llm-d

Distributed Inference with llm-d and Kubernetes

Antonio Cardace
Principal Machine Learning Engineer
Inference Engineering
Red Hat

Agenda

Overview of the project

- What is llm-d
- Why llm-d
- What challenges does llm-d solve
- Phases of Inference
- Architecture overview
- Well-lit paths

Demo

- Intelligent Inference Scheduling

What is llm-d?

any model, any accelerator, any cloud

llm-d

An open-source framework for distributed large language model (LLM) inference that **runs natively on Kubernetes.**

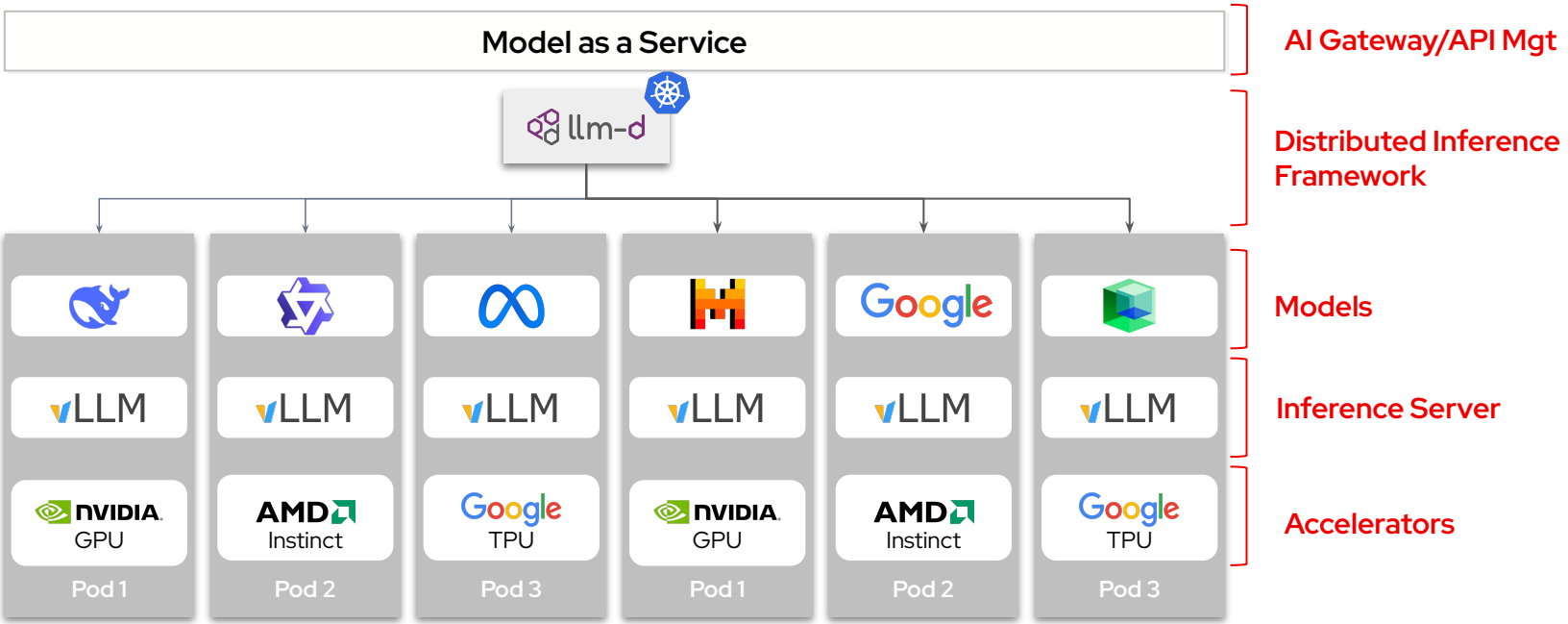
- Joint **open source** initiative by Red Hat, Google, IBM Research, NVIDIA and many more organizations
- **Deployable anywhere** - k8s + increasing hardware support
- **Distributed and pluggable architecture**

NVIDIA · Hugging Face · CoreWeave · IBM · Google

CLOUD NATIVE COMPUTING FOUNDATION

Why llm-d?

From token consumers to token providers



Why llm-d?

