



KubeCon



CloudNativeCon

Europe 2026

#KubeCon #CloudNativeCon

Route, Serve, Adapt, Repeat: Adaptive Routing for AI Inference Workloads in Kubernetes

Kellen Swain & Nir Rozenbaum



Speakers Intro



Kellen Swain

Sr Software Engineer
Google



Nir Rozenbaum

Sr Principal Software Engineer
Red Hat

By the end of the talk I hope you...

- Understand how IGW works today
- What makes IGW great/useful today
- What shortcomings do we have today
- How adaptive routing can help
- What other ways are being explored to solve these problems
- How you can get involved



KubeCon



CloudNativeCon

Europe 2026

What is GIE/IGW? Why do I care?

The Inference-specific router pitch

- The request body of an inference request contains data relevant to significant routing efficiencies, such as...
 - Prompt content (prefix routing)
 - Model name (LoRA)
- Leverage existing L7 implementations for reliable performance
- A global view of the model server state can provide operational benefits, multi-workload reuse, etc.



KubeCon

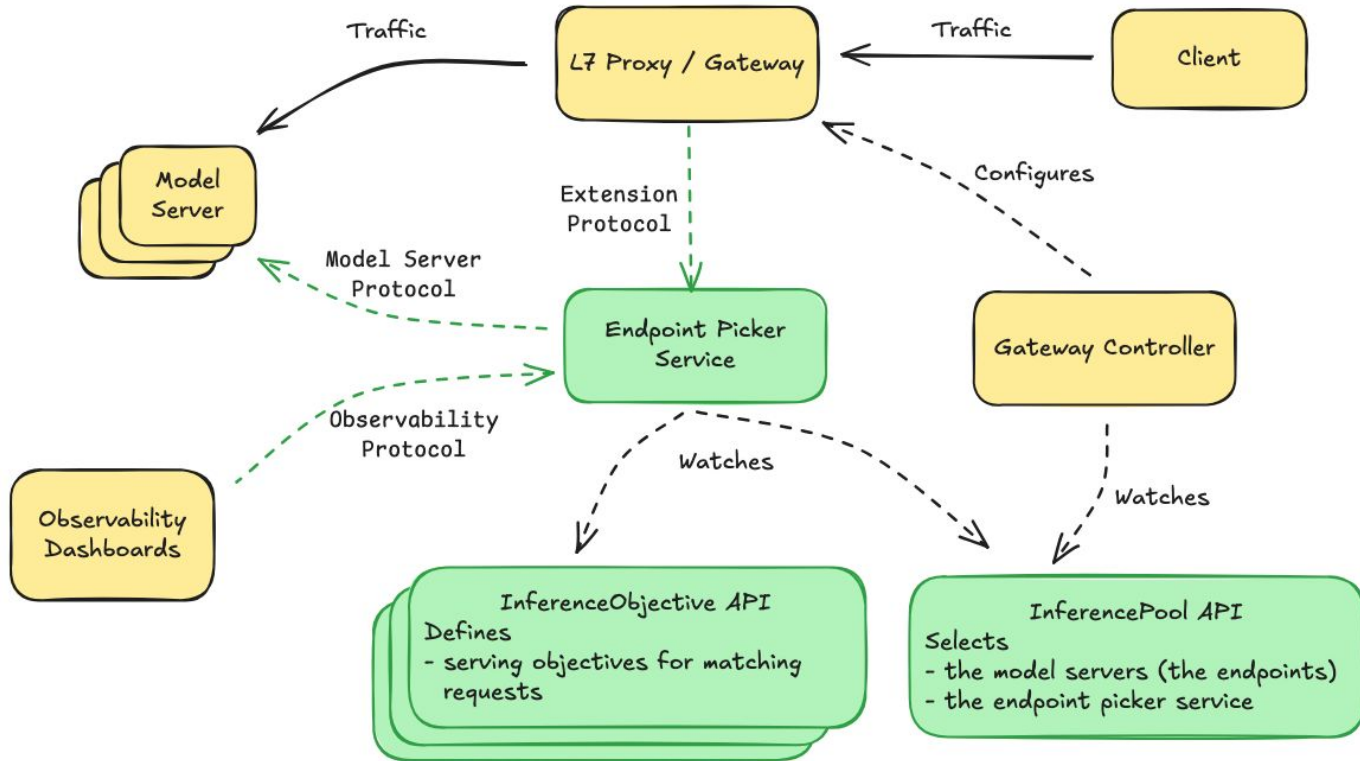


CloudNativeCon

Europe 2026

How does IGW integrate with gateways?

