Scaling Sustainable and Cost-Efficient LLMs with llm-d (and vLLM)

Camille Nigon - Solution Architect, Red Hat

May 20th, 2026

CH Open

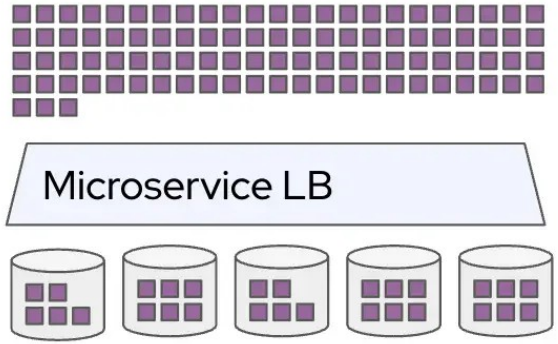
Source | Business | Community

... imagine you need to provide coffee to your employees.

Machine for everyone

Espresso Only

Modern HTTP requests:
Fast, uniform, cheap



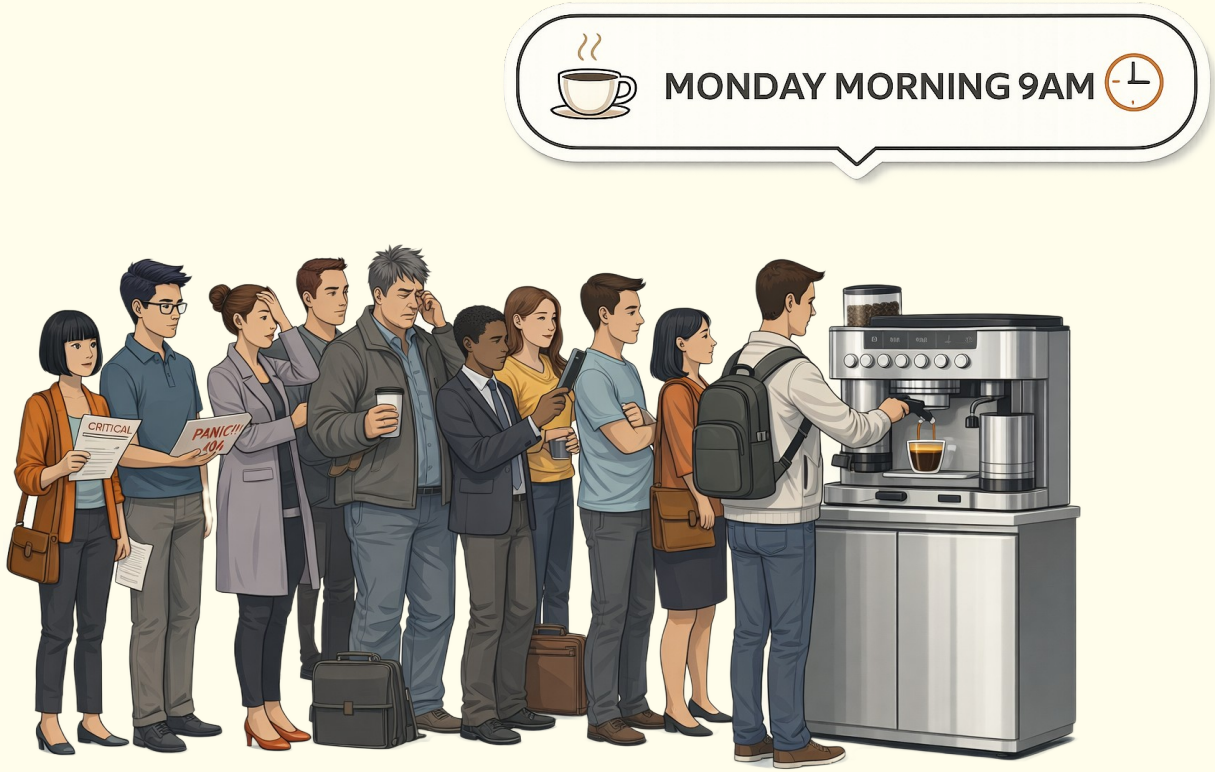
- e.g. loading a webpage, submitting a form
- Requests take approx. the same amount of time
- Service time is predictable
- LB doesn't have to think, just blindly throws traffic at server

