Weights and Measures

What it takes to serve an LLM on Kubernetes

Márton Enyedi

KCD Budapest · 30 October 2026

origoss SOLUTIONS

Who I am, and what Origoss does.

Márton Enyedi

DevOps engineer at Origoss
since February

Two months ago, I had never run a language model myself.

Origoss Solutions

Budapest, since 2018

We design, build, and operate
open-source platforms.

Kubernetes, and the AI platforms that run on it.

Vendor-independent.
For Hungarian and European clients.

A quick show of hands.

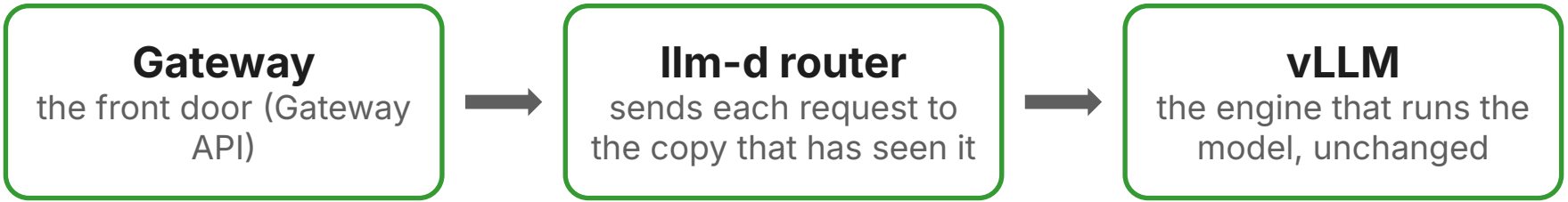
Who runs Kubernetes at work?

Who has used vLLM?

Who has heard of llm-d?

Who has run a language model on their own GPU?

I was asked: what is llm-d, and what does it take to run an LLM on Kubernetes?



llm-d is a traffic controller for LLMs, built from ordinary Kubernetes resources.

Deploying it took a day.

Knowing what the numbers meant took three weeks.

You already have most of the skills to run an LLM on Kubernetes.

Cost control, capacity planning, storage, caching, alerting.

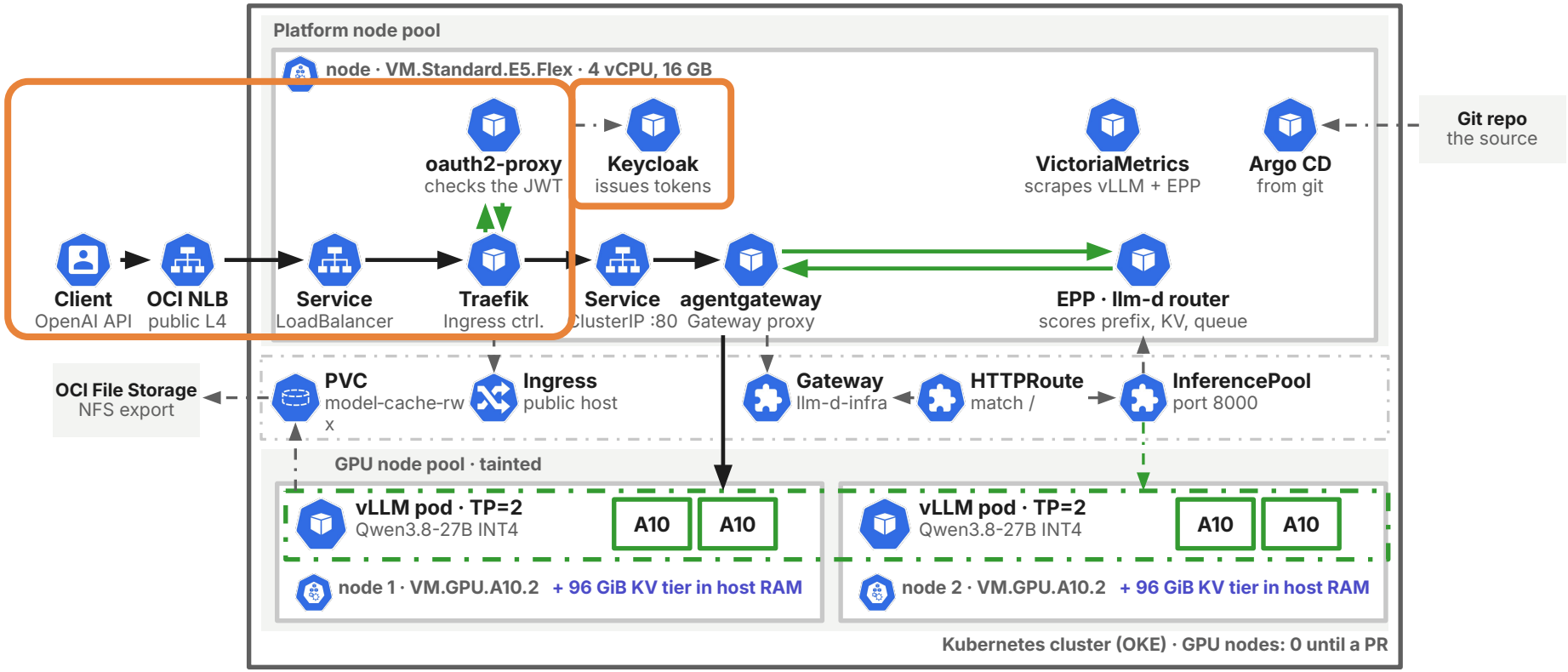
Only one skill is new: checking the answers.