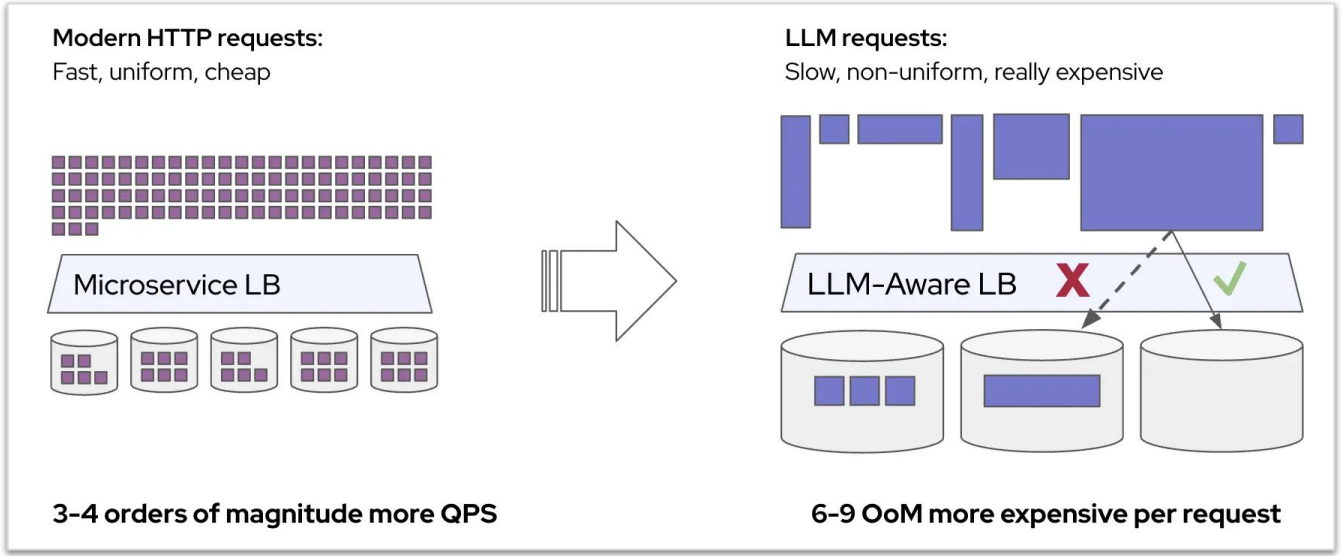
Why can’t we solve these challenges with the existing kubernetes primitives?

Core challenges:

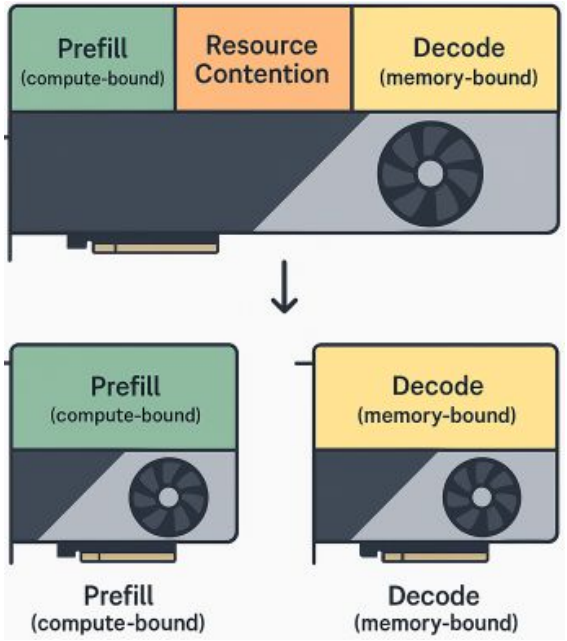
- Scalability
- Reliability
- Speed / Efficiency

Web Apps vs AI apps

We can exploit the unique properties of LLM inference to improve perf/$ over naive load balancing



Red Hat



Inference is split into two phases:

