Red Hat Summit

Connect

Distributed Inference with Red Hat AI

Scaling LLMs from experimentation to a production-grade service

Erkan Ercan
Principal Solution Architect,
Red Hat
https://www.linkedin.com/in/erkanercan/

Agenda

- Introduction
- What Is LLM Inference?
- vLLM – as the Defacto Runtime For GenAI
- What Problem is vLLM Solving?
- Challenges in Scaling Inference Workloads
- Distributed Inference At Scale
- Questions & Answers

What are Large Language Models (LLMs)?

Neural Networks

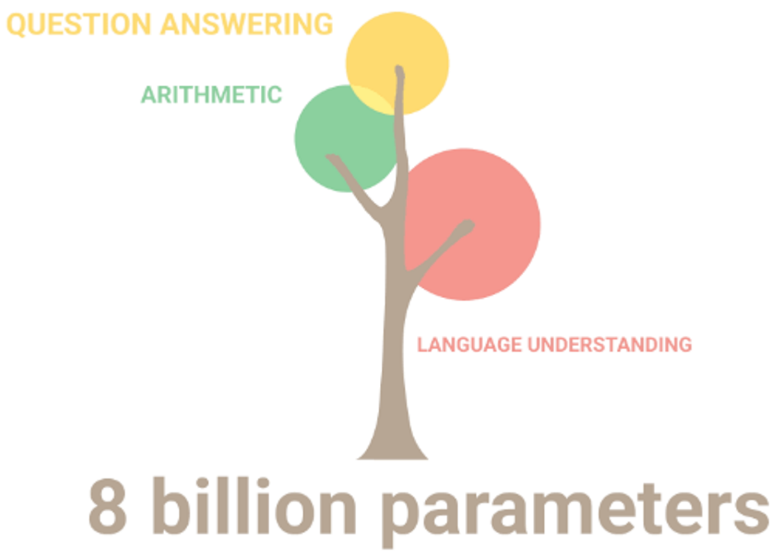
- Recognize, Predict, and Generate **text**
- **Trained** on a **VERY** large corpuses of text
- Deduce the **statistical** relationships between tokens
- Can be **fine-tuned**

ChatGPT
Llama
Qwen
DeepSeek
Gemini
Mistral
Molmo
Phi
Kimi-K2
Nemotron
Granite
GLM

An LLM **predicts the next token**

based on its training data and statistical deduction

More parameters means more capabilities

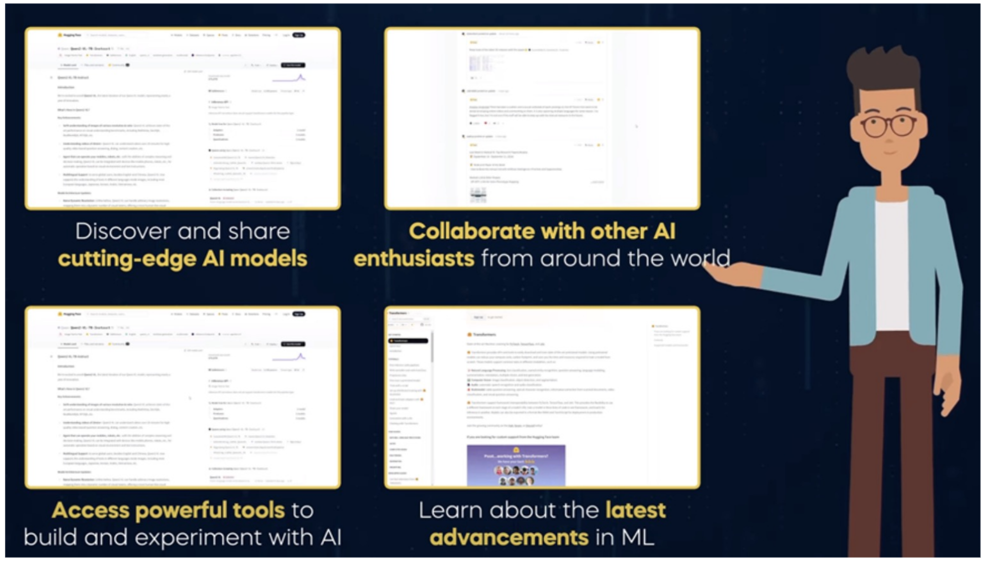


https://research.google/blog/pathways-language-model-palm-scaling-to-540-billion-parameters-for-breakthrough-performance/

Advantages of open weight models and serving stack

Open models play an important role in the enterprise AI landscape

- **Cost**
  - Self managed infrastructure
  - 1B – 1000B size – match task difficulty to model
- **Customization**
  - Improve accuracy and costs with task specific tuning
- **Control**
  - Model lifecycle (no changes to the model in place)
  - Resources (no rate limits / API downtime)
- **Security**
  - Complete data privacy (no 3rd party APIs)



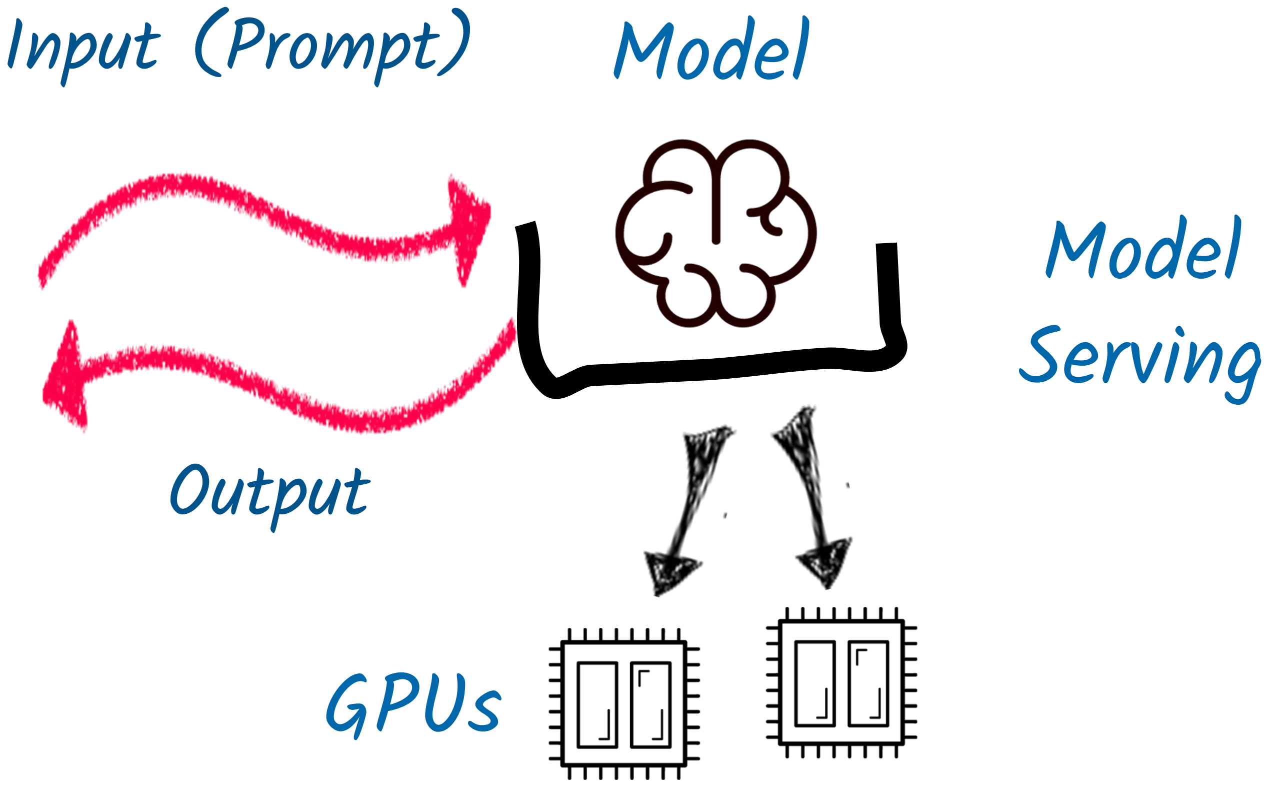
Inference (Model Serving)

Model Serving

- Run the model
- CPU / GPU
- Expose an API

Input

- Prompt (text)
- Instructions to give to the model
- Taming a model is hard



LLM Inference: a birds-eye view

Input prompt:

"Translate from English to French: It's nice to meet you."

Translate from

English to French:

...

Output text:

"Enchanté de faire votre connaissance"

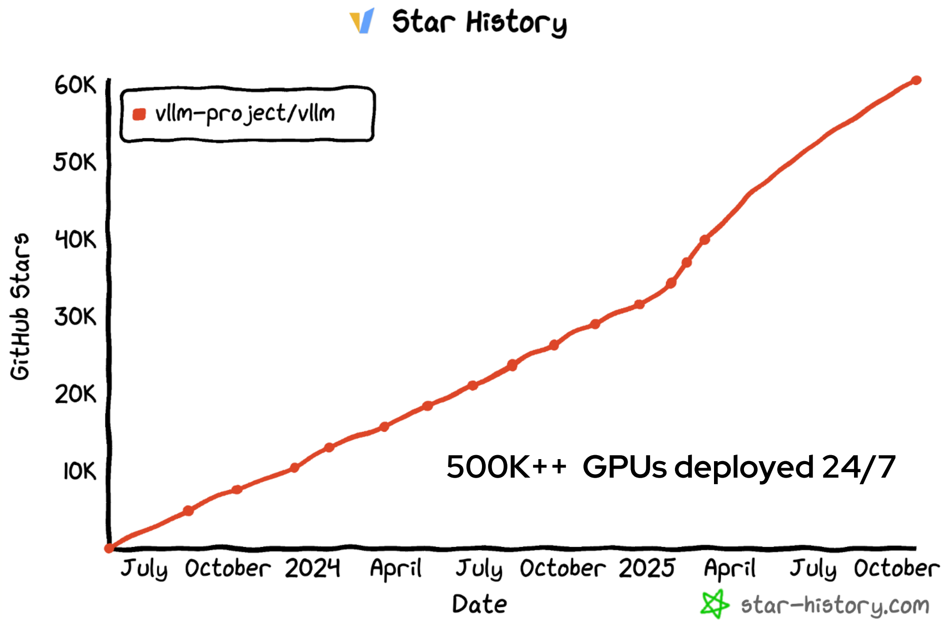
vLLM: The De Facto Open GenAI Inference Platform