The road here: two months from a simulator to a 27B model on four GPUs.



Each step raised the next question: can I afford it, what fits on the card, how fast does it start, how much does it serve, how do I watch it, and can I trust its answers?

Where the road ended.

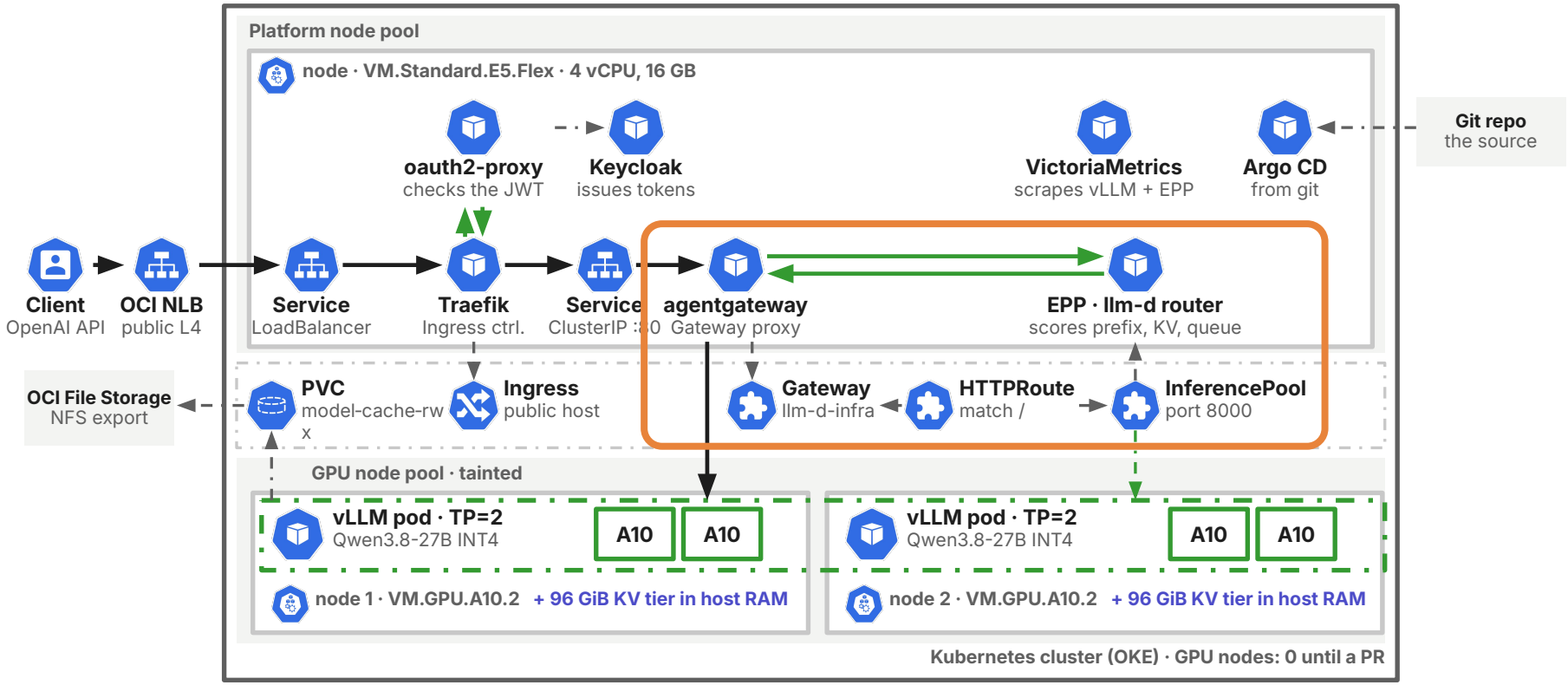


→ request ⇄ side call on the path → refers to dashed band: API objects · dashed green: pods the InferencePool selects

Icons: kubernetes/community, CC BY 4.0

origoss SOLUTIONS

Where the road ended.