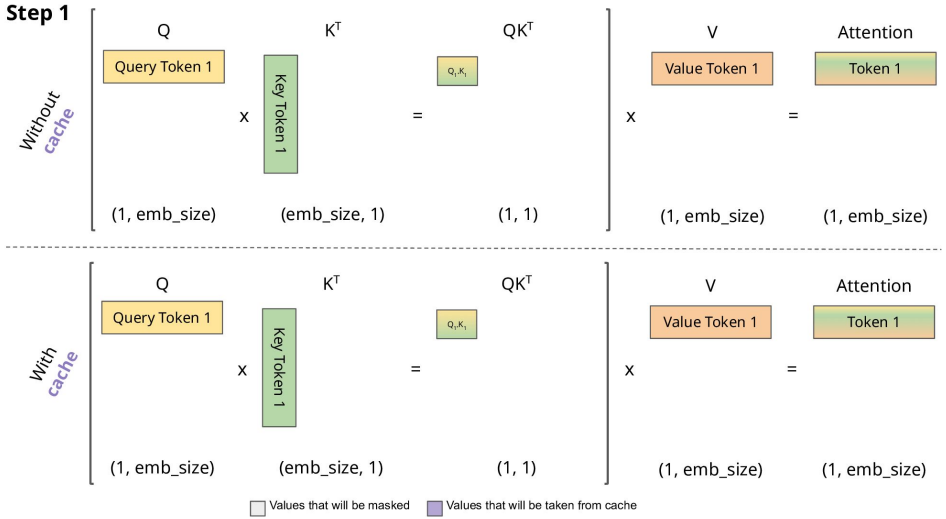
prefill and **decode**.

- **Prefill** generates the first output token and runs in parallel over all the prompt tokens **(compute bound)**
- **Decode** generates tokens one at a time by doing a full pass over the model (**memory bandwidth bound**)