Gateway Integration





KubeCon

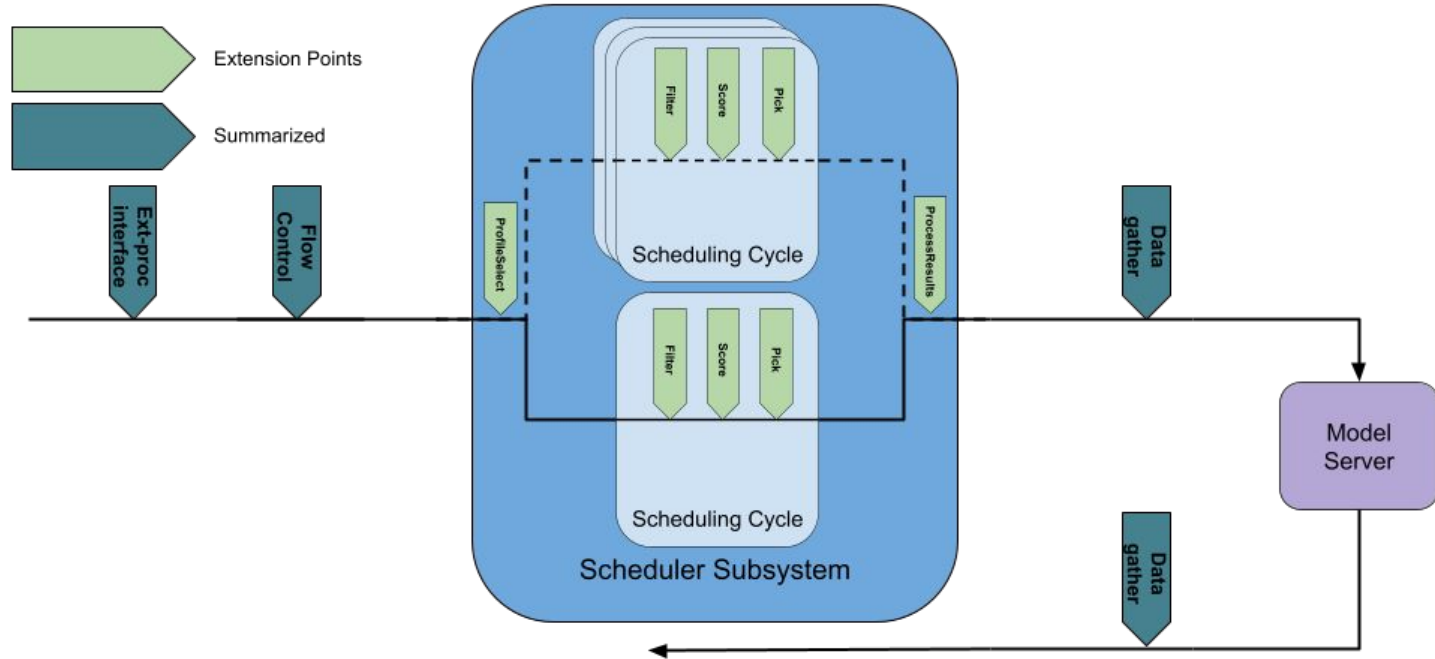


CloudNativeCon

Europe 2026

How does endpoint selection work within the EPP?

EPP Scheduler Framework





KubeCon



CloudNativeCon

Europe 2026

Inference Routing Crash Course

Prefix Aware routing

- Assume you have two prompts from two different users
 - “What is the weather like in Amsterdam?”
 - “What is the weather like in Phoenix?”
- Due to the properties of attention, we can reuse the generated KV-cache values, turning this into a memoization, or dynamic programming problem
- This then requires an load-balancer (EPP) to be aware of all the KV-cache states and route appropriately