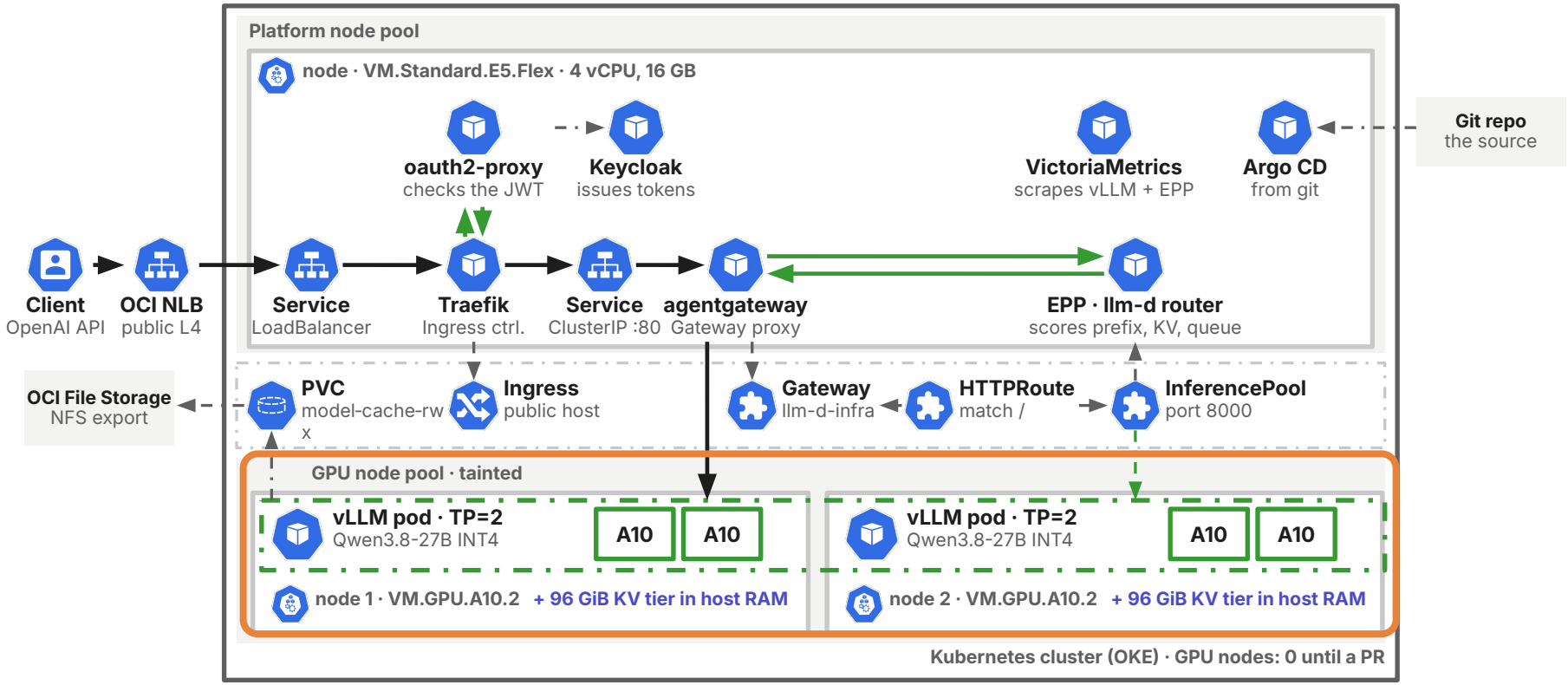


→ request ⇄ side call on the path → refers to dashed band: API objects · dashed green: pods the InferencePool selects

Icons: kubernetes/community, CC BY 4.0

origoss SOLUTIONS

Where the road ended.

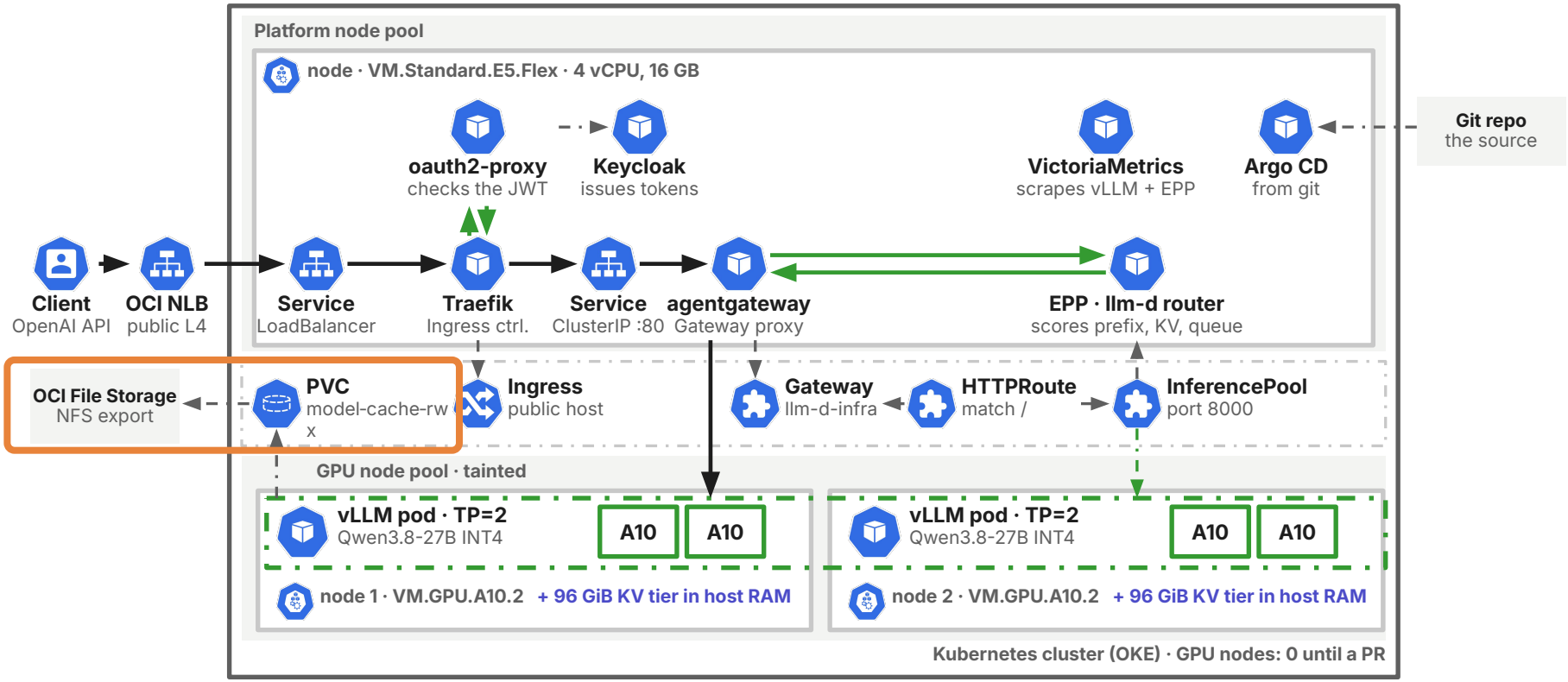


→ request ⇄ side call on the path → refers to dashed band: API objects · dashed green: pods the InferencePool selects

Icons: kubernetes/community, CC BY 4.0

origoss SOLUTIONS

Where the road ended.