Red Hat

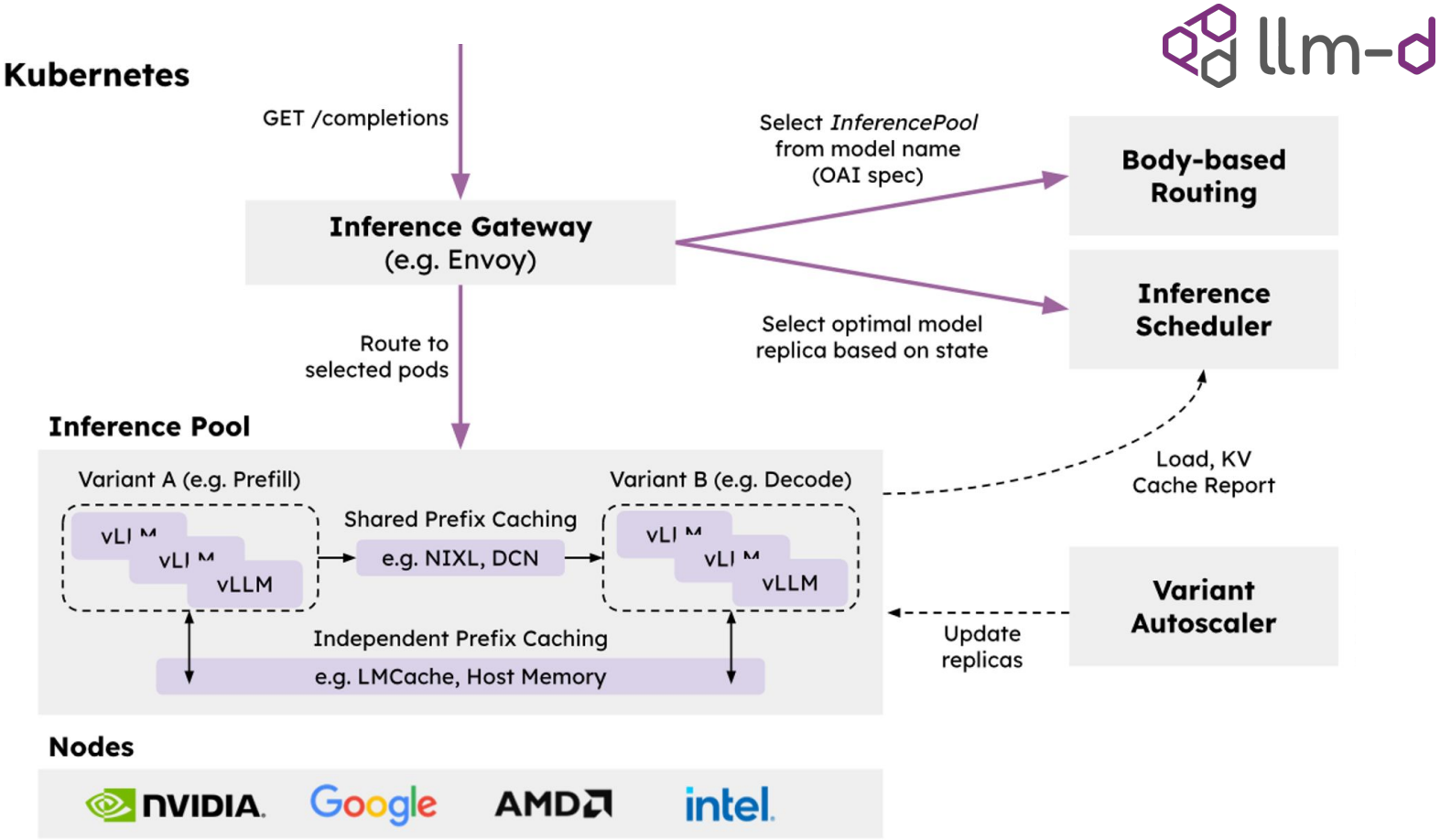
KV Caching

KV Cache: Caching Key and Value vectors in self-attention saves redundant computation and accelerates decoding - *but takes up memory*!



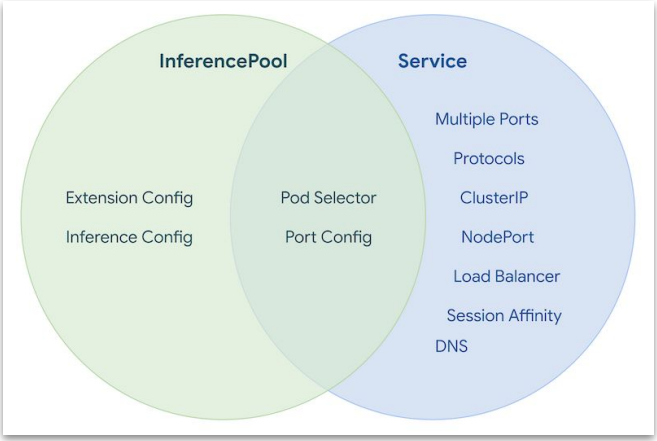
Red Hat

LLM-D Architecture Overview

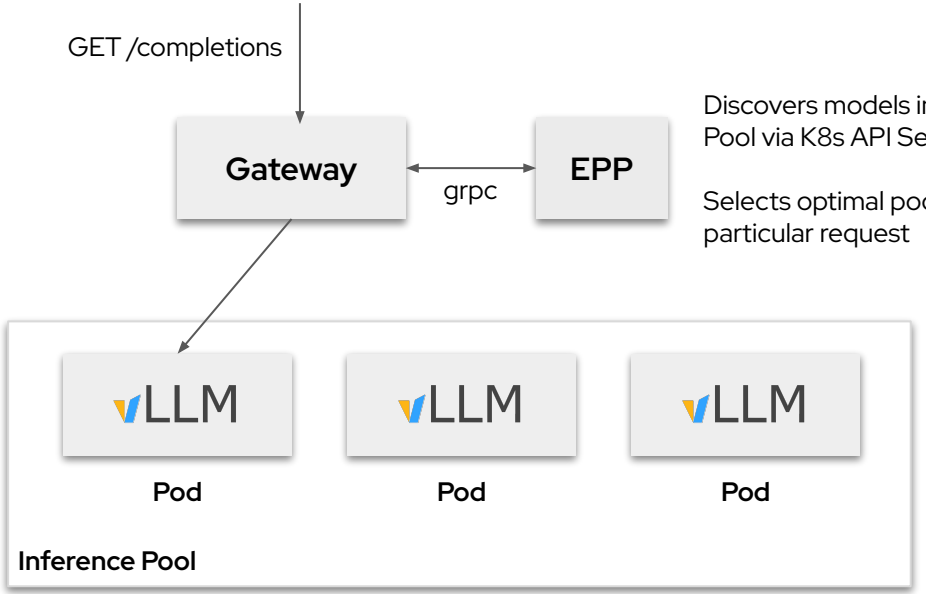


Architecture

Inference Pool API



Abstraction for managing LLM servers, allowing focus on pool management rather than networking