KubeCon



CloudNativeCon

Europe 2026

Current perf improvement

Check with your doctor to see if IGW is right for you

Metric	k8s (Mean)	llm-d (Mean)	Δ (llm-d - k8s)	$\Delta\%$ vs k8s
Requests/sec	5.7306	14.4123	+8.6817	+151.5%
Input tokens/sec	43,417.86	109,192.83	+65,774.97	+151.5%
Output tokens/sec	5,362.16	13,574.37	+8,212.21	+153.2%
Total tokens/sec	48,780.02	122,767.19	+73,987.17	+151.7%
Request latency (s)	105.4133	67.8649	-37.5484	-35.6%
TTFT (s)	34.9145	0.1203	-34.7942	-99.66%
Inter-token latency (ms)	70.42	67.66	-2.76	-3.9%

Sourced from:

<https://github.com/llm-d/llm-d/blob/main/guides/inference-scheduling/README.md#benchmarking-report>



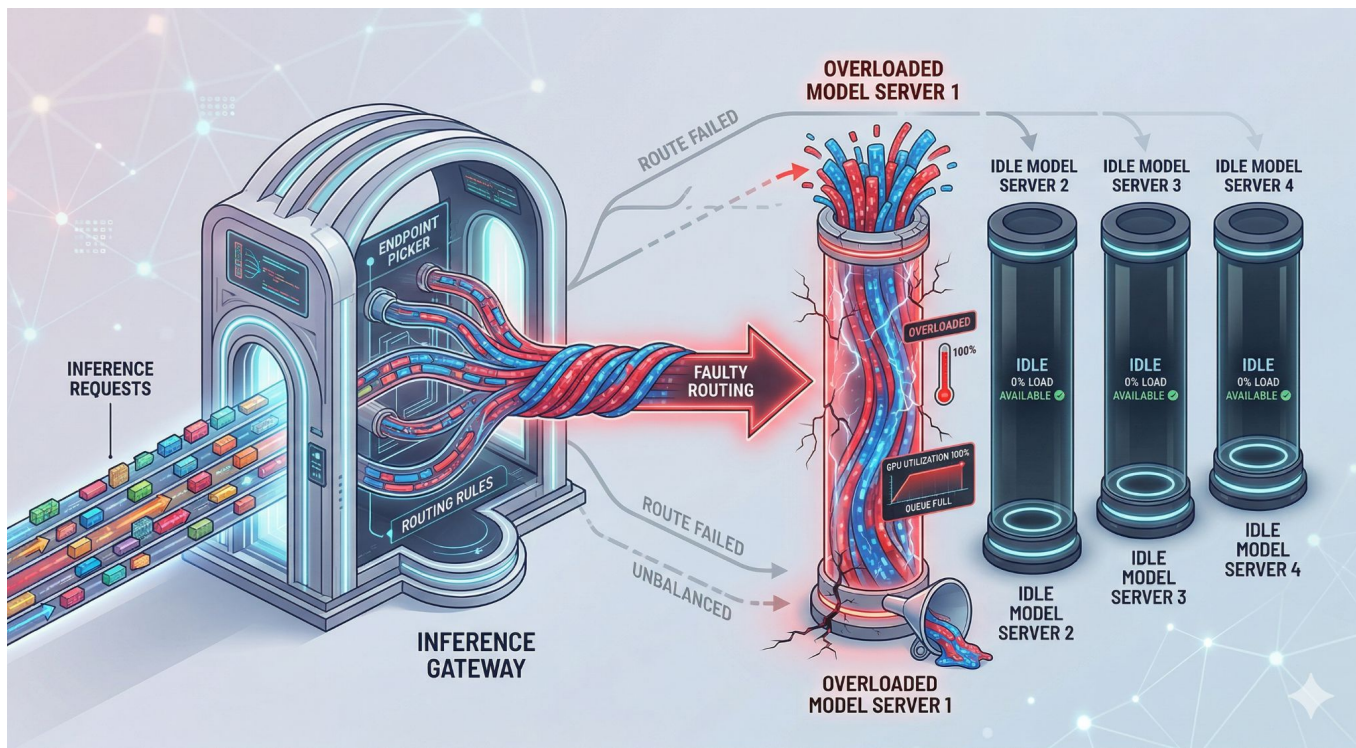
KubeCon



CloudNativeCon

Europe 2026

Pitfalls of the current system



Config-based Labyrinth

- Determining optimal config requires significant benchmarking
 - Tuning scorer weights
 - Selecting appropriate plugins
- Well-lit path algorithms may not cover a specific traffic pattern
 - The perfect weighting may fail when novel traffic patterns occur





KubeCon



CloudNativeCon

Europe 2026

How can we help?