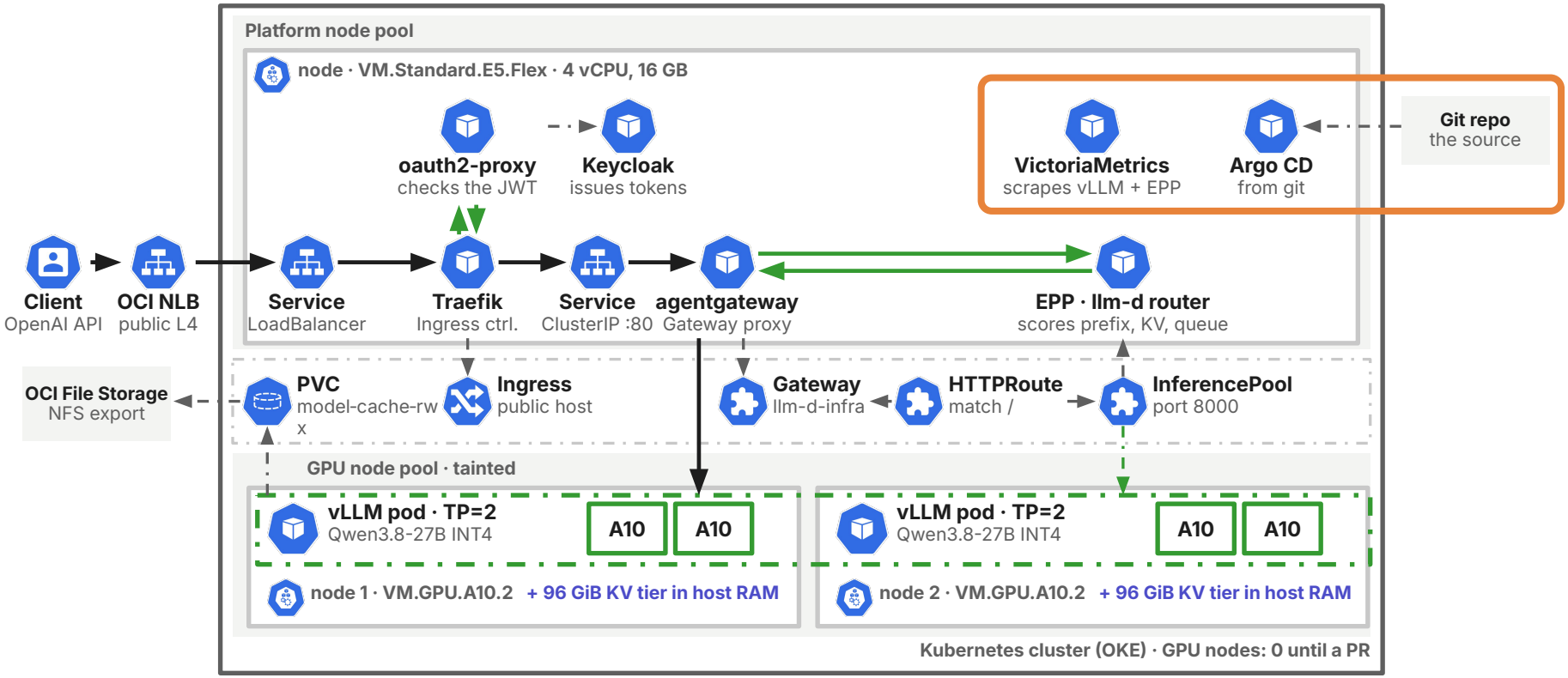


→ request ⇄ side call on the path → refers to dashed band: API objects · dashed green: pods the InferencePool selects

Icons: kubernetes/community, CC BY 4.0

origoss SOLUTIONS

Where the road ended.



→ request ⇄ side call on the path → refers to dashed band: API objects · dashed green: pods the InferencePool selects

Icons: kubernetes/community, CC BY 4.0

origoss SOLUTIONS

Question 1 of 6: Can I afford to experiment?

Afford	**Can I afford to experiment?**
Fit	What fits on the card?
Start	How fast does it start?
Serve	How much does it serve?
Watch	How do I watch it?
Trust	Can I trust its answers?

GPUs bill for existing, not for computing, so the pool sits at zero until a PR summons it.