Red Hat

“Well-lit” Paths

llm-d provides “well-lit” paths for running LLM inference workloads

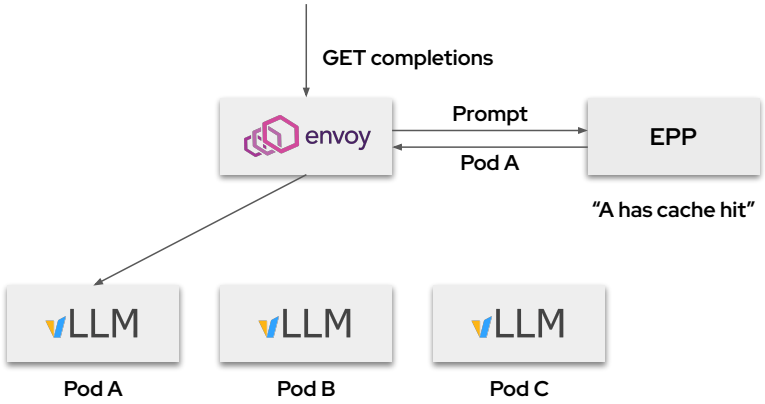
- **Intelligent Inference Scheduling**
- **Tiered KV Caching**
- **P/D Disaggregation**

Red Hat

Intelligent Inference Scheduling

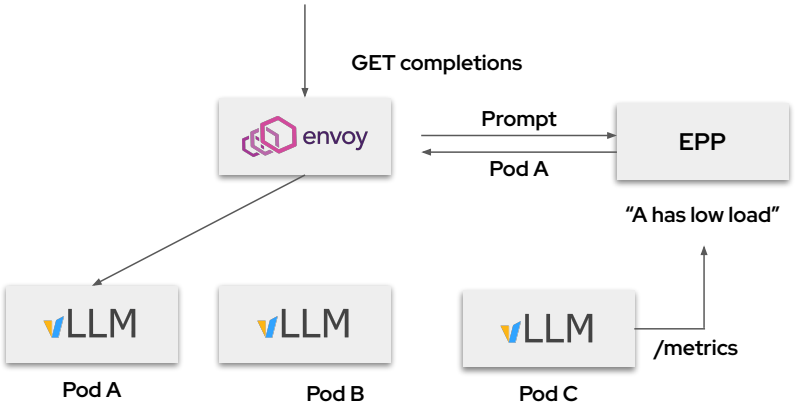
vLLM-aware load-balancing enables smarter request routing that improve SLOs