Goals for end user UX

- Simple, low-overhead way for the most end users to reliably use our system with no changes to the default config
- Adding new, novel scorers does not require significant tuning to optimize
- The average user should not be required or expected to access our pluggable framework



KubeCon



CloudNativeCon

Europe 2026

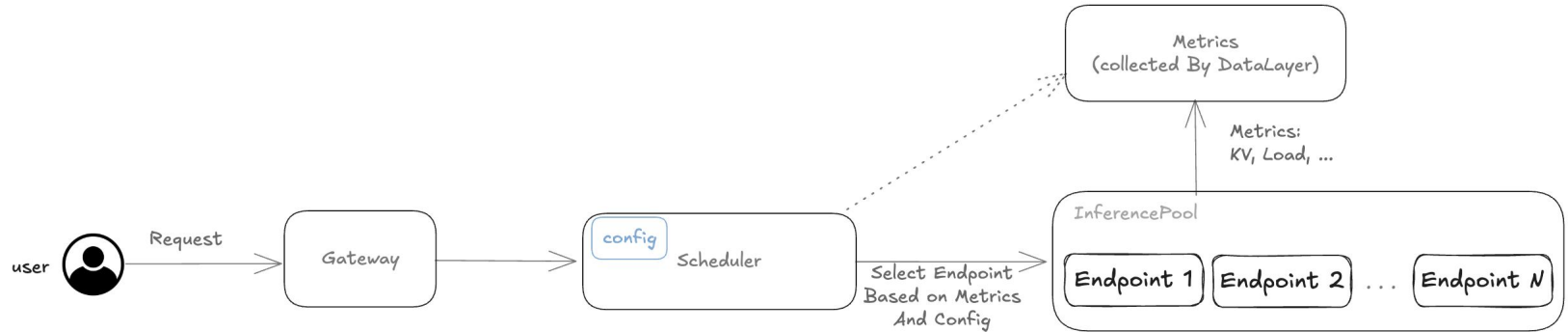
Adaptive Routing

Adaptive Routing

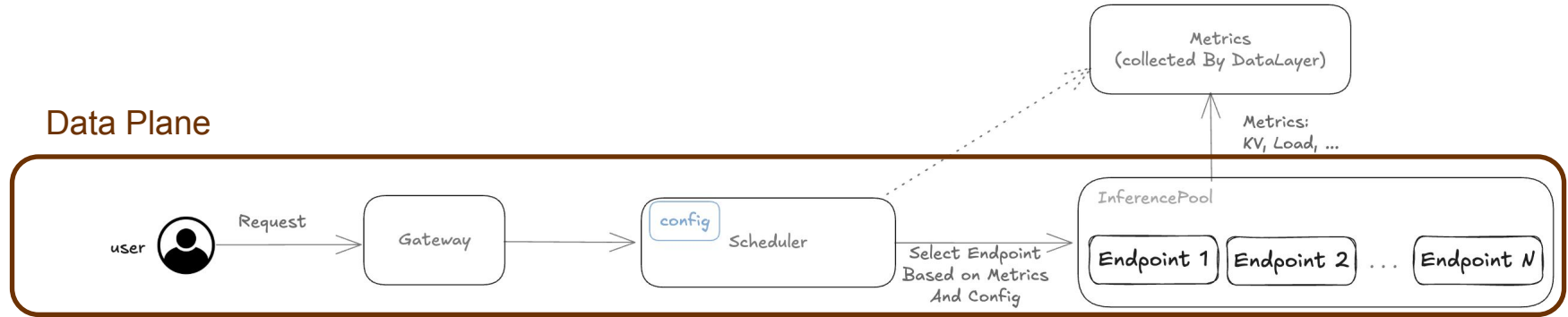
- Design Principles:
 - No request path changes.
 - No per-request overhead
 - Deterministic & explainable
 - Async + decoupled
 - Can be extended & customized