vLLM Inference Server in Red Hat AI

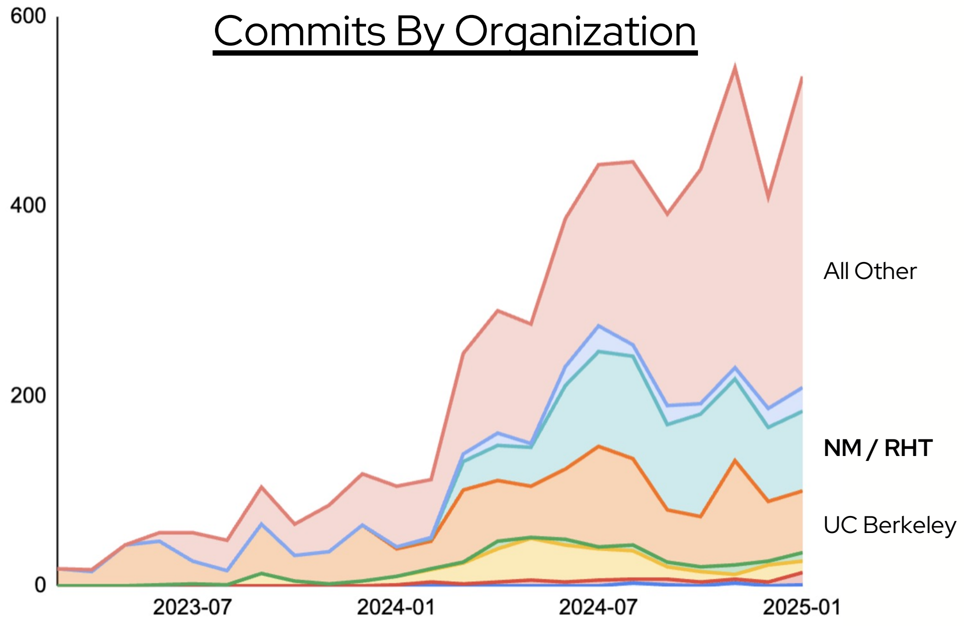
Neural Magic Boosts Our Community Leadership & Enterprise Support

vLLM: #1 OSS Inference Server

>60K Github Stars



Commits By Red Hat & Neural Magic



vLLM: The De Facto Open GenAI Inference Platform

vLLM has emerged as the Linux of GenAI Inference

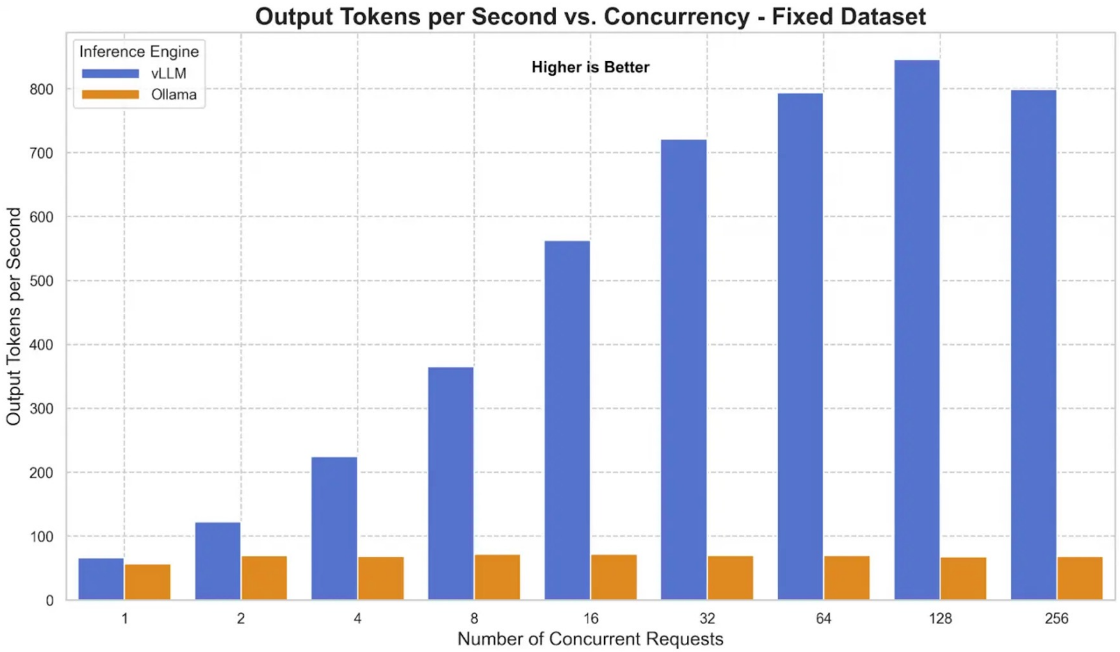
Llama | Qwen | GPT-OSS | DeepSeek | Gemma | Mistral | Molmo | Phi | Nemotron | Granite

vLLM

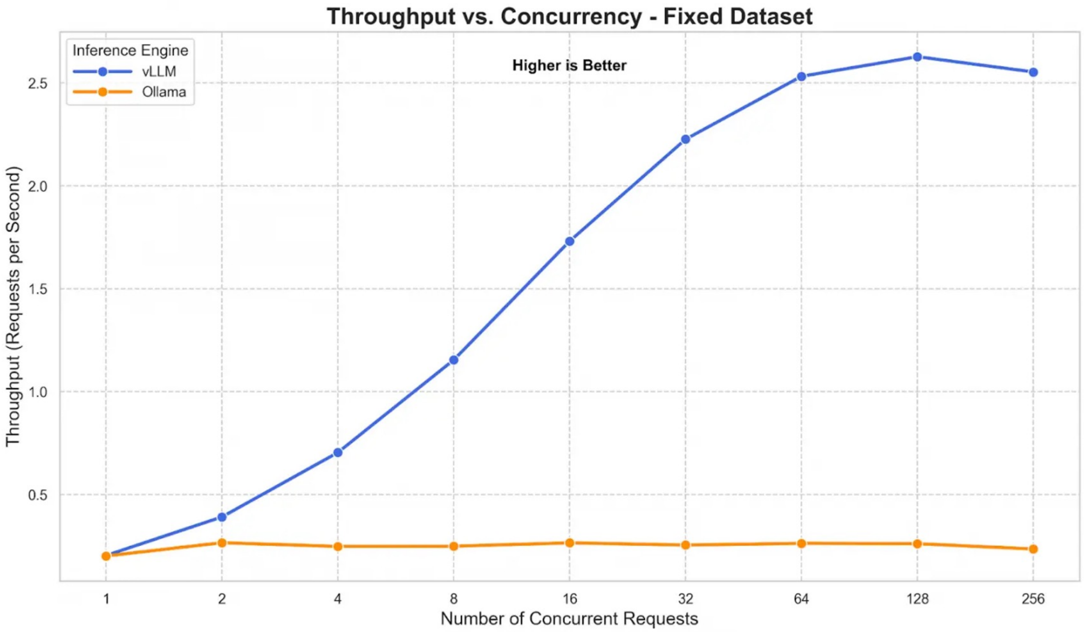
NVIDIA GPU | AMD Instinct | Google TPU | aws Neuron | intel Gaudi | IBM Spyre

Physical | Virtual | Private Cloud | Public Cloud | Edge

vLLM vs Ollama Performance Benchmark



Model: Llama3.1-8B | GPU: Nvidia A100 | Dataset: Fixed Dataset



Model: Llama3.1-8B | GPU: Nvidia A100 | Dataset: Fixed Dataset

Red Hat AI repository on Hugging Face

Collection of third-party models

- Llama
- Qwen
- Gemma
- Mistral, Voxtral
- DeepSeek
- Phi
- Molmo
- Granite
- Nemotron
- OpenAI GPT-oss
- K2
- SMOLLM3 3B

Choice of Models

- Transformers (Dense, MOE), Multi-modal LLMs, Embeddings Models, Hybrid / Novel Attention, Vision
- Hugging Face compatible (safe tensors), OCI-compatible containers

Validated models

- Tested using realistic scenarios
- Assessed for performance across a range of hardware
- Done using GuideLLM benchmarking and LM Eval Harness

Optimized models

- Compressed for speed and efficiency
- Designed to run faster, use fewer resources, maintain accuracy
- Done using LLM Compressor with latest algorithms

What Problem is vLLM Solving?

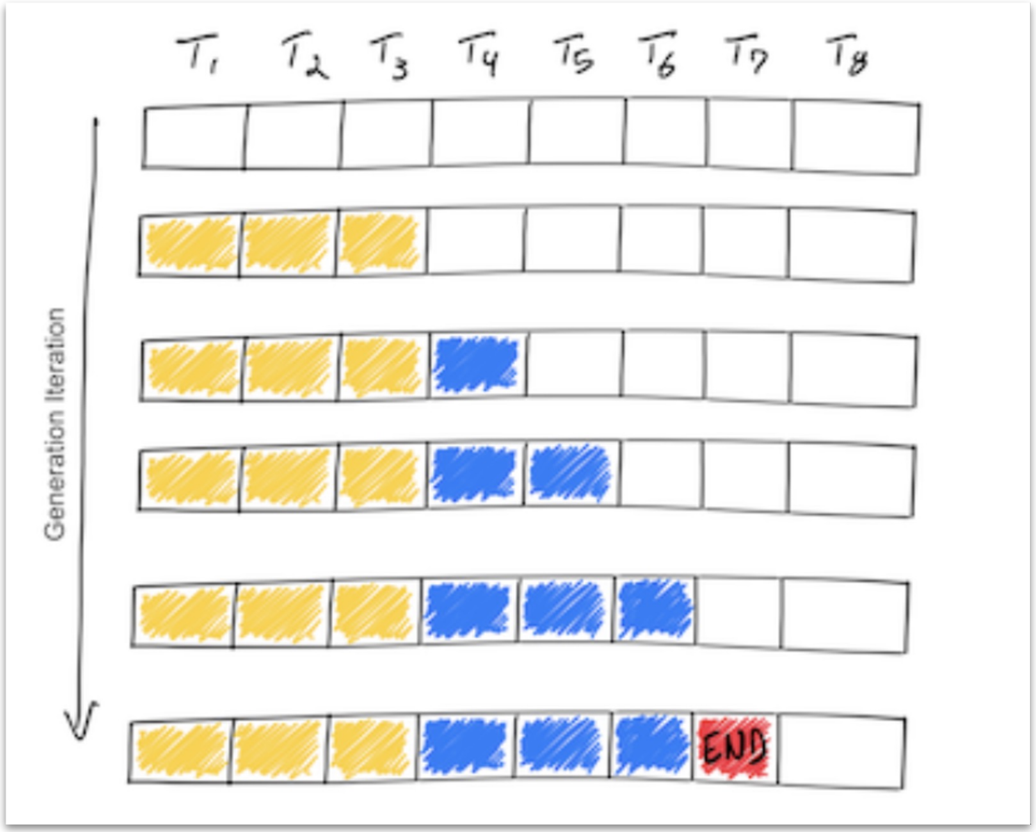
What Problem is vLLM Solving?

Production Inference Serving

- **Batch Size > 1 & Data Center Hardwares**
  - Not the same workload as on-device inference for a single user
- **How do you?**
  - Efficiently schedule requests into the next forward pass?
  - Manage KV cache context and runtime memory footprint?

Why Is This A Hard Problem?