LB = Load Balancer


Espresso
Latte
Macchato
Espresso
Latte
Macchato
A
B
Customised
Water-to-coffee ratio
Ratio
Volume
100 ml
WE WANT CUSTOMIZATION
DEMAND CHOICES!
NO STANDARD LIMITS
NO STANDARD RATIOS
NO STANDARD VOLUMES

Multi-Functional

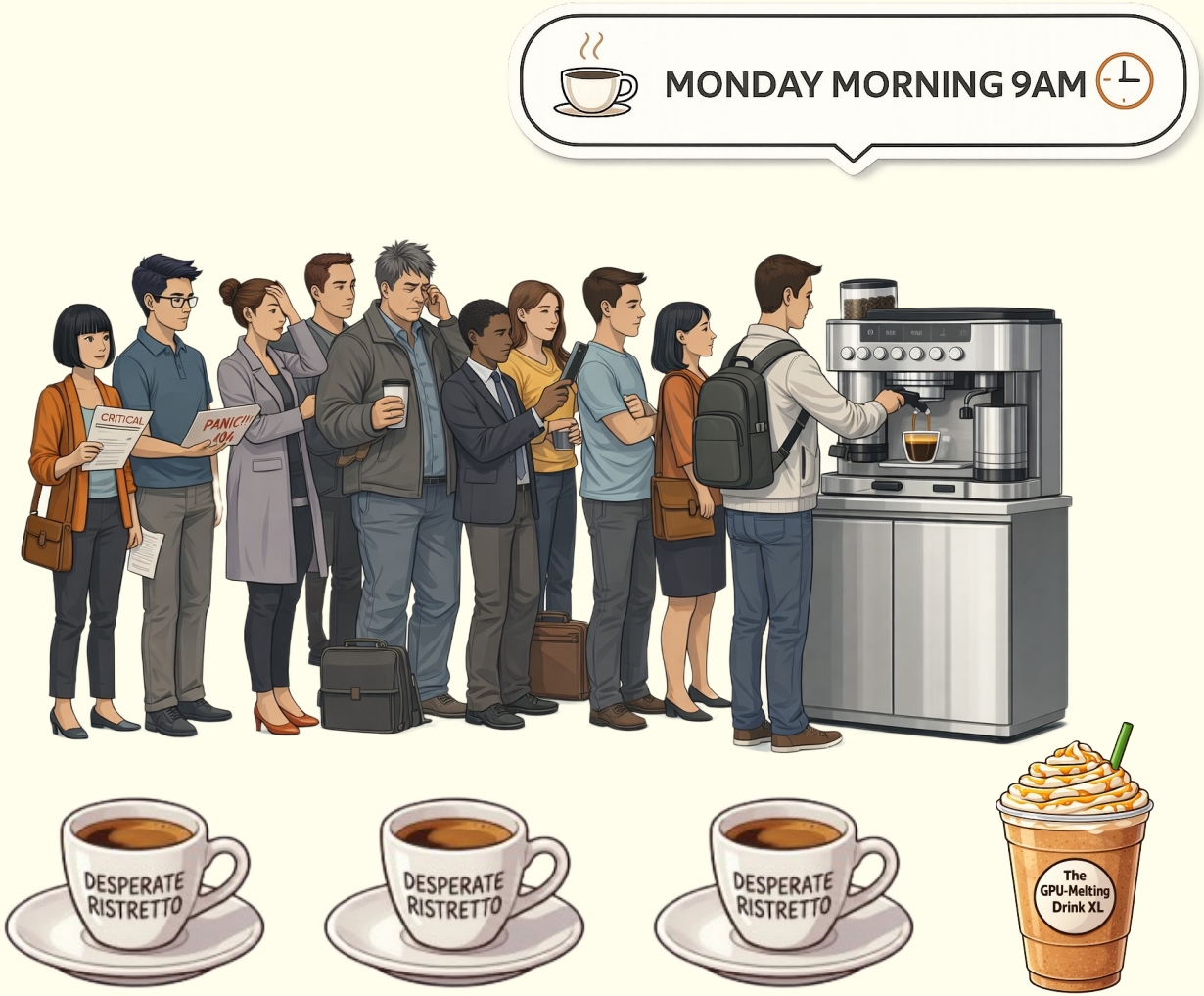

MONDAY MORNING 9AM



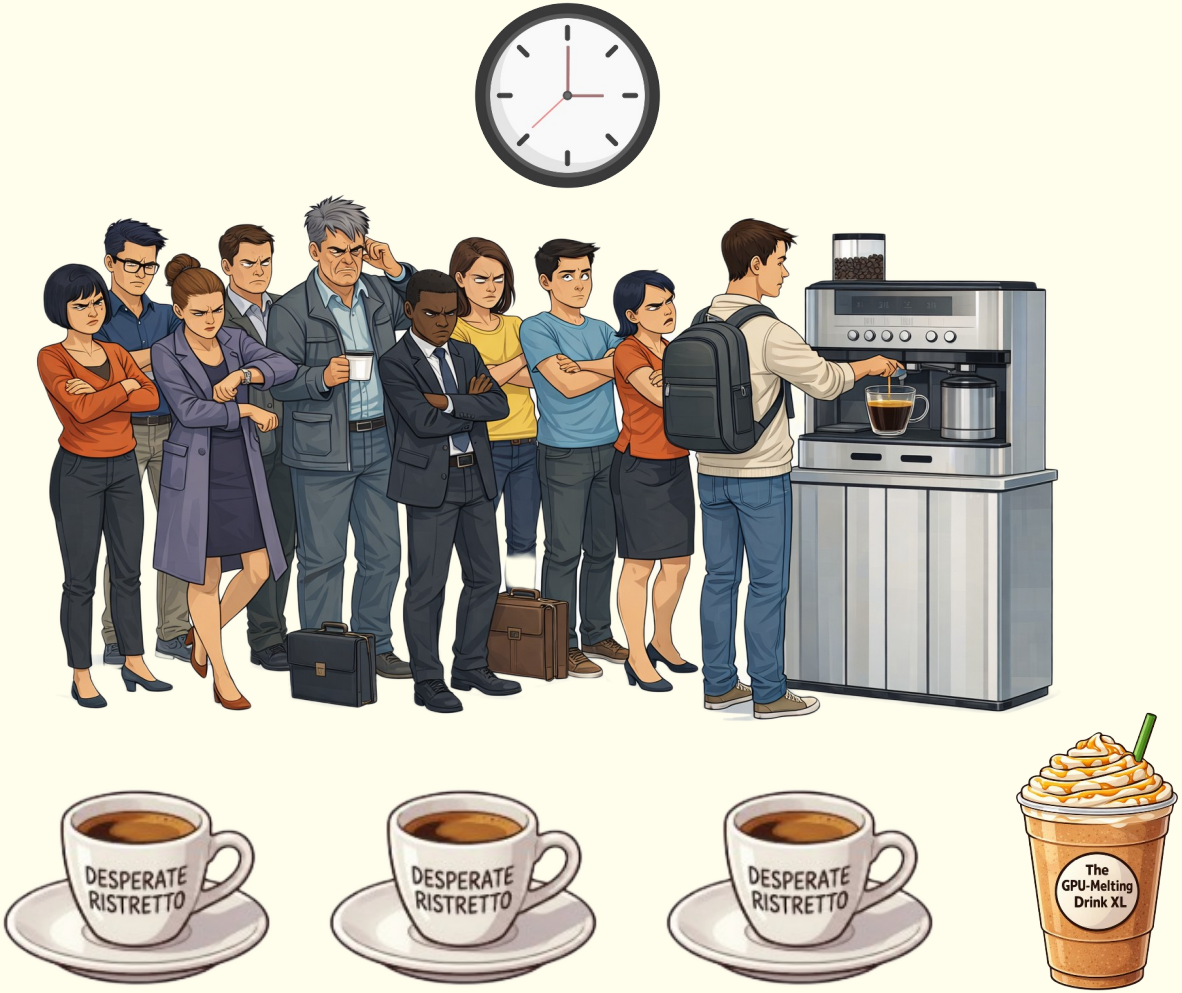
Traffic arrives in bursts.