$2.00

per GPU-hour, one NVIDIA A10 on OCI

$48

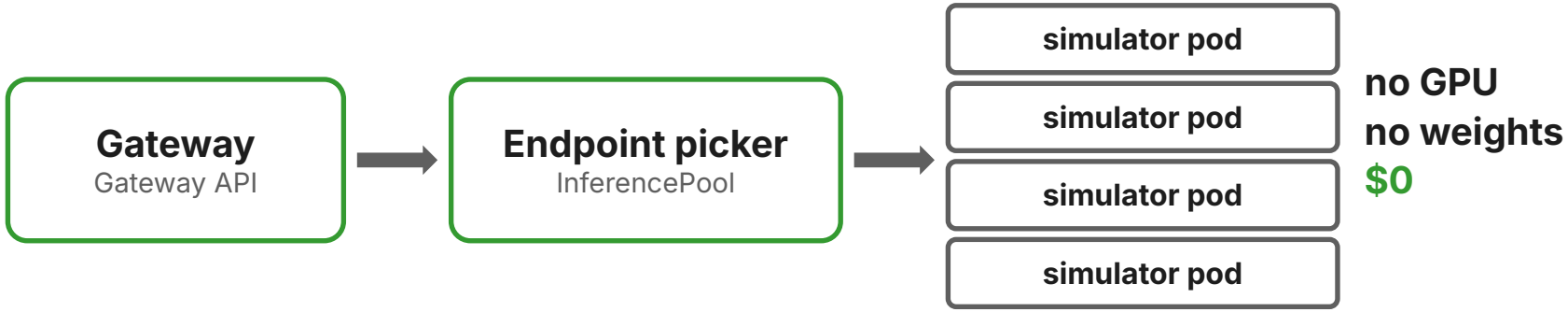
per day for one node left on by mistake

0

GPU nodes while parked; a pull request summons them

OCI list prices: VM.GPU.A10.1 (one GPU) $2.00/hour, VM.GPU.A10.2 (two GPUs) $4.00/hour.

Everything except the GPU was built against a simulator first.



19 / 20

the repeated prompt landed where it was cached

0 / 20

different prompts: nothing to reuse

Proves the routing works. Says nothing about speed: simulated latency is configuration.

Question 2 of 6: What fits on the card?

Afford	Can I afford to experiment?
Fit	**What fits on the card?**
Start	How fast does it start?
Serve	How much does it serve?
Watch	How do I watch it?
Trust	Can I trust its answers?

Two words for the rest of this talk.

KV cache

The model's short-term memory.
While reading a prompt, it saves its work on every token.
It lives in GPU memory, next to the model.
Each user at once needs their own KV cache.

Time to first token

How long the user looks at an empty screen before the answer starts.
The number users actually feel.

One NVIDIA A10 · 24 GB · Qwen3-14B, 8-bit



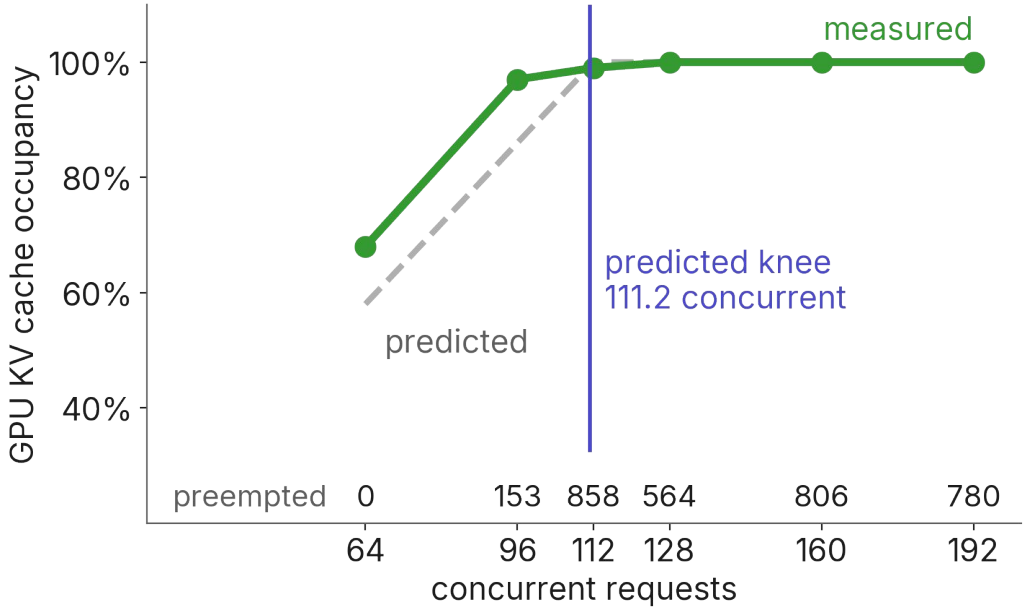
Weights first. Whatever is left becomes KV cache.

Smaller weights leave more KV cache, and more KV cache serves more users.

Model	Weights	On one 24 GB A10	What we did
Qwen3-14B, 8-bit	~15 GB	fits, little room left	first model served
Qwen3-14B, 4-bit	~9 GB	fits, 2.7× the room	served from 14 Sep
Qwen3.8-27B, 4-bit	**16.5 GiB**	**fits, no room for KV cache**	**split across two GPUs**

Splitting one model across two GPUs (tensor parallelism) stores the weights once: **+44% KV cache** per GPU pair. It stays inside one machine: there is no fast link between these VMs.

Each user needs their own KV cache, so capacity is a division.