Before Adaptive Routing

- Request flow before adaptive routing

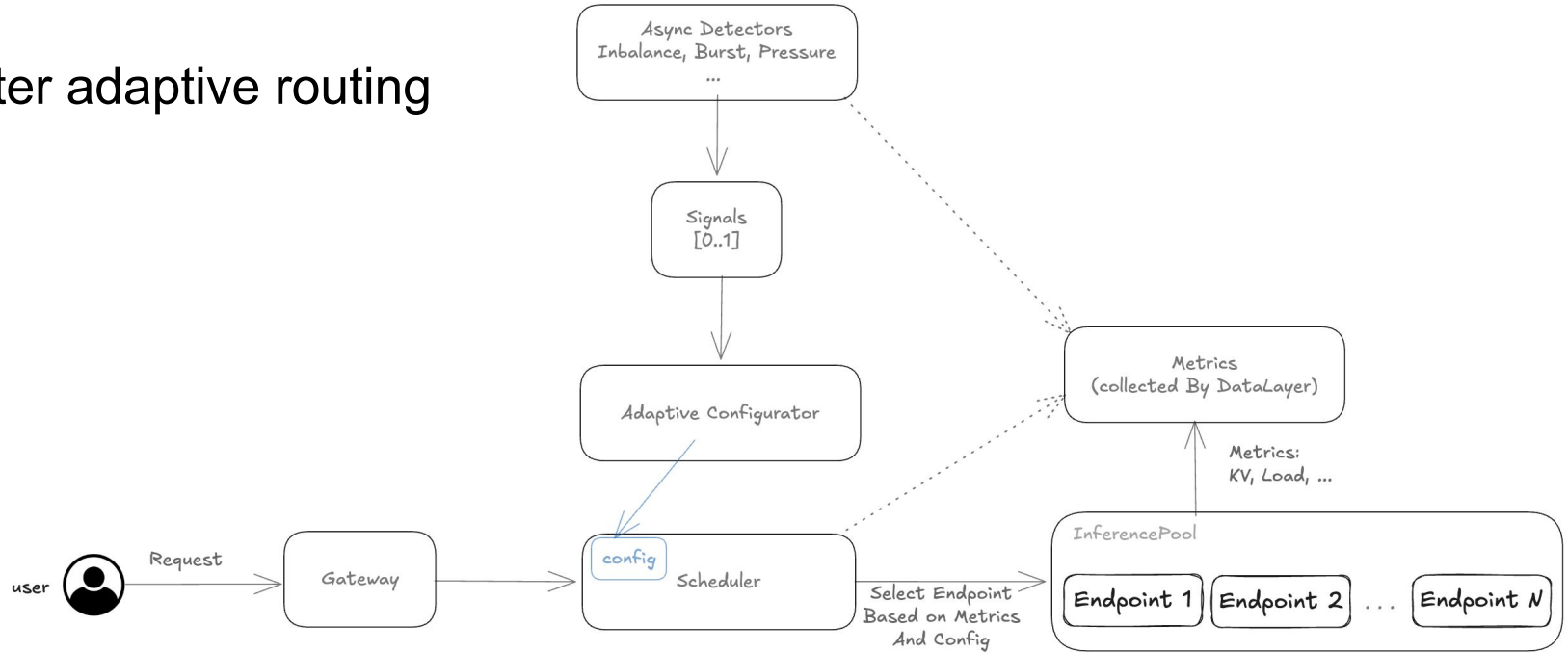


Before Adaptive Routing



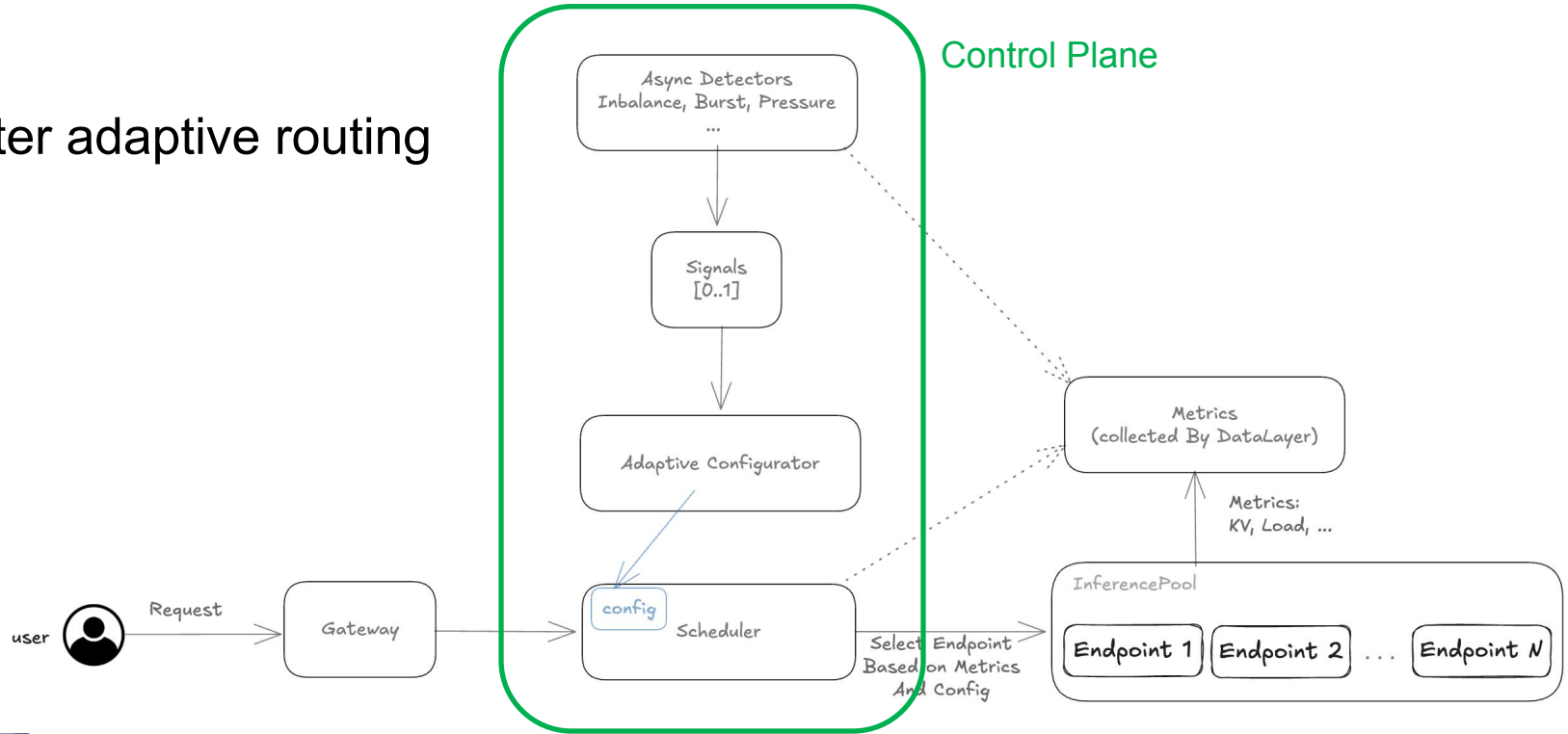
After Adaptive Routing

- After adaptive routing



After Adaptive Routing

- After adaptive routing



Concepts

- Two core abstractions
 - Async Detector
 - Adaptive Scheduler Configurator

Async Detector

- **Asynchronous**: periodically calculates a pool “state” (e.g., imbalance, burst) based on metrics.
- **Rate Limited**: runs at a fixed interval tied to metrics scraping cadence.
- **Stable**: produces a stable signal $\in [0,1]$.
 - smoothing
 - asymmetric thresholds for entering vs. exiting a state

Example: Imbalance Detector

- Reads per endpoint load metrics:
 - KVCache Utilization
 - Waiting Queue depth (Waiting Requests)
 - Running Requests
- Calculates pool imbalance using statistics methods.
- Emits a signal (detector doesn't take any scheduling decisions!)

Adaptive Configurator

- Periodically consumes Async Detector(s).
- Smooth and monotonic translation from a signal to new scorer weights.
- Dynamically reallocates scorer weight budgets (based on scorer category).

Affinity vs Distributive

Affinity-scorers

- Prefix match
- LoRA state

Distributive scorers

- KV-cache utilization
- Queue depth
- # running requests
- Etc..