- **A LLM is a function to predict the next token in a sequence**
  - P(X_n | X_0 … X_n-1)
- **To generate text, we "chain together" passes through the model**
  - → A single request requires multiple passes through the model
  - → A single generation request can last multiple seconds
- **Key Challenge: How to handle multiple concurrent requests**



Challenge 1: Batching

Naive/Static Batching 🙅‍♂️🙅‍♂️🙅‍♂️

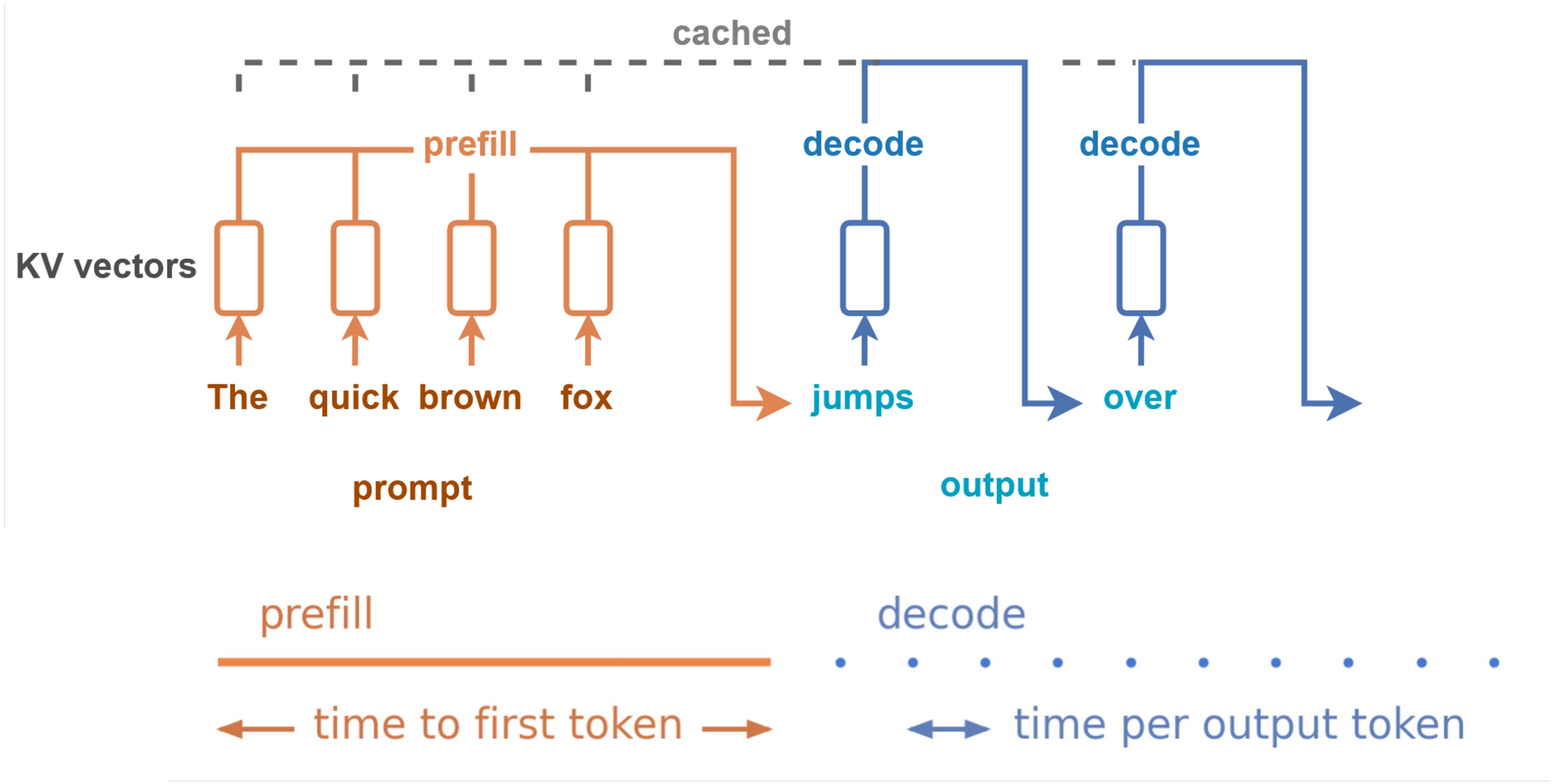
T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16
S_1	S_1	S_1	S_1				S6	S6	S6	S6		S11	S11	S11	S11
S2	S2	S2					S7	S7	S7			S12	S12		
S3	S3	S3	S3	S3	S3	S3	S8	S8				S13	S13	S13	
S4	S4	S4	S4				S9	S9	S9	S9					
S5	S5						S10	S10	S10	S10	S10				

Continuous Batching 🙏🙏🙏

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
S_1	S_1	S_1	S_1	S8	S8	S10	S10	S10	S10	S10	
S2	S2	S2	S7	S7	S7		S11	S11	S11	S11	
S3	S3	S3	S3	S3	S3	S3	S12	S12			
S4	S4	S4	S4	S9	S9	S9	S9				
S5	S5	S6	S6	S6	S6			S13	S13	S13	

• • •

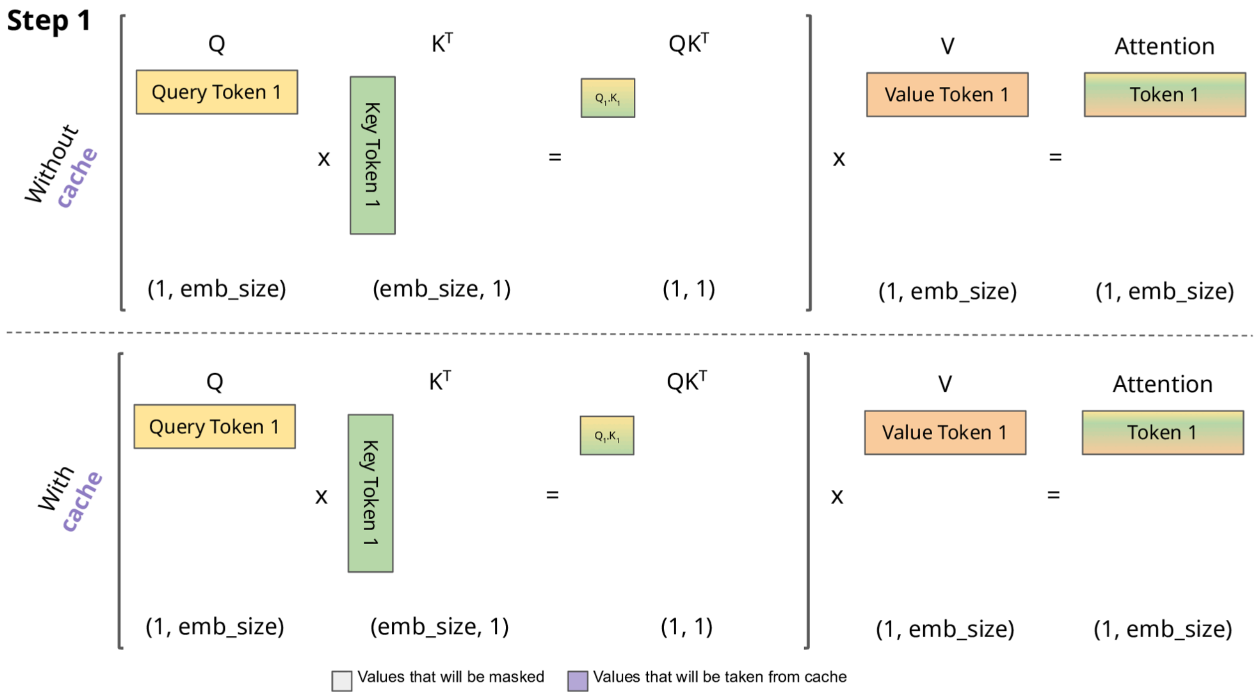
Challenge 2: KV Caching

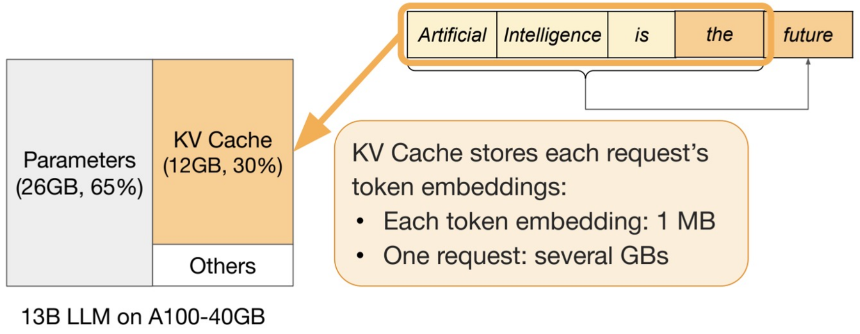


[Source: Blog by Benjamin Merkel]

Challenge 2: KV Caching

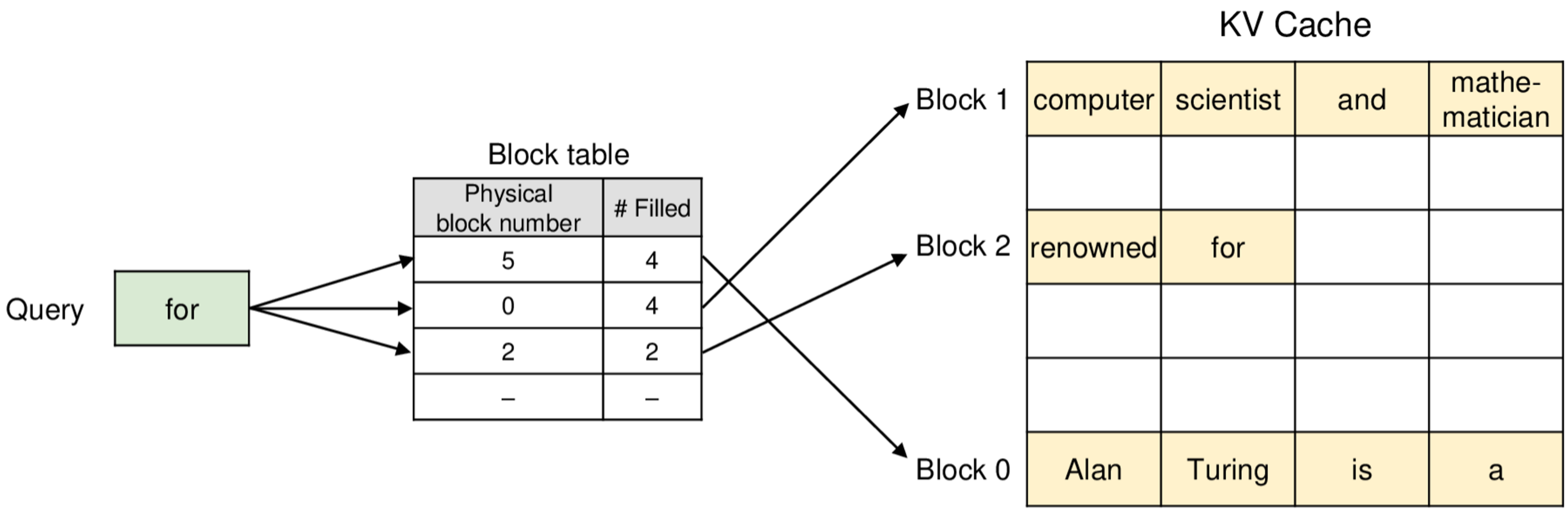
KV Cache: Caching Key and Value vectors in self-attention saves redundant computation and accelerates decoding - *but takes up memory!*





vLLM's Original Innovation: Paged Attention

An attention algorithm that allows for storing continuous keys and values in non-contiguous memory space.