KV cache ÷ KV cache per request

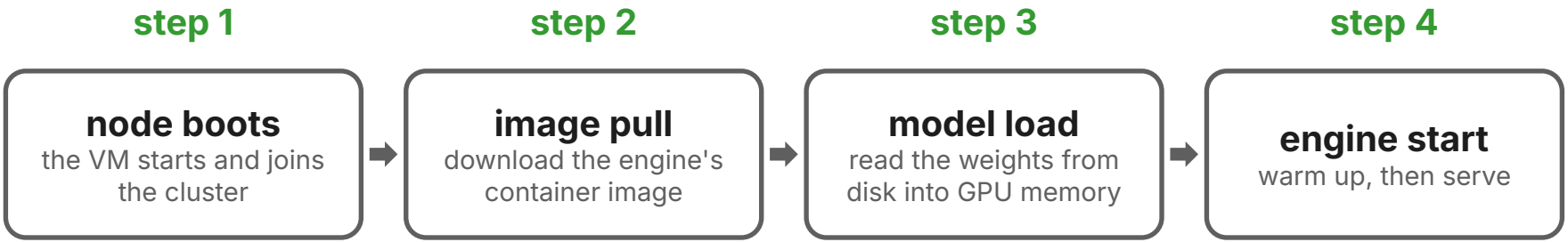
```
2 × 128,128 tokens
÷ 2,304 per request
= 111 users
```

Predicted 111, measured 112. Past that, the first word took 27.7 s instead of 3.8 s. The sum ran ~15% optimistic: plan with headroom.

Question 3 of 6: How fast does it start?

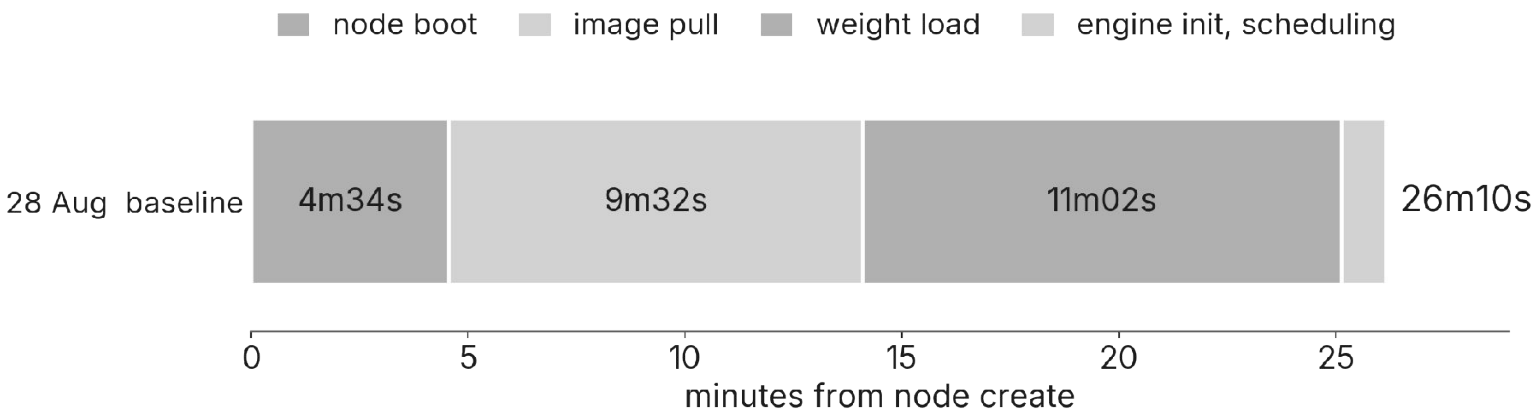
Afford	Can I afford to experiment?
Fit	What fits on the card?
Start	**How fast does it start?**
Serve	How much does it serve?
Watch	How do I watch it?
Trust	Can I trust its answers?

What happens on a cold start: four steps, and the GPU bills from the first.



The GPU is paid for from step 1. It serves nothing until step 4.

The first cold start took 26 minutes, and most of it was disk I/O we could fix.



Qwen3-14B-FP8 · 1 × A10 · 28 Aug 2026

Bigger disks, then one shared copy of the model: about five minutes.

2m44s

loading the model, was 11 min: on this cloud, a bigger disk is a faster disk

2m59s

pulling the image, was 9½ min: the same trick on the boot disk

~5 min

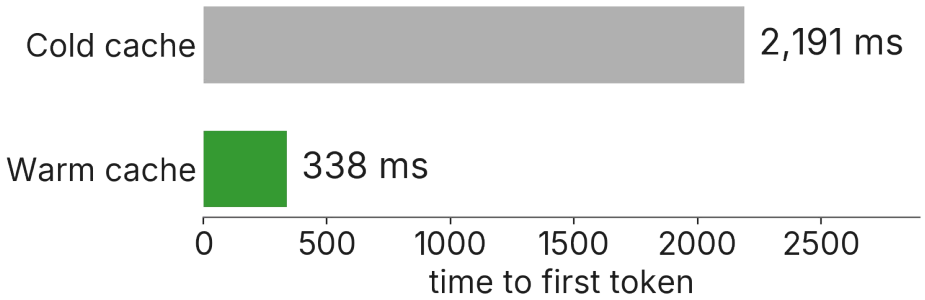
from nodes up to serving, today: the 27B model on shared file storage

Every fix moved the bottleneck somewhere else: profile the cold start in order.

Question 4 of 6: How much does it serve?

Afford	Can I afford to experiment?
Fit	What fits on the card?
Start	How fast does it start?
Serve	**How much does it serve?**
Watch	How do I watch it?
Trust	Can I trust its answers?

The prefix cache is the biggest lever: same prompt start, 2191 ms → 338 ms.



→ The llm-d router sends repeats to the engine that already holds their KV cache.

On today's 27B setup: 1,020 ms → 687 ms.

6.5×

sooner first word when the start repeats

Put the stable part first:
system prompt, documents, tool definitions.