MONDAY MORNING 9AM

Some requests are ‘small’…



…others are ‘huge’.



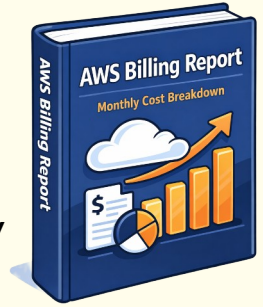
Latency spikes.


DESPERATE
RISTRETTO


The
GPU-Melting
Drink XL

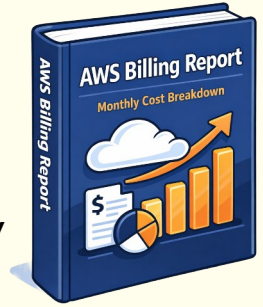
"How do I exit Vim? "

“Attached is our 500-page AWS billing report. Please explain why we are broke.”

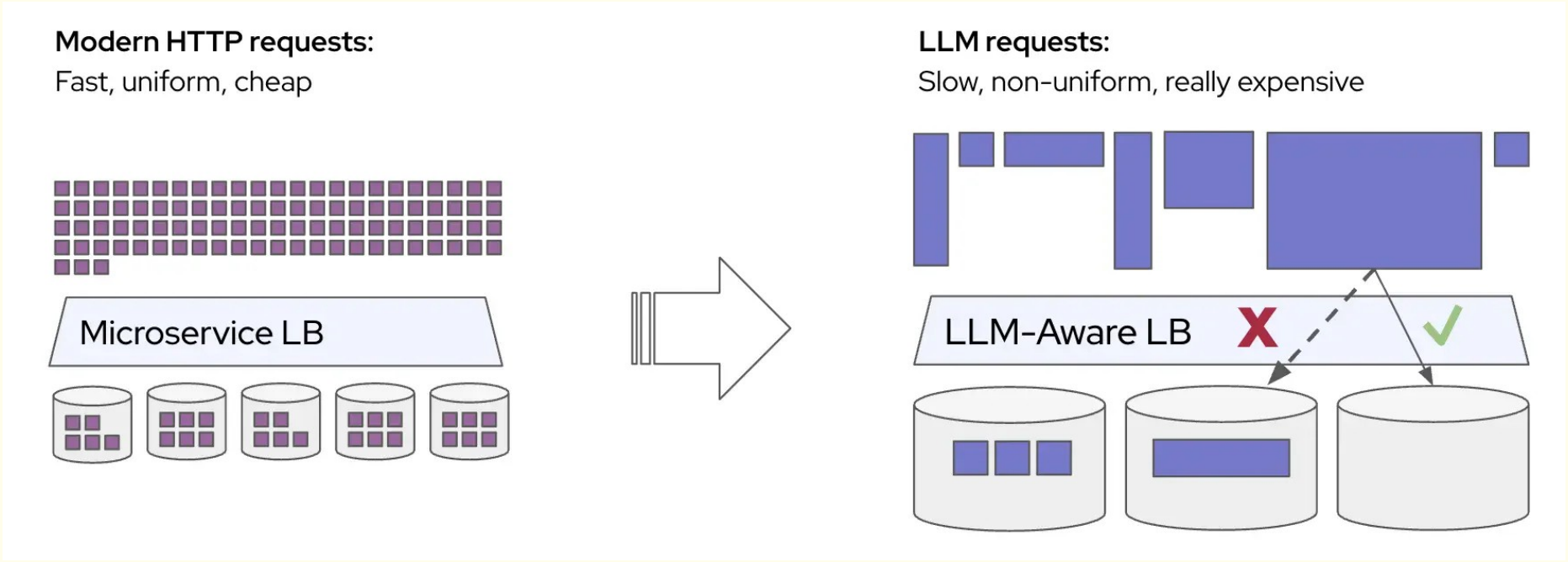


"How do I exit Vim? "

“Attached is our 500-page AWS billing report. Please explain why we are broke.”

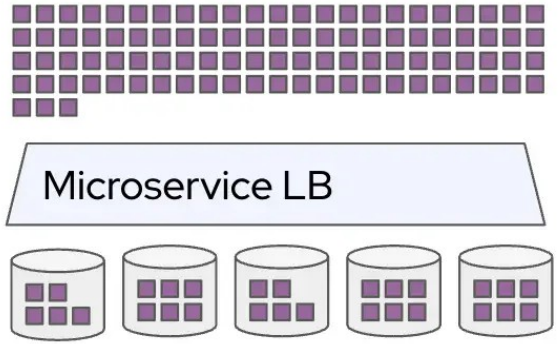


—> ***LLM requests are non-uniform***

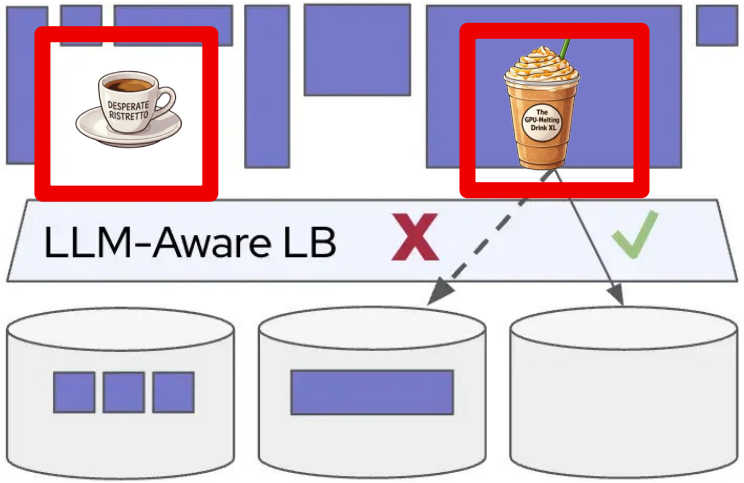


LB = Load Balancer

Modern HTTP requests:
Fast, uniform, cheap



LLM requests:
Slow, non-uniform, really expensive



LB = Load Balancer

...but what is llm-d?

- An open-source framework for distributed LLM inference
- Runs natively on Kubernetes
- Designed to optimize scale, latency, and flexibility for AI workloads
- Joint open source initiative



Disaggregated Serving

Intelligent Scheduling

Some Fundamentals

	Prefill Phase (Compute Bound)	*Decode Phase (Memory Bound)*
Input	The entire user prompt (multiple tokens).	The single, most recently generated token.
Computation	One large, parallel forward pass.	Many small, sequential forward passes.
KV Cache Action	Populates/Builds the initial cache for the whole prompt.	Appends one token's K/V pairs to the cache at each step.

Concept 1 - Disaggregated Serving

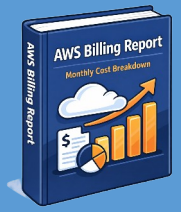
- Disaggregate prefill and decode into specialized pools

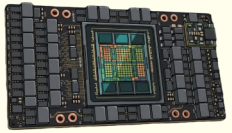
Concept 2 - Intelligent Scheduling

Let's start with an example …

Without Intelligent Scheduling

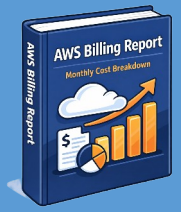
“Did I forget to turn off an EC2 instance?”