Automatic Prefix Caching

Re-use KV cache blocks across requests! Improves time-to-first-token by skipping prefill

Example: Multi-turn conversation

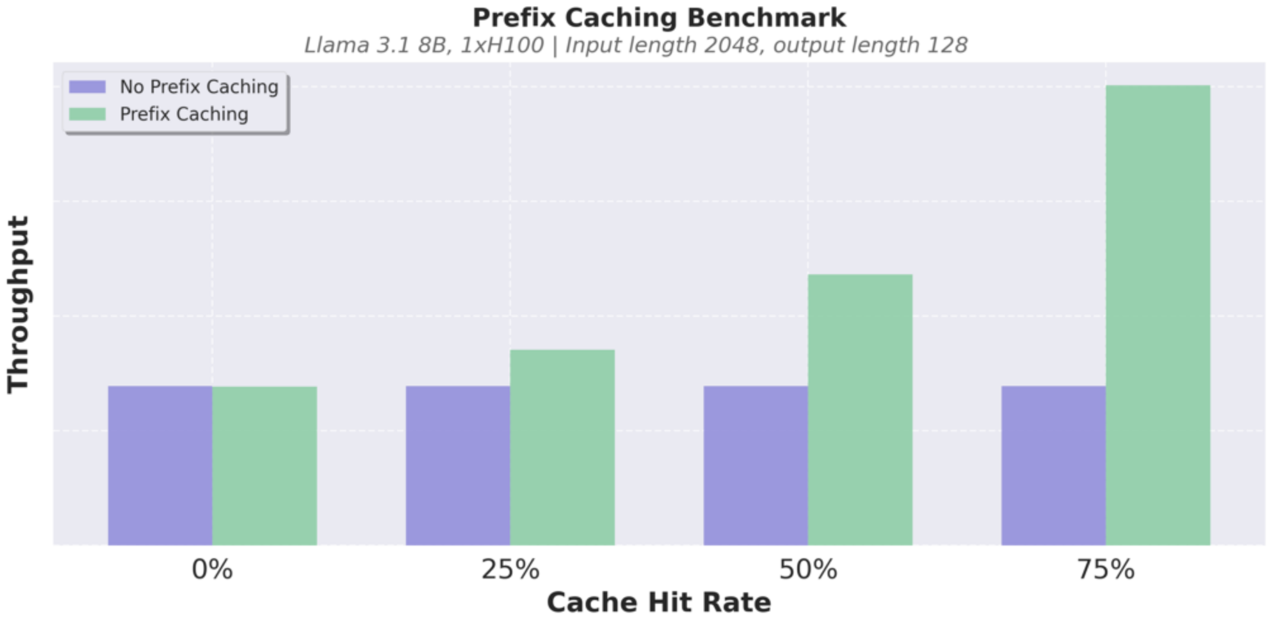
Prompt (round 1)
Human: *What's AI?*

Cached

LLM Result (round 1)
LLM: *AI is technology that simulates human intelligence, like Siri or Google Maps.*

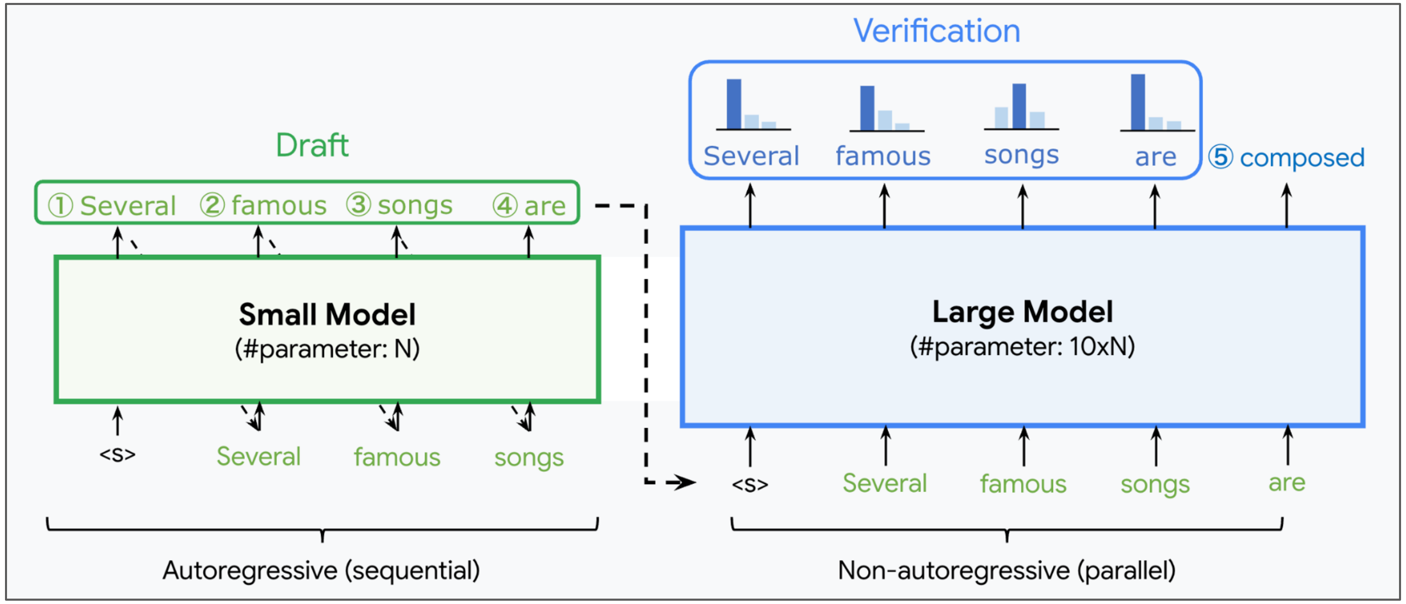
Prompt (round 2)
Human: *What's AI?*
LLM: *AI is technology that simulates human intelligence, like Siri or Google Maps.*
Human: *Cool, thanks!*

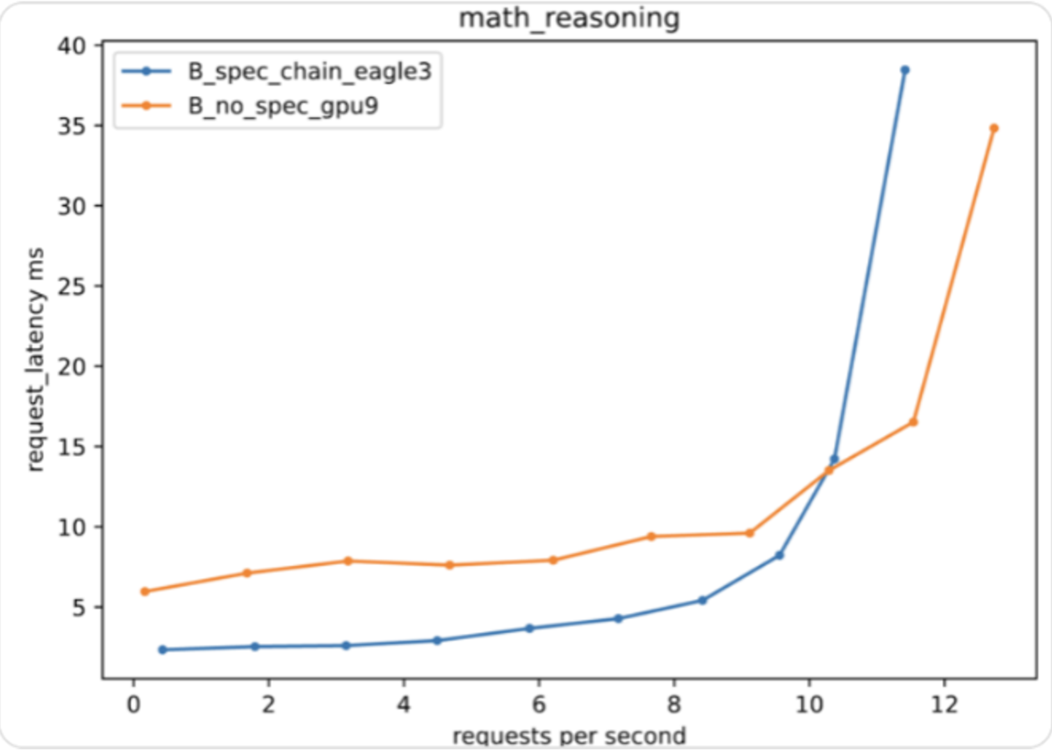
LLM Result (round 2)
LLM: *No problem!*



Speculative Decoding

Accelerate decoding phase with speculation - variety of methods: ngram, draft model, EAGLE, etc





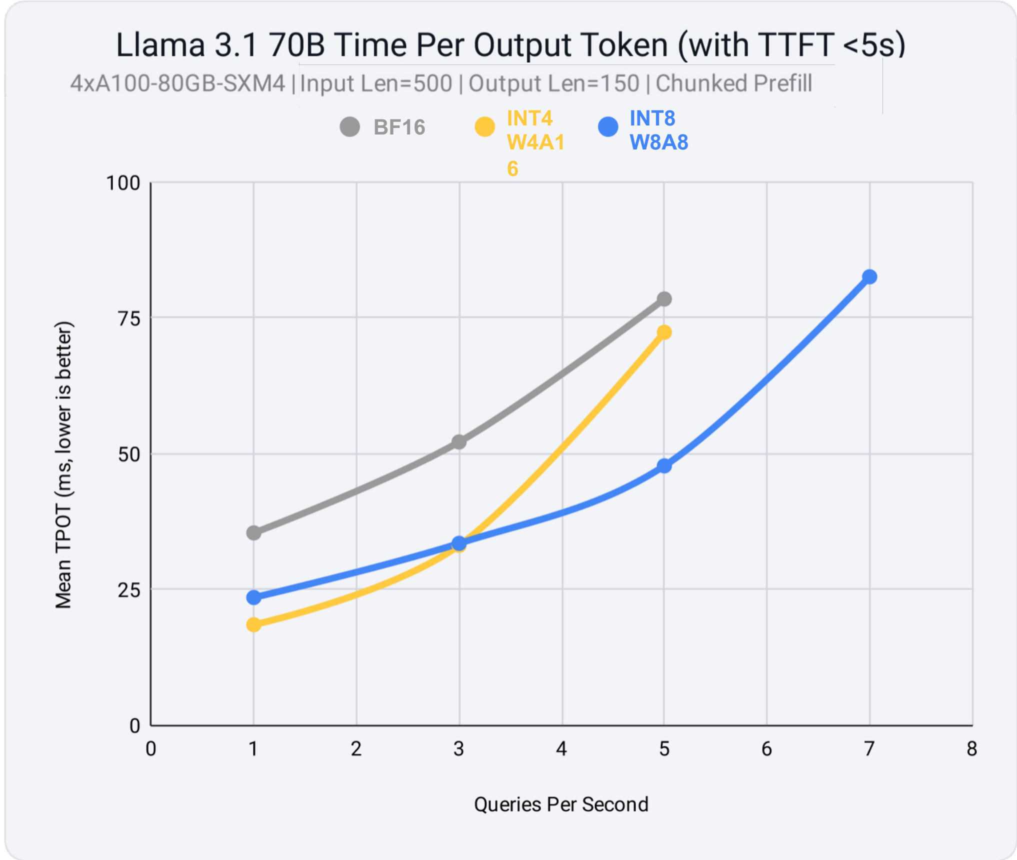
Quantization in vLLM

Use low bit precisions (e.g., FP8, INT8, FP4) to store and compute

1. **Weight Quantization**
   - Reduced storage & memory footprints

1. **Activation Quantization**
   - Faster linear layers with low precision tensor cores