KubeCon



CloudNativeCon

Europe 2026

Benchmarks

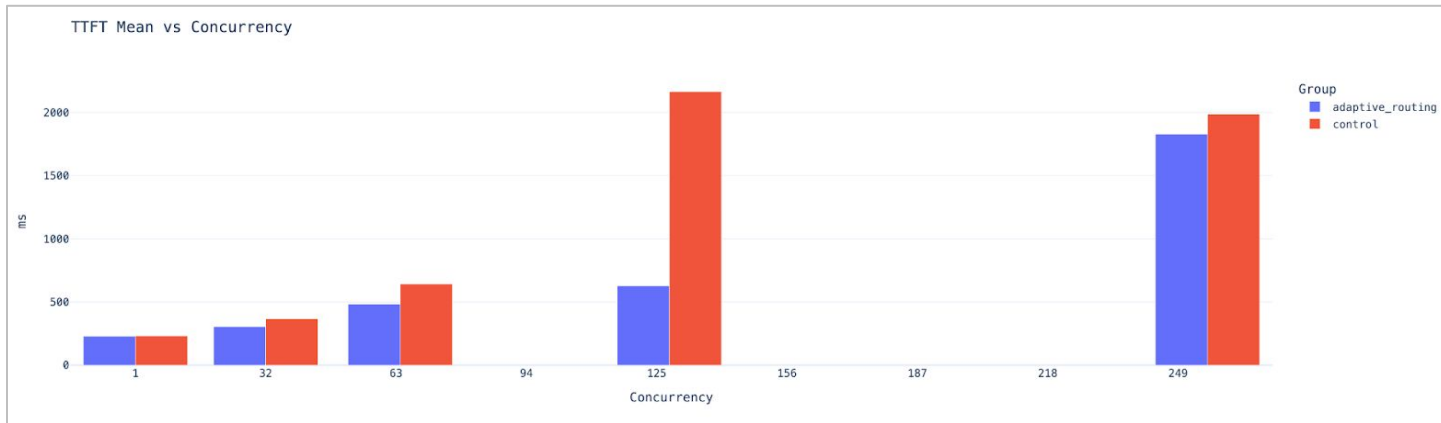
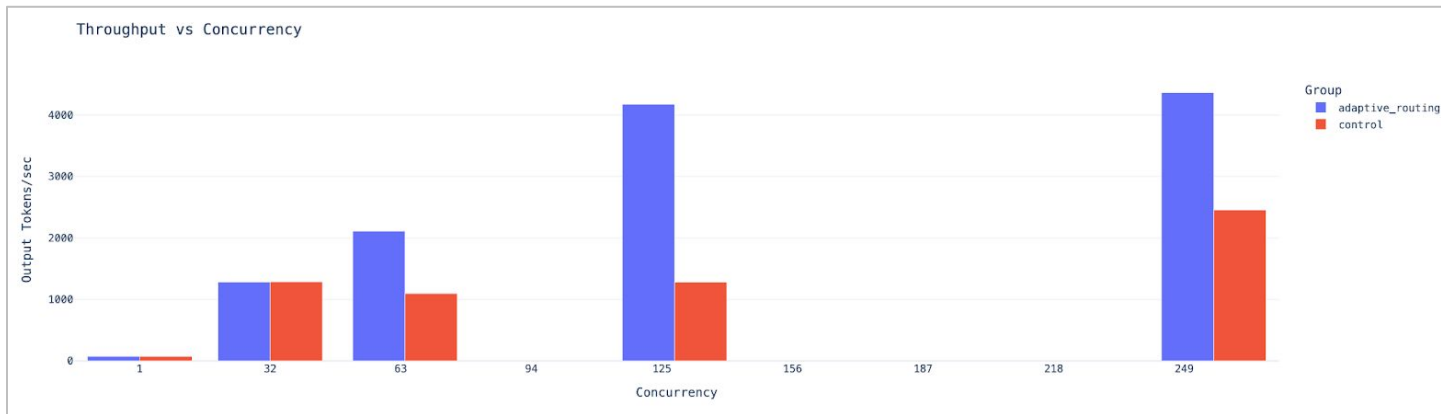
Benchmarks

- Initial benchmarks were done with **ImbalanceDetector** only.
- Imbalance Detector uses Normalized Coefficient Of Variation (CV) to calculate imbalance of the pool (reactive).
- Adaptive Configurator consumes it and uses generalized Sigmoid function to ensure it's monotonic and smooth.
 - Low imbalance: favor **affinity**
 - High imbalance: favor **distribution**

Hotspot Benchmark

- 8 endpoints / 8 GPUs A100 / llama 8b
- Guidellm
- Compared:
 - Static Configuration baseline (control group)
 - Adaptive (weights only)

