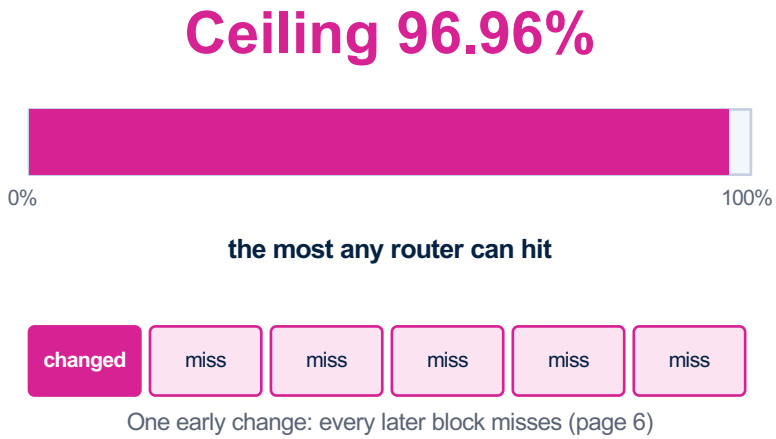
Fix Keep the engine default. The router keys on a divisor of it (16).

TBU · hit rate
Before → After

Working directory in the prompt → every session's prefix differs

First requests were 1,924 tokens, but only 1,568 were shared across runs.

Fix (recommended) Put anything that varies at the end of the prompt.



Precise routing alone cut TTFT by 45%

Only the routing policy changed: load-aware by KV-cache utilization, then precise prefix on KV events.

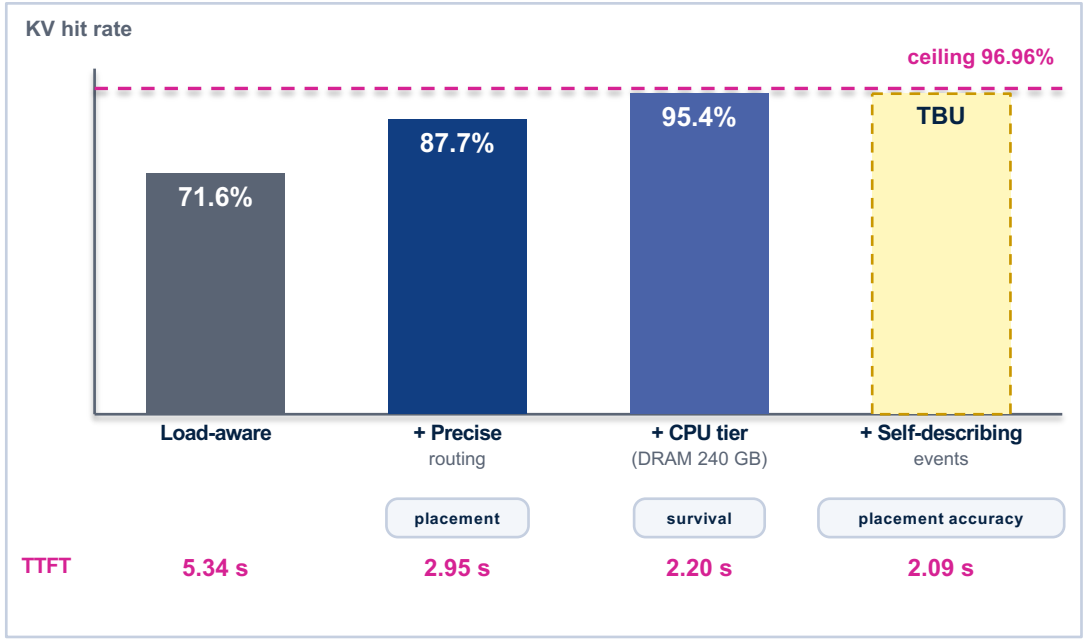
0.6%



KV hit rate 71.6% → 87.7% at the same step (page 18) · TTFT statistic being confirmed (page 27)

The climb to 95%

Routing raised placement, the CPU tier raised survival, self-describing events sharpened the index.



Precise routing alone
TTFT −45% (calculated)

Self-describing KV events
EPP prediction 63% → 82% · pod switches per turn 24% → 16%

Misses
28.4% → 4.6% of prompt tokens (calculated)

Then the queue: execution slots

Once prefill was cached, the next lever was the queue: how many sequences run at once.