1. **KV Cache Quantization**
   - Reduced KV cache footprint & faster attention



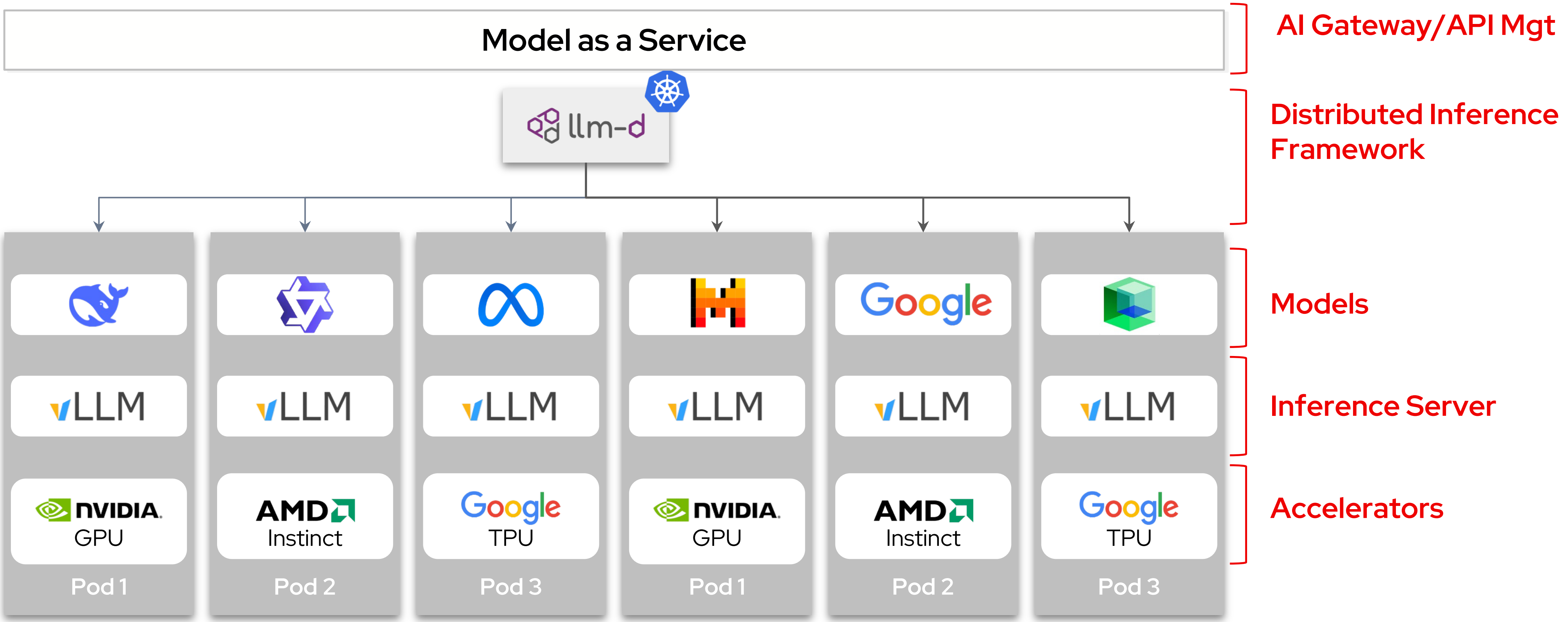
vLLM Combines All Optimizations Together

Without Optimizations		vLLM	
Prompt	<SYSTEM> You are a helpful assistant. ... Keep your answers precise and concise. <USER> Generate a description for this item: ...	Prompt	<SYSTEM> You are a helpful assistant. ... Keep your answers precise and concise. <USER> Generate a description for this item: ...
Output		Output	

Distributed Inference At Scale

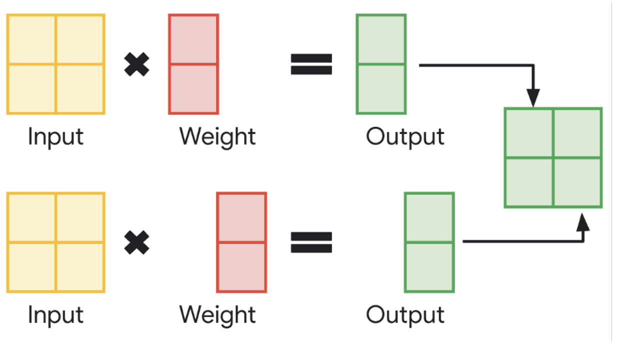
Enterprise GenAI inference platform

Holistic approach to optimize and operationalize deployment and scaling of open-source LLMs

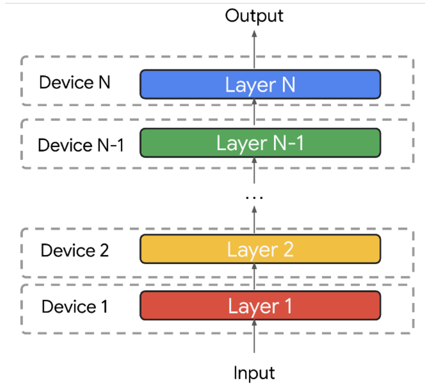


Forms of Parallelism

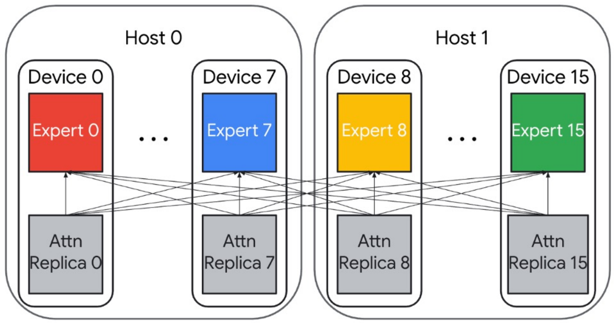
Tensor Parallelism (TP)



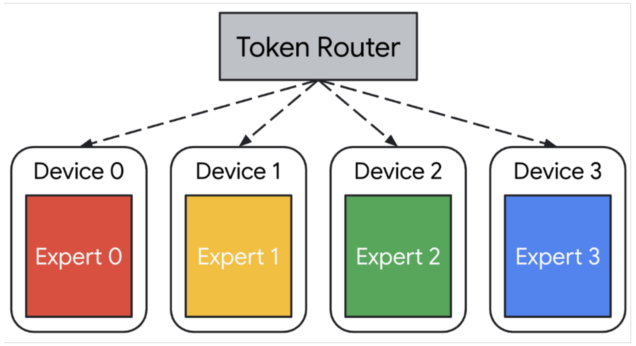
Pipeline Parallelism (PP)



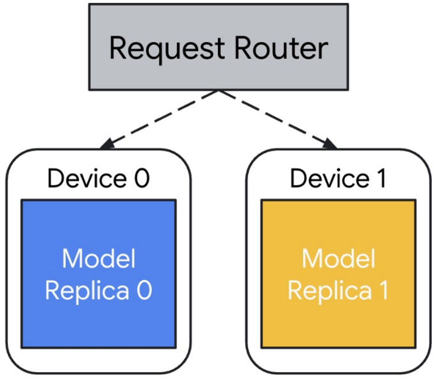
Mixed Parallelism



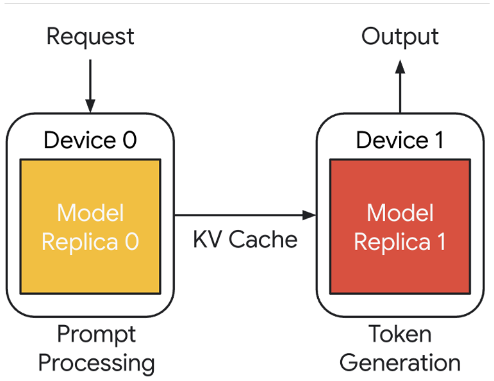
Expert Parallelism (EP)



Data Parallelism (DP)