Hotspot Benchmark



Ilm-d intelligent routing benchmark

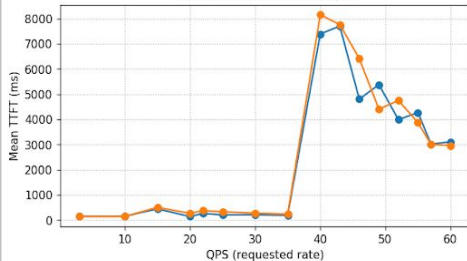
- 8 endpoints, 16 GPUs H100 (2 GPUs per replica), Qwen 32b
- Inference-perf
- Compared:
 - Static Configuration baseline (control group)
 - Adaptive (weights only)

Ilm-d intelligent routing benchmark

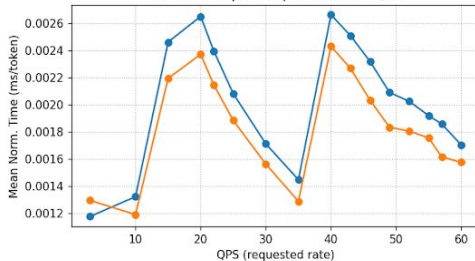
Latency vs Request Rate

control-group-runs
adaptive

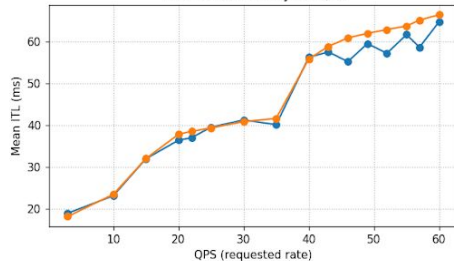
Time to First Token vs. QPS



Norm. Time per Output Token vs. QPS



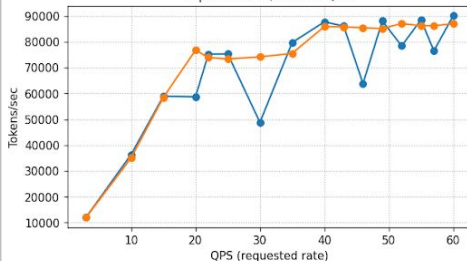
Inter-Token Latency vs. QPS



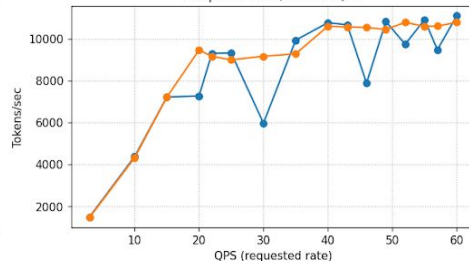
Throughput vs Request Rate

control-group-runs
adaptive

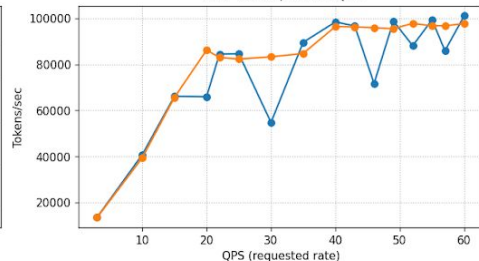
Input Tokens/sec vs. QPS



Output Tokens/sec vs. QPS

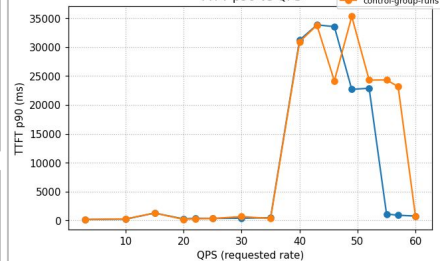


Total Tokens/sec vs. QPS



TTFT p90 vs QPS

adaptive
control-group-runs



KubeCon



CloudNativeCon

Europe 2026

Adaptive Routing - Summary

- Adaptive has comparable perf to the default IGW algo
- Fully async, zero request-path cost
- Smooth, explainable adaptation

Takeaway: Promising results, but there is still much to explore



KubeCon



CloudNativeCon

Europe 2026

What's next?

Next Steps, alternatives

- Adaptive
 - More robust benchmarking
 - Additional scorer analysis
 - Interaction with other systems (such as FC-based filtration)
- Predictive Latency
- Flow Control w/Saturation Detection based filtration
- P/D considerations

Takeaway: Promising results, but there is still much to explore, and many ways to solve the problem



KubeCon



CloudNativeCon

Europe 2026

Join us!

Shameless plug time!



llm-d-sched



py-scheduler
(RL-focus)



K8s GIE/IGW



KubeCon



CloudNativeCon

Europe 2026

Questions?