Question 5 of 6: How do I watch it?

Afford	Can I afford to experiment?
Fit	What fits on the card?
Start	How fast does it start?
Serve	How much does it serve?
Watch	**How do I watch it?**
Trust	Can I trust its answers?

Two places for the KV cache: GPU memory and host RAM.

On the GPU

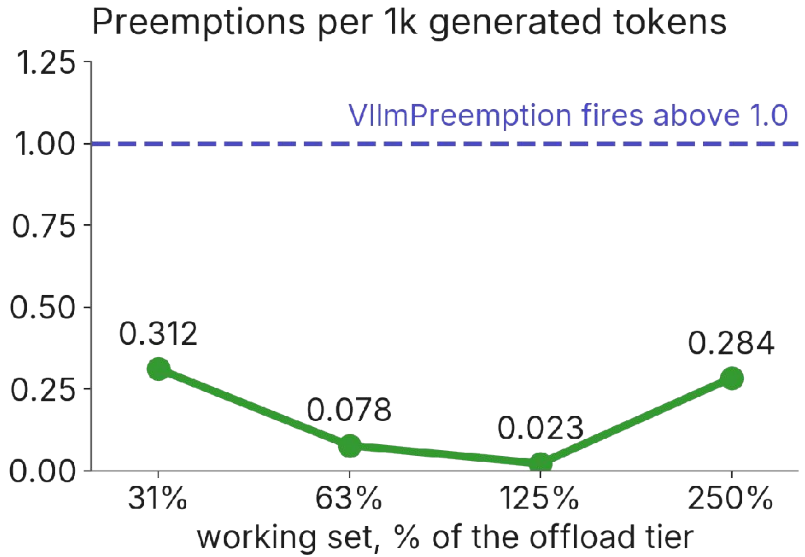
Fast, but small.
When full, the engine pauses some requests. Those pauses are the usual overload signal.

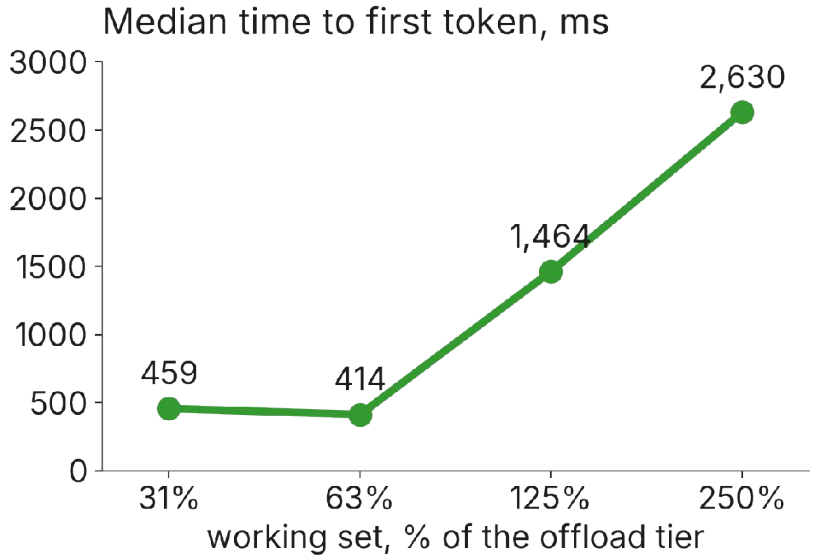
Offload tier, in host RAM

Bigger, but slower: 96 GiB per engine here.
When the GPU part is full, KV cache moves here instead of pausing requests.

With an offload tier, nothing pauses, so the overload signal stays low. Users can still wait.

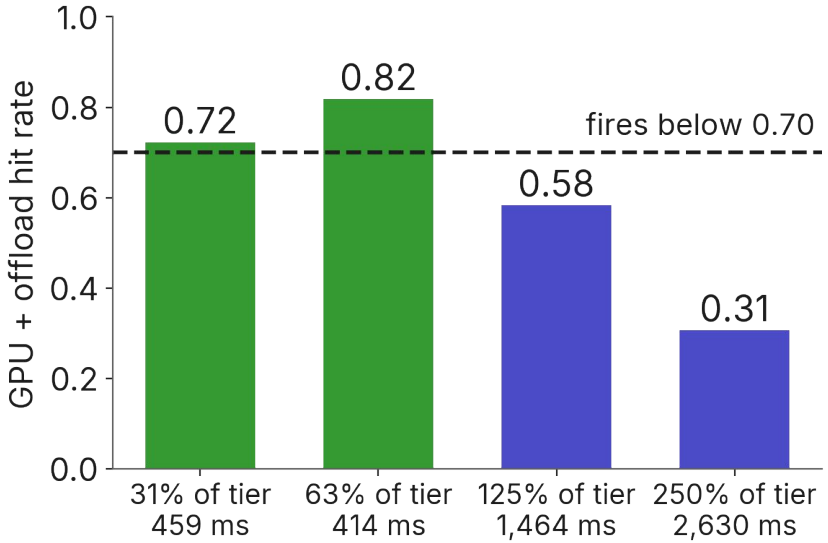
With an offload tier, a cache running out of room preempts less, not more.





Same number of users throughout; only the amount of different text grows. Left: the overload signal never reaches its alarm line. Right: the wait for the first word.

Alert on what users feel: how often the KV cache gets reused.



healthy 0.82 · slow 0.58 · bad 0.31

Alert below 0.70 for 15 minutes,
but only when there is real traffic.