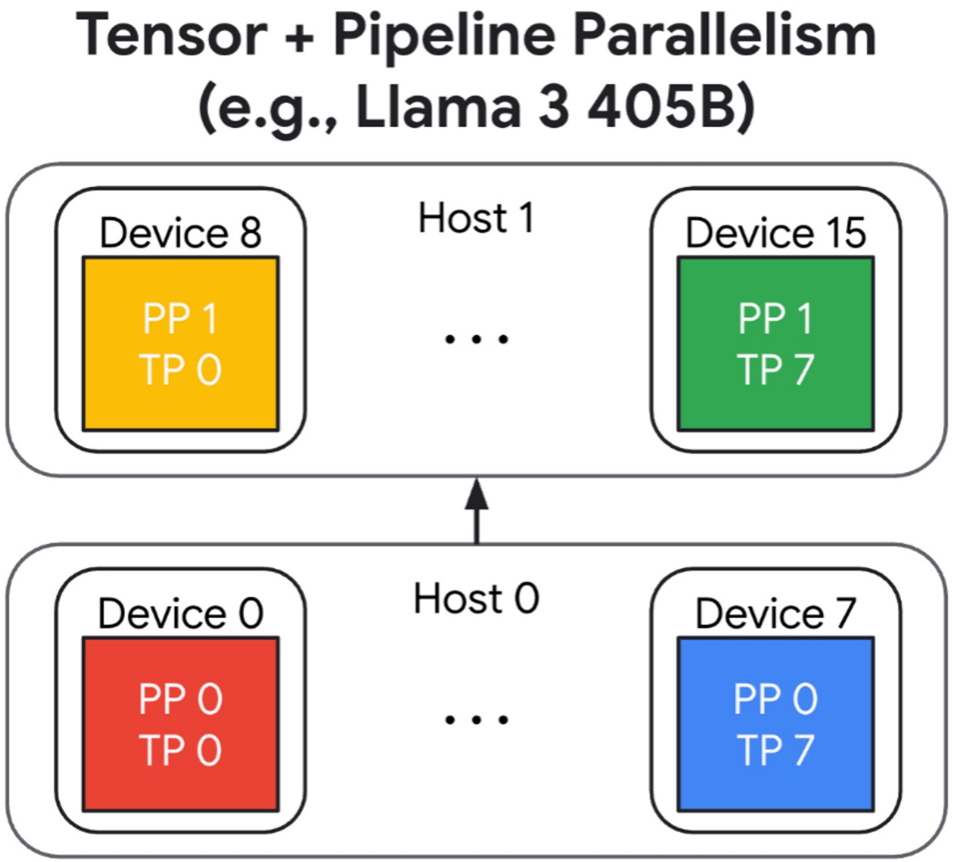


Disaggregated P/D



Example: Mix Parallelism with Red Hat AI

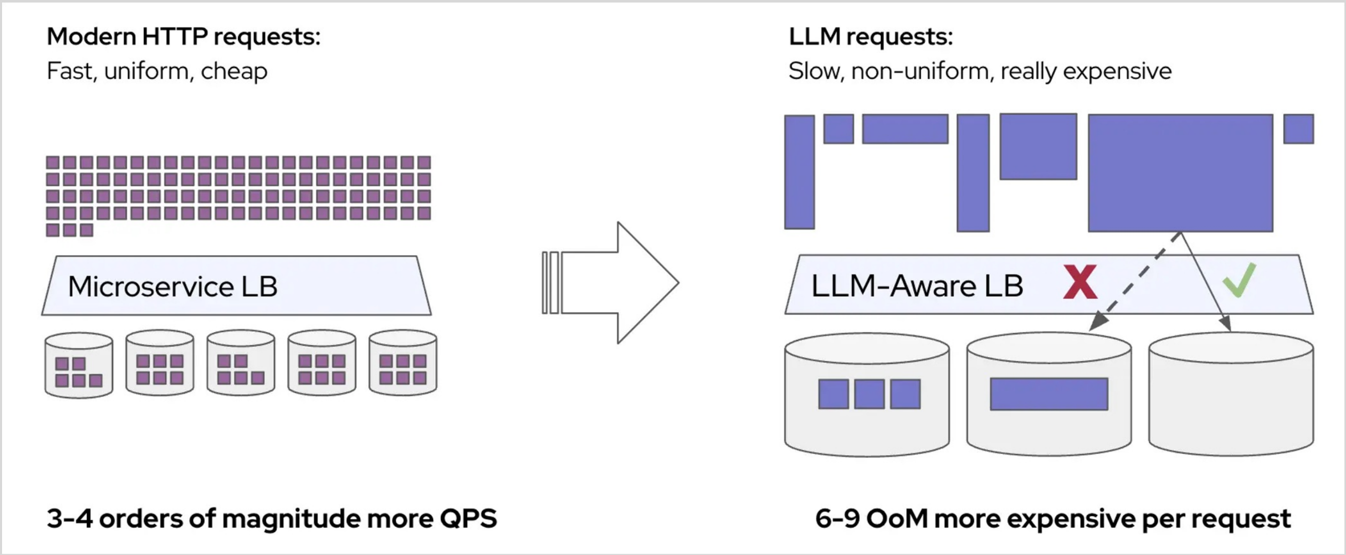
```yaml
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  annotations:
    serving.kserve.io/deploymentMode: RawDeployment
    serving.kserve.io/autoscalerClass: external
  name: vllm-llama3-405b
spec:
  predictor:
    model:
      modelFormat:
        name: vLLM
      runtime: vllm-multinode-runtime
      storageUri: pvc://model-pvc/hf/instruction_tuned
    workerSpec:
      tensorParallelSize: 8
      pipelineParallelSize: 2
    tolerations:
      - effect: NoSchedule
        key: nvidia.com/gpu
```



Scaling Inference

Distributed inference is essential for cost-effective GenAI at scale, but introduces unique **operationalization challenges**

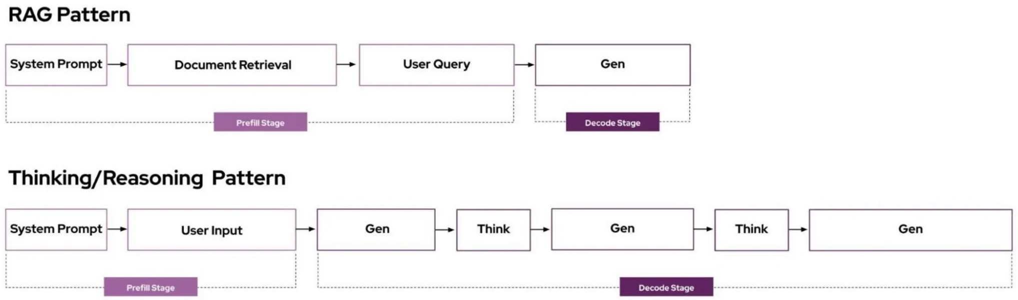
- LLM inference workloads **break traditional** Kubernetes **scaling** due to variable, resource-heavy and hardware-affinity nature of requests
- **Ensuring SLO** (throughput, TTFT, latency) while **minimizing** resource utilization and operational complexity
- Leveraging and managing **heterogenous hardware** for better cost-efficiency
- Distributed **KV cache management** as key part in inference efficiency



maximizing overall system throughput, while meeting SLOs

Why I should care… | Target use cases

Requests with significant variance in resource utilization



LLM inference requests vary in shape—different input/output token lengths cause uneven compute demands.

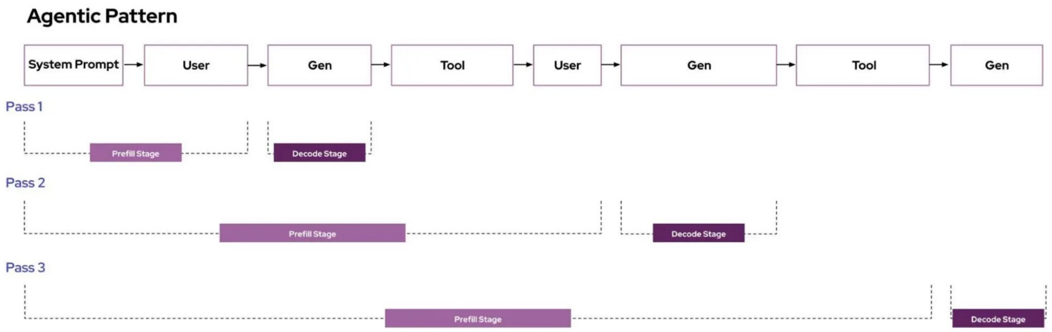
RAG: long inputs (prompt + retrieved docs), short outputs.

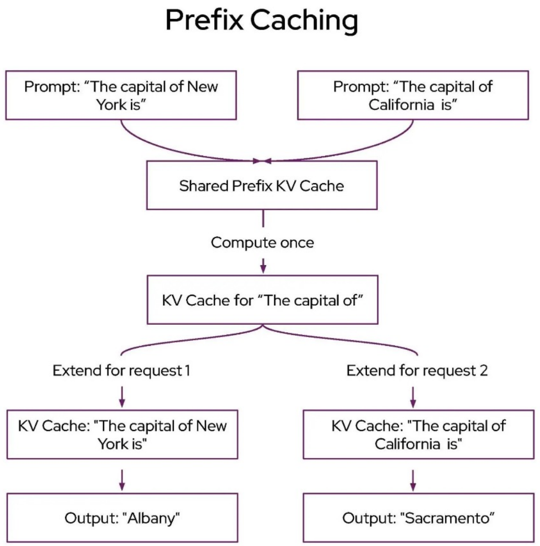
Reasoning: short/medium inputs, long outputs.

These patterns create imbalances across instances, especially during decode

Overloaded instances increase Inter-Token Latency (ITL), creating a feedback loop of worsening performance.

Routing to specific replicas with cached prior computation can achieve orders of magnitude better latency.