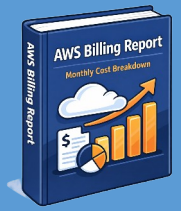


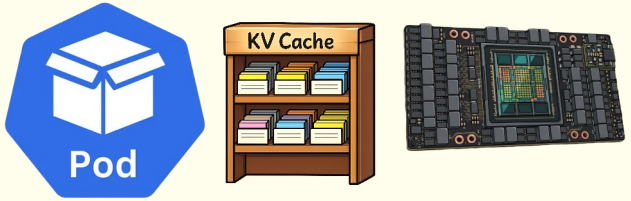
Without Intelligent Scheduling

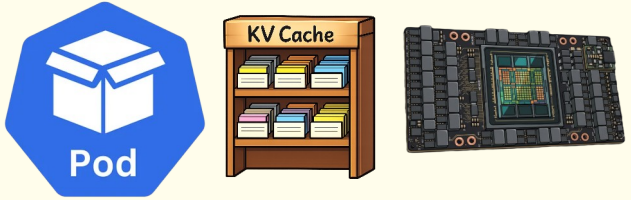
“Did I forget to turn off an EC2 instance?”



“ Which service is personally responsible for my financial downfall this month”?

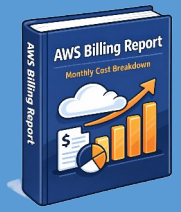




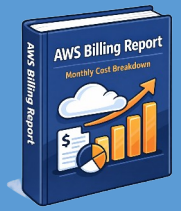


Without Intelligent Scheduling

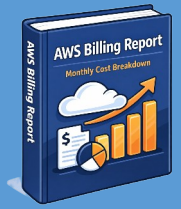
“Did I forget to turn off an EC2 instance?”



“ Which service is personally responsible for my financial downfall this month”?

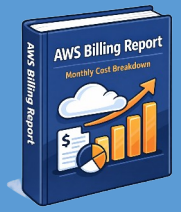


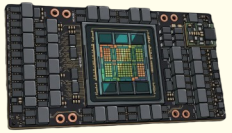
“Which region is secretly running a side hustle on my credit card?”



With Intelligent Scheduling

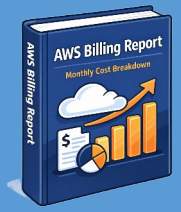
“Did I forget to turn off an EC2 instance?”



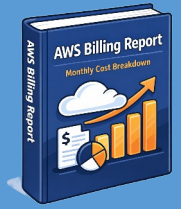


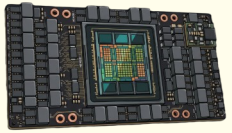
With Intelligent Scheduling

“Did I forget to turn off an EC2 instance?”



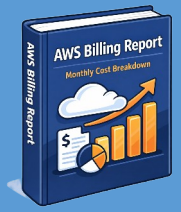
“ Which service is personally responsible for my financial downfall this month”?



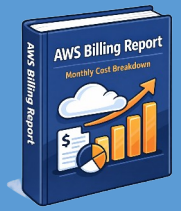


With Intelligent Scheduling

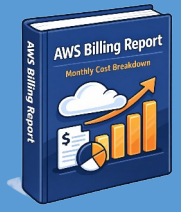
“Did I forget to turn off an EC2 instance?”

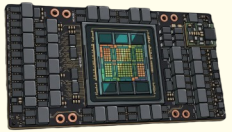


“ Which service is personally responsible for my financial downfall this month”?



“Which region is secretly running a side hustle on my credit card?”





Conclusion

- LLM requests are not uniform
- llm-d solves that problem using:
  - disaggregated serving
  - intelligent scheduling system
- Result
  - Lower latency
  - More efficient GPU utilization (lower cost)
  - Standardized serving across teams

Thank you , let's grab a coffee :)