Without that last condition it would fire all night, while the GPUs are parked.

```
(hits + external_hits) / queries < 0.70
and queries > 1e5 over 10m, for: 15m
```

Question 6 of 6: Can I trust its answers?

Afford	Can I afford to experiment?
Fit	What fits on the card?
Start	How fast does it start?
Serve	How much does it serve?
Watch	How do I watch it?
Trust	**Can I trust its answers?**

Every speed-up has a price somewhere.

4-bit weights
quantized: each weight stored in fewer bits

2.7× more users
cost: less precision

quality: −3.6 points on a maths test. Hold that thought.

Two GPUs per model
tensor parallelism

+44% KV cache · first word 4.8 s vs 6.3 s
cost: twice the GPUs per engine

quality: same model, same answers

Speculative decoding
a fast guesser drafts, the big model checks

15% faster
cost: 27% less KV cache

quality: every guess is verified

Speed and capacity are easy to measure. Quality is the one most teams skip.

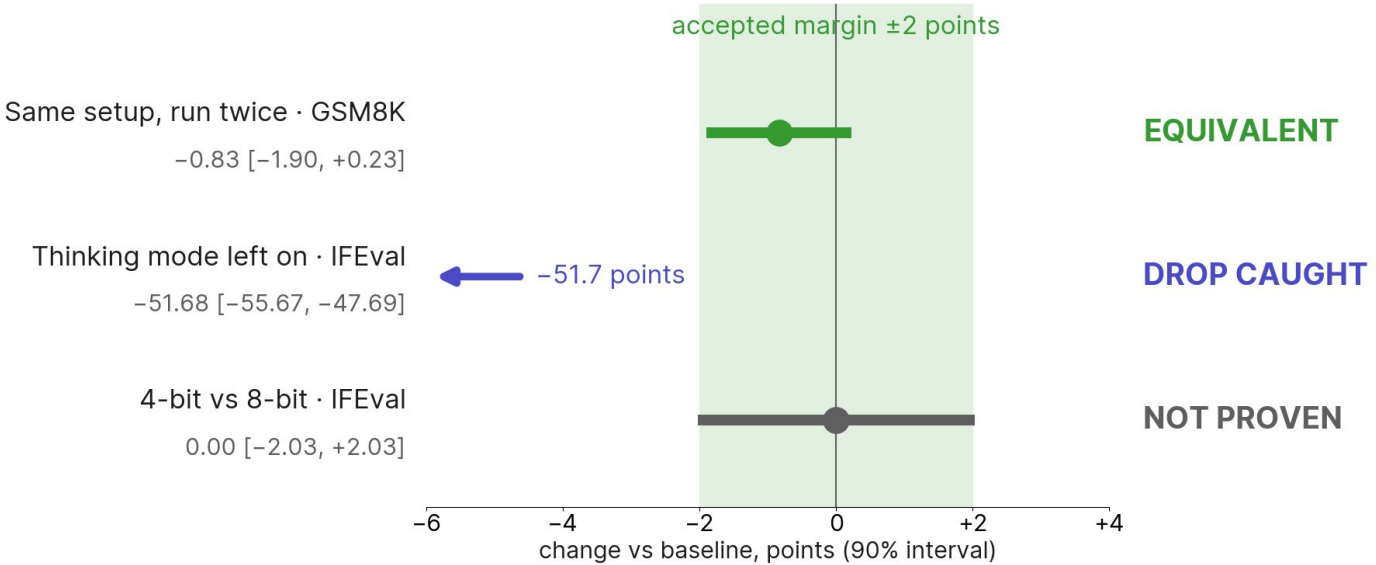
Don’t ask “is it the same?”. Ask “is it worse by more than 2 points?”

Like a kitchen scale: weigh the same thing twice, then something you know is heavier, then what you care about.

Step 1	Same setup, twice	how much it wobbles
Step 2	Thinking mode left on	can it see a real drop?
Step 3	Quantized: 4-bit vs 8-bit weights	the real question

Proving “no difference at all” would take ~90,000 test questions; public tests have a few thousand. The ±2-point margin was written down before the first run.

A difference of exactly zero, and the test still said “not proven”.



Same score, but each got 21 different questions wrong. A score table would print “identical”; with 541 questions, the test can’t tell.

Read the raw answers: 4-bit wasn't worse at maths, only at formatting.

A 60-mile bike trip; stops at mile 20 and 15 miles before the end. How far apart?

```
8-bit:   **45 miles - 20 miles = 25 miles**.
       #### 25
```

```
4-bit: 45 miles - 20 miles = <<45-20=25>>25 miles.
       **Answer: 25**
```

11.1% vs 6.1%

right answer, wrong last line, so the grader marked it wrong (4-bit vs 8-bit)

0.340 → 0.850

the grader was scoring the model thinking out loud; one flag fixed it

What it takes: six things I wish I had known on day one.

Afford	Most of the work needs no GPU: build it against a simulator first.
Fit	KV cache ÷ KV cache per request = users at once. Do the sum before you buy.
Start	A slow cold start is a storage problem: 23 of 26 minutes were disk, GPU idle.
Serve	Stable text first in the prompt, and a router that knows: 6.5× faster.
Watch	A full cache is normal. Alert when reuse drops: that’s when users wait.
Trust	Read the raw answers: the 4-bit “quality drop” was all formatting.

Thank you for your attention

Questions?

Márton Enyedi

KCD Budapest · 30 October 2026

origoss SOLUTIONS