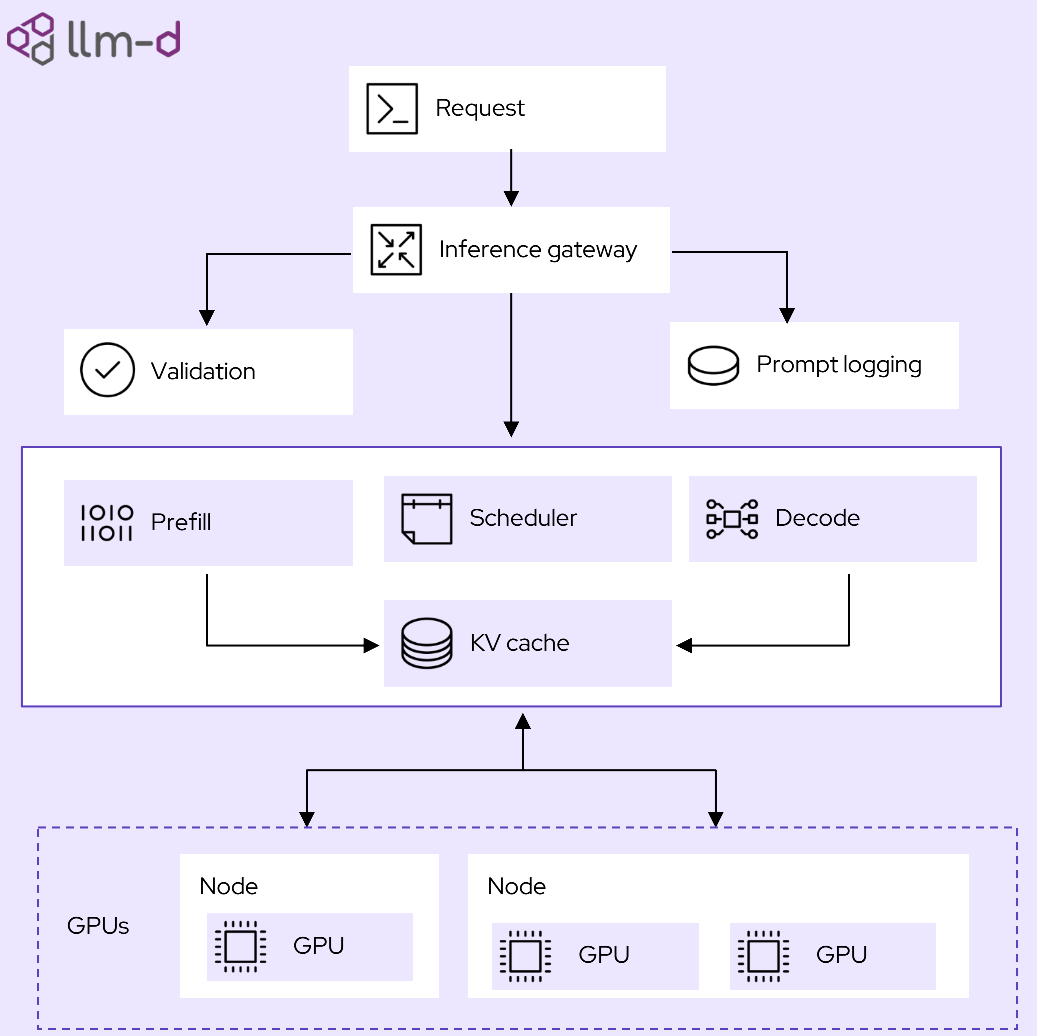




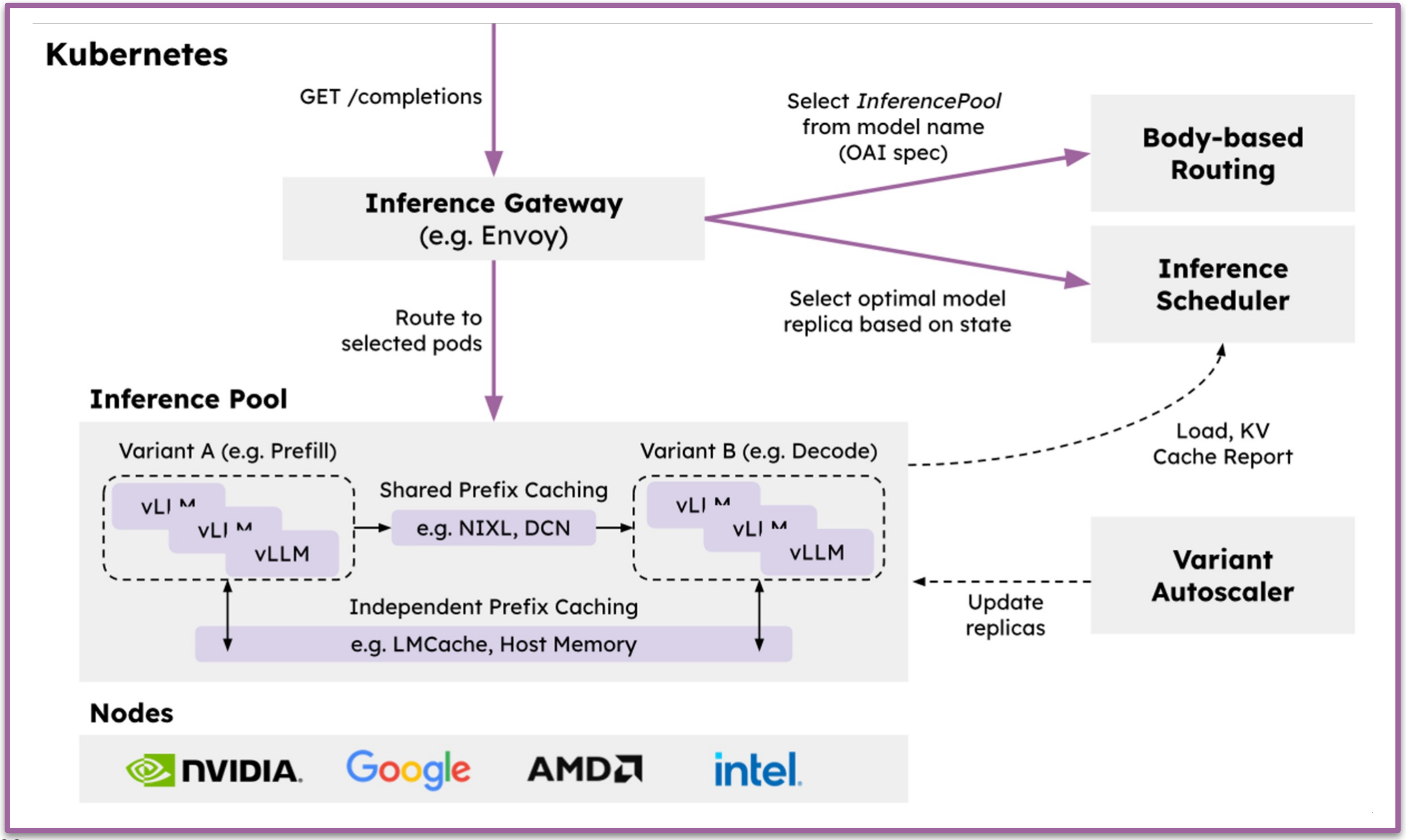
Distributed Inference with llm-d

Maximize GPU Utilization for GenAI: Distributed Inference that Delivers SLOs

- Joint open source project by **Red Hat, Google**, **NVIDIA**, **AMD**, **Hugging Face**, and many more
- **Kubernetes-Native** Architecture for simple deployment and management of GenAI models
- **Optimized GenAI Inference** to accelerate LLM's and MoE
- **Intelligent Resource Utilization** to reduce inference costs
- **High Performance and Scalability** to meet demanding Service Level Objectives (SLOs).
- Supported on **Heterogeneous Hardware** like NVIDIA and AMD GPUs (and many more to come in the future)



vLLM 🤝 llm-d 🤝 kubernetes



Operationalizability

- Modular and resilient architecture with native integration into Kubernetes via Inference Gateway API

Flexibility

- Cross-platform with extensible implementations of key composable layers of the stack

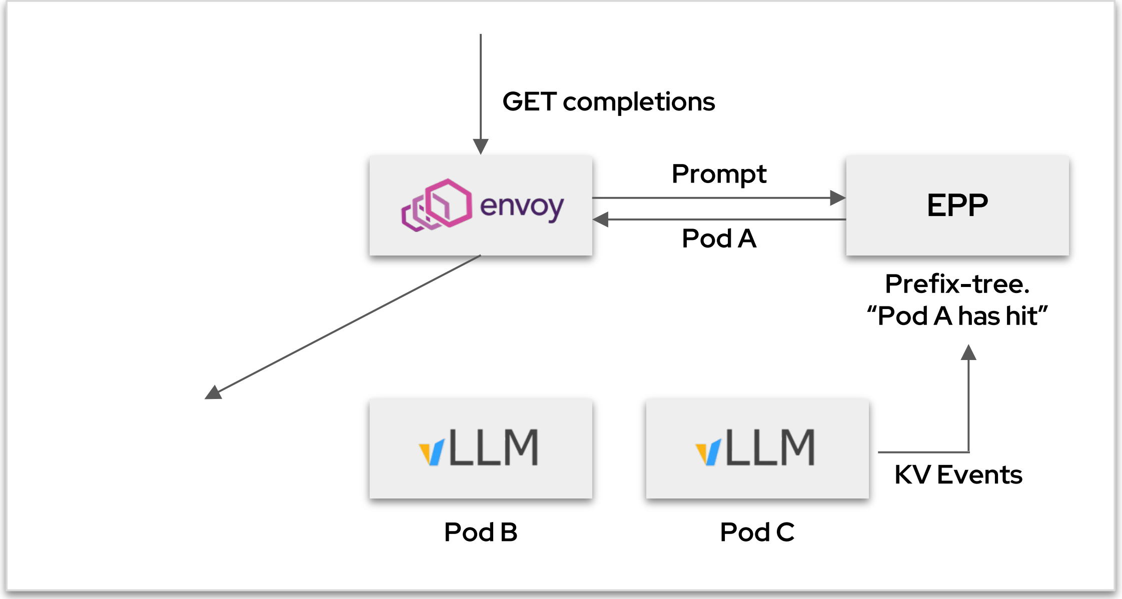
Performance

- Leverage distributed optimizations like prefix-aware routing and disaggregation to achieve the highest throughput while meeting SLOs

"Well-lit" Path: Intelligent Inference Scheduling

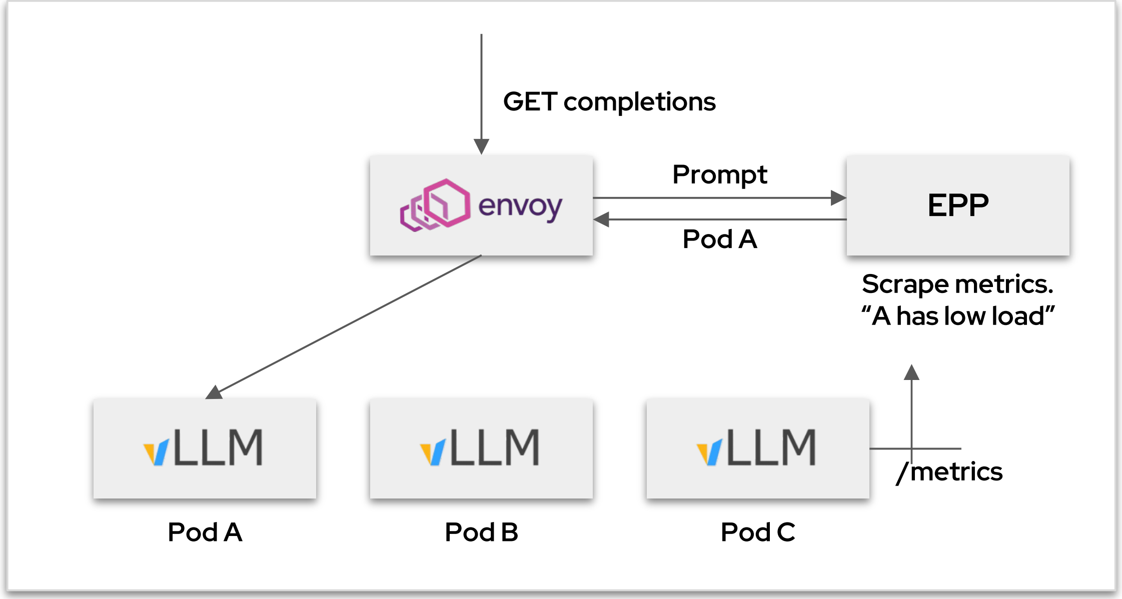
vLLM-aware load-balancing enables smarter request routing that improve SLOs