TTFT 2.09 → 0.39 s
`max-num-seqs → 24`
−81%, slots step (calculated)

`max-num-seqs 16 → 24`
Preemption 0 → 0
Inter-token latency 23.5 → 24.3 ms

fewer slots — more slots

Too few slots: requests wait, TTFT rises

Too many: KV pushed out, preemptions, inter-token latency rises

24: tuned by watching preemptions

Signals we watch: TTFT p95 · KV usage · preemptions · ITL · EPP prediction accuracy · pod switches

Prefill/decode split: results

The P/D configuration under test on the same stack, and what it changes.

TBU · P/D topology
Seol · due 11/05

prefill replicas · TP
?

→

KV transfer
?

→

decode replicas · TP
?

TBU · TTFT and ITL vs aggregated
Seol · due 11/05

TBU · KV transfer time
Seol · due 11/05

TBU ms
prefill end → decode start

Where LG U+ landed

Measured by LG U+ on 16 vLLM replicas behind a precise prefix router.

01

0.4%

KV hit rate, up from 71.6%

routing, CPU tier, self-describing
page 18

02

0.00s

TTFT, down from 5.34 s

caching to 2.09 s, then 24 slots
pages 18 and 19

100% − hit rate, if it is token-level
definition, page 27

Each step had its own doc. The order is what the docs did not tell us.

KubeCon CloudNativeCon North America 2026

Act 3

The 95% playbook



YoungJoon Jeong · 5 min

The 95% playbook

Raise each factor with the layer that owns it, and prove every change before it ships.

C Ceiling
prompt

- Keep per-session values (working directory, time) out of the prefix
- Re-baseline after any block-size change
- Measure your ceiling (ours: 96.96%)

Application · agent harness

P Placement
routing

- Precise prefix routing from KV events
- Calibrate prefill throughput on your model (ours: 19,000 tok/s)
- Turn on self-describing KV events

Inference Extension · llm-d Router

S Survival
capacity

- Don't let load balancing duplicate sessions
- Tier KV to DRAM so eviction is not a miss
- Size max-num-seqs by watching preemptions

vLLM · CPU offloading

PROVE IT

Compare every routing change on hit rate and TTFT p95 before it ships.

Where we go next

Two are LG U+'s next steps; the rest build on what this talk measured.



Item 1: a gateway hit never reaches the engine
count it apart from KV hits · counters are illustrative

1	**Semantic caching at the gateway** Report its hit rate apart from the KV hit rate	LG U+ next
2	**Small/large model cascading** An escalation prefills again on the large model: keep its cache warm	LG U+ next
3	**P/D at scale** Build on the page 20 results	in progress
4	**Routing policy as a gated rollout** Canary, compare hit rate and TTFT p95, roll back	our plan
5	**Share back upstream** Calibration data, block-size keying, alert rules	upstream

Checked against upstream docs on 2026-10-30 and 2026-11-04; sources on page 28.

KubeCon CloudNativeCon
North America 2026

Thank you

Please rate this session

Scan the code to leave feedback.

Hit rate = ceiling × placement × survival. Own all three.
Then tune slots for the queue.

Questions

YoungJoon Jeong · Seol Heo

TBU · links: manifests (D1), report (D4)

What the Docs Don't Tell You: A Telecom's Year Operating Cache-Aware LLM Inference on Kubernetes



Backup: sources

LG U+ (Seol Heo): data and EPP config (2026-11-04)
email to the speakers, not public

vLLM KV offloading usage: self_describing_kv_events (checked 2026-11-04)
github.com/vllm-project/vllm/blob/main/docs/features/kv_offloading_usage.md

llm-d precise prefix-cache routing guide and router values (checked 2026-11-04)
github.com/llm-d/llm-d/tree/main/guides/precise-prefix-cache-routing

llm-d P2P KV cache sharing guide (checked 2026-11-04)
github.com/llm-d/llm-d/tree/main/guides/p2p-kv-cache-sharing

llm-d tiered prefix cache guide (checked 2026-10-30)
github.com/llm-d/llm-d/tree/main/guides/tiered-prefix-cache

llm-d P/D architecture (checked 2026-10-30)
github.com/llm-d/llm-d/blob/main/docs/architecture/disaggregation/pd-disaggregation.md

vLLM prefix caching design (checked 2026-10-30)
docs.vllm.ai/en/latest/design/prefix_caching.html

Gateway API Inference Extension v1.6.2 (checked 2026-10-21)
gateway-api-inference-extension.sigs.k8s.io

llm-d v1.0.0 release (2026-10-20)
github.com/llm-d/llm-d/releases/tag/v1.0.0

LMCache README (checked 2026-10-30)
github.com/LMCache/LMCache

NVIDIA Dynamo README (checked 2026-10-30)
github.com/ai-dynamo/dynamo