Prefix-Aware Routing



Dramatically increase prefix-cache hit rate

Load-Aware Routing

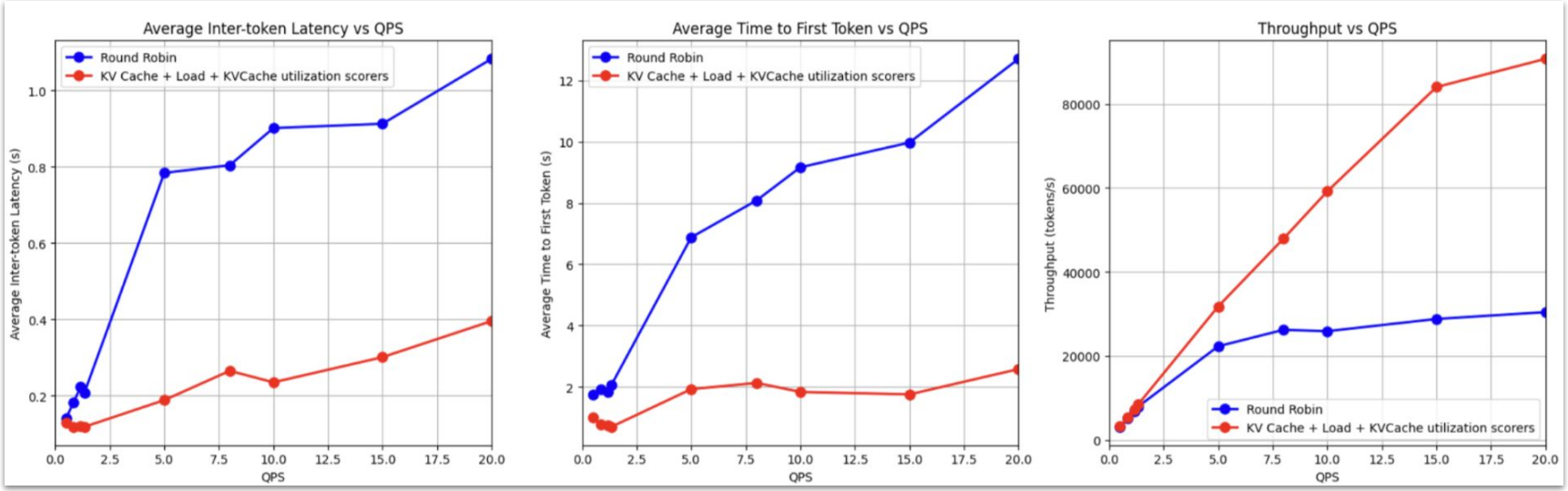


Load-balancing based on actual replica state

Red Hat

Intelligent Inference Scheduling

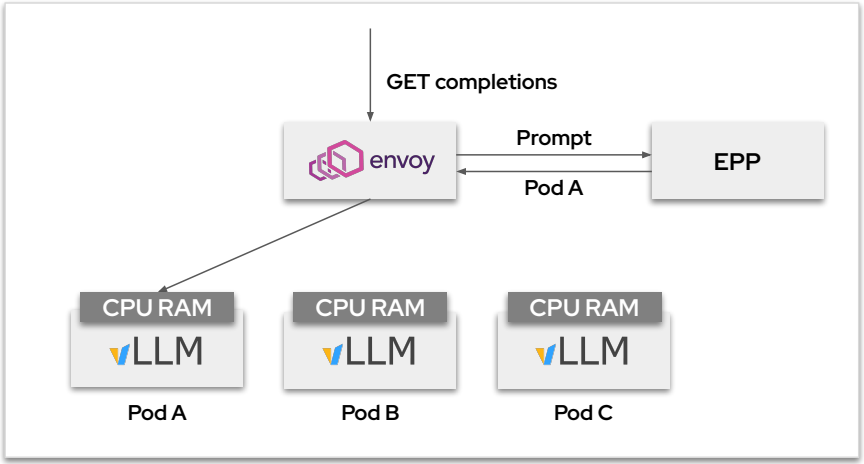
Inference scheduling is a no-brainer optimization which can have huge impacts on repeated prompts



Tiered KV Caching

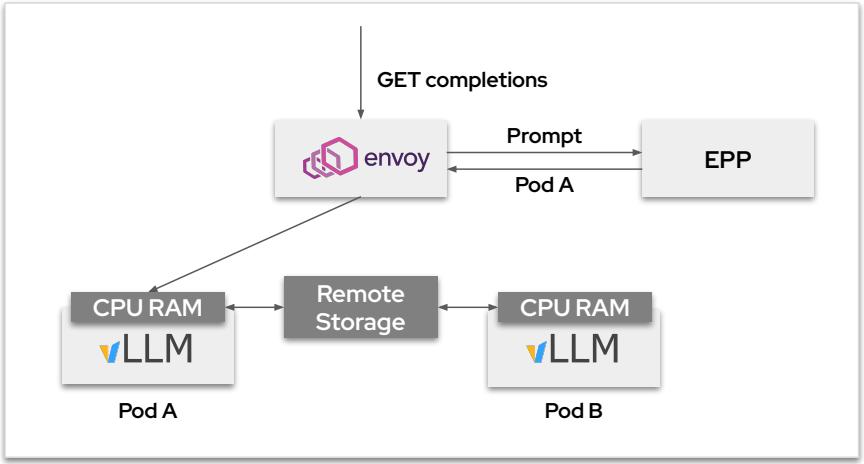
Leverage all system resources to maximize prefix cache hit rate within the cluster

CPU KV Cache Offloading



Offload KV caches to local memory with global index

Storage KV Cache Offloading

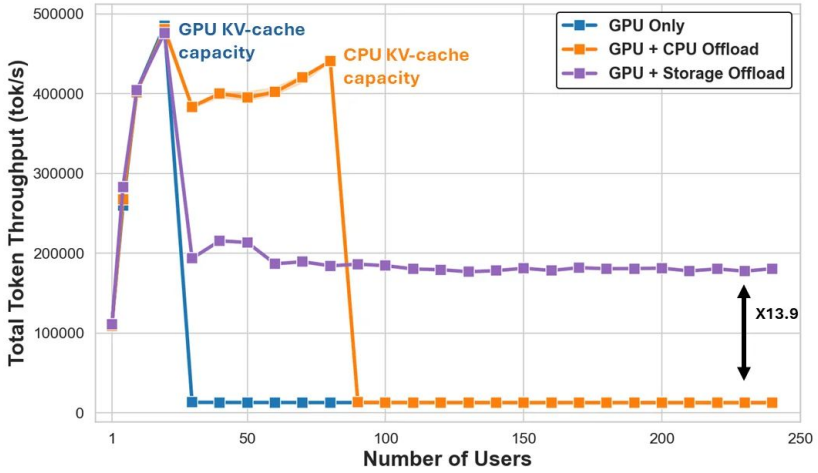


Offload KV caches to global shared remote storage

Red Hat

Tiered KV Caching

Leverage all system resources to maximize prefix cache hit rate within the cluster

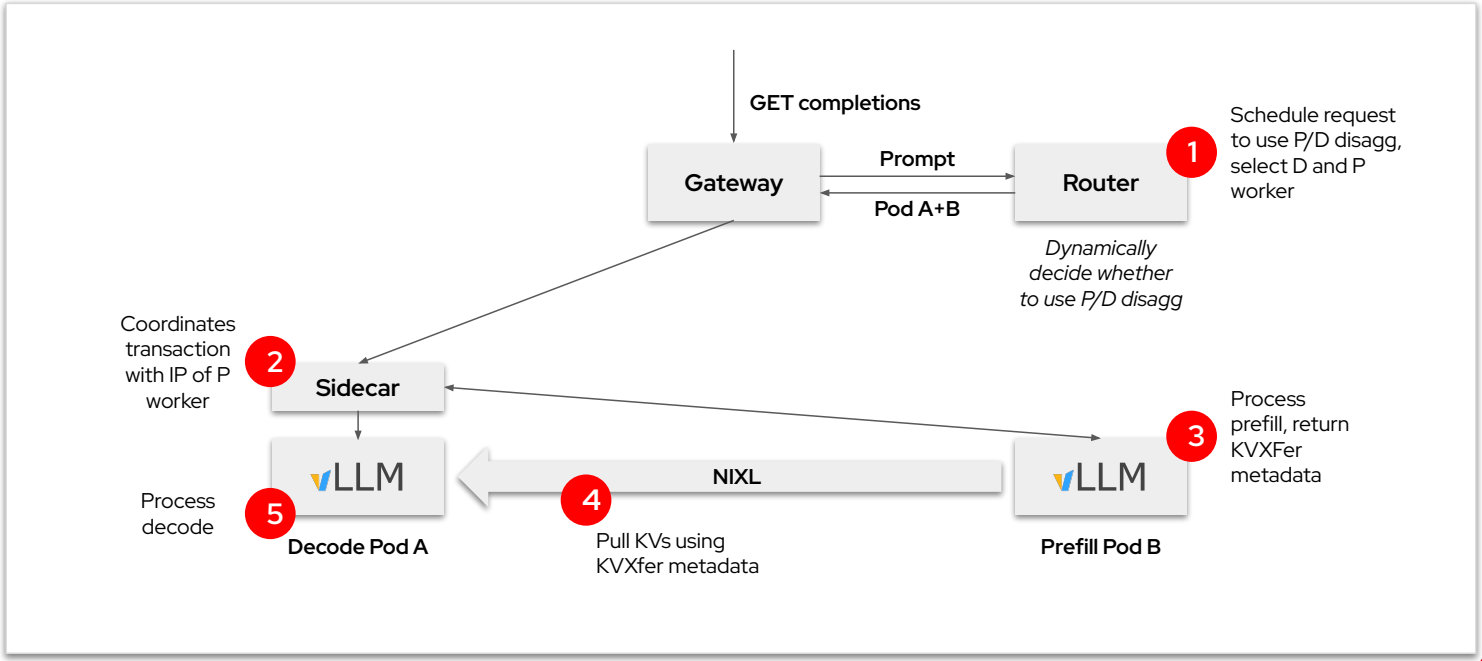


Dramatically expands the "working set size" (i.e. number of requests we can keep the caches around for)

Red Hat

P/D Disaggregation

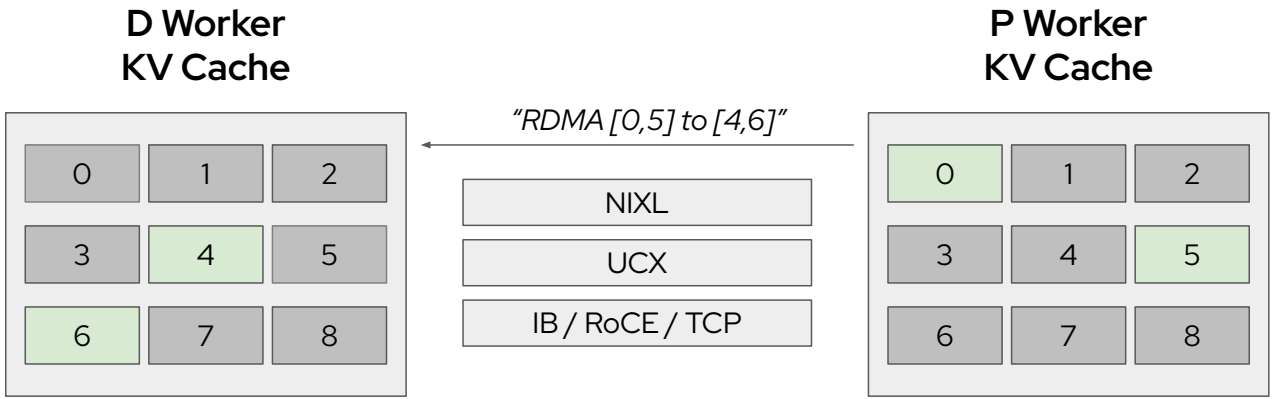
llm-d composes vLLM and Gateway to enable P/D disaggregation



Red Hat

Efficient KV Transfer in vLLM vis NIXL

For high performance KV Transfer, vLLM uses GPU direct RDMA via our NIXL integration

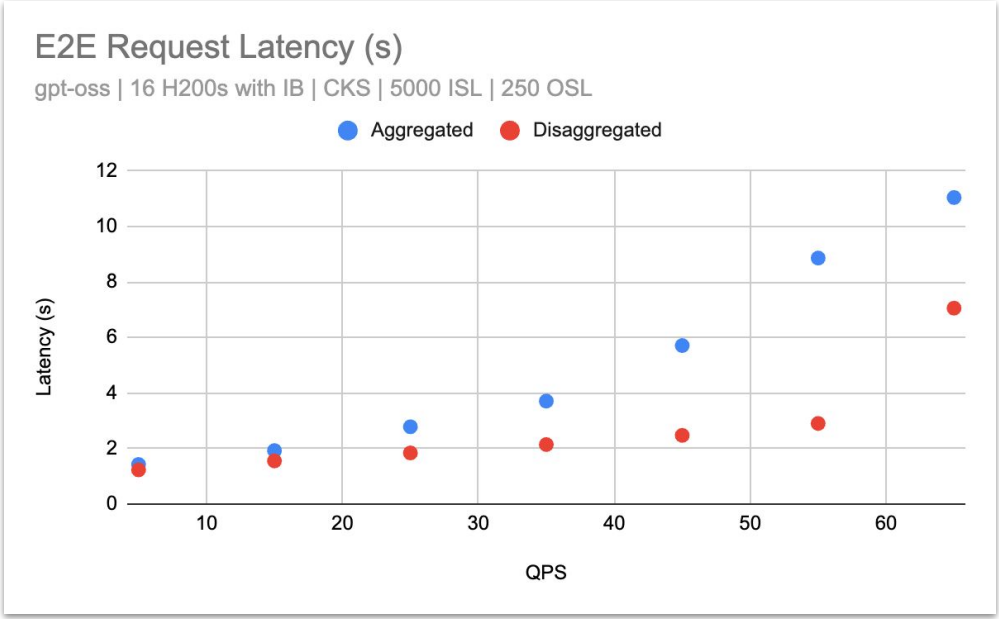


Zero Copy, GPU Direct, Zero Memory Overhead

Red Hat

P/D Improves Latency

For openai/gpt-oss-120b model, we can see a ~50% latency improvement for prefill-heavy workloads



Note: disagg setup uses 2DTP4 | 8PTP1

Red Hat

Demo!

https://github.com/acardace/llm-d-demo

Red Hat

Where to next?

- Latency Prediction Plugin
- Batch Gateway
- Flow Control

Red Hat

Thank you

https://llm-d.ai

https://llm-d.ai/slack

https://github.com/llm-d/llm-d

Red Hat