Prefix-Aware Routing



Dramatically increase prefix-cache hit rate

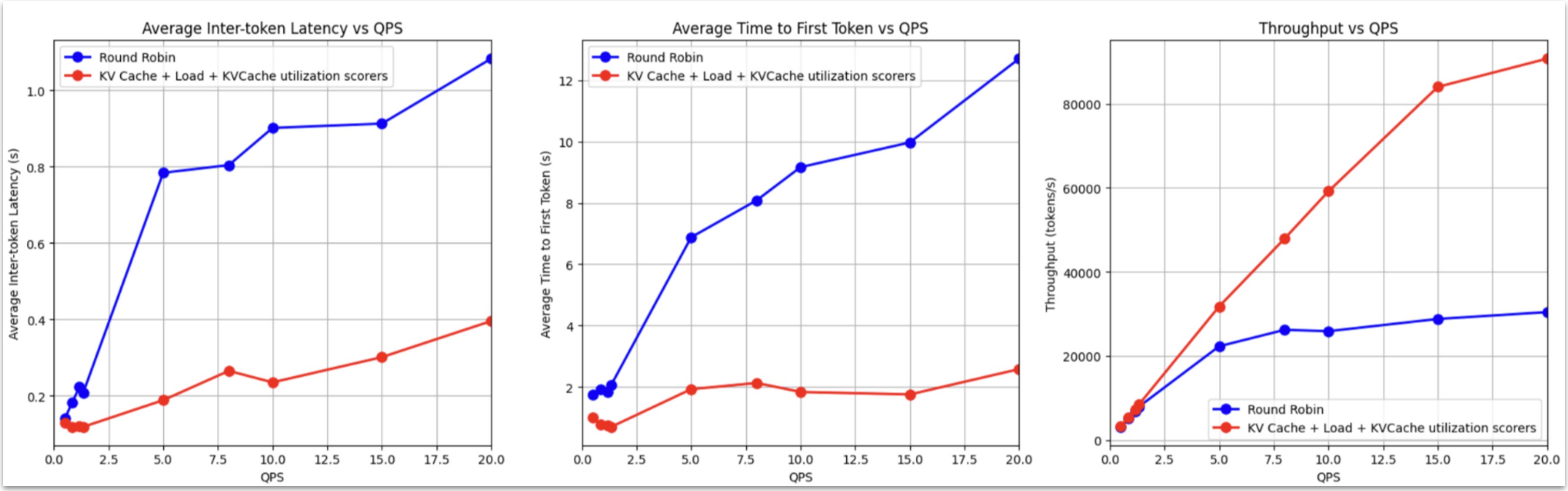
Load-Aware Routing



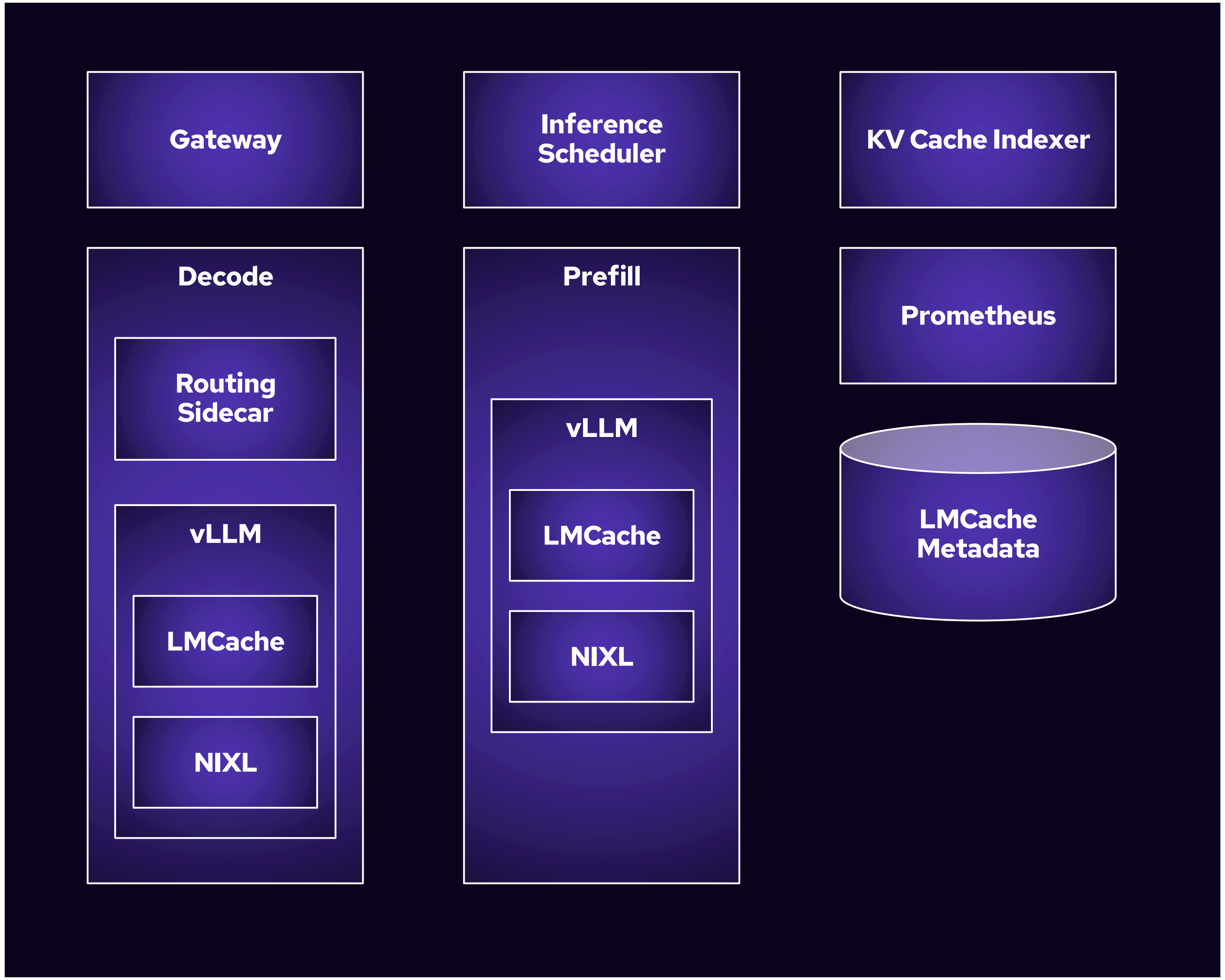
Load-balancing based on actual replica state

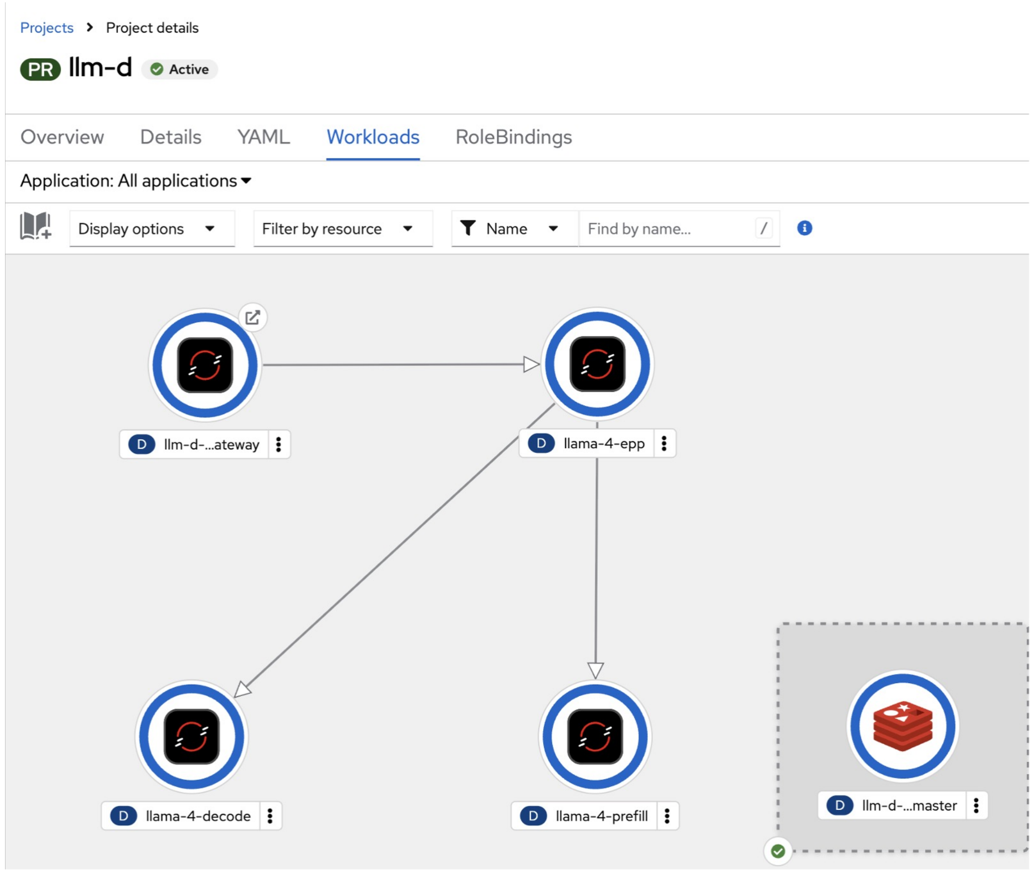
Intelligent Inference Scheduling

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

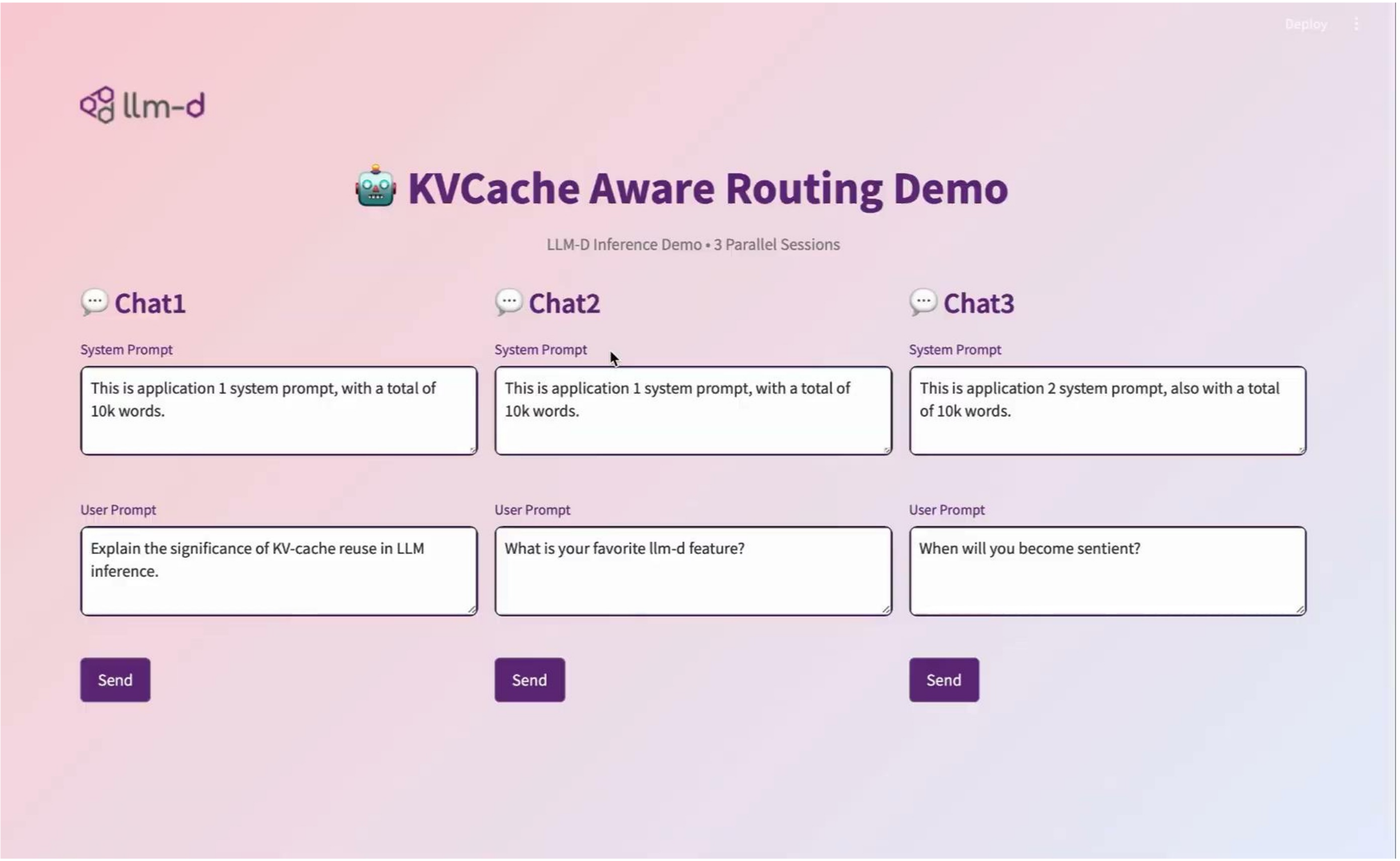


Request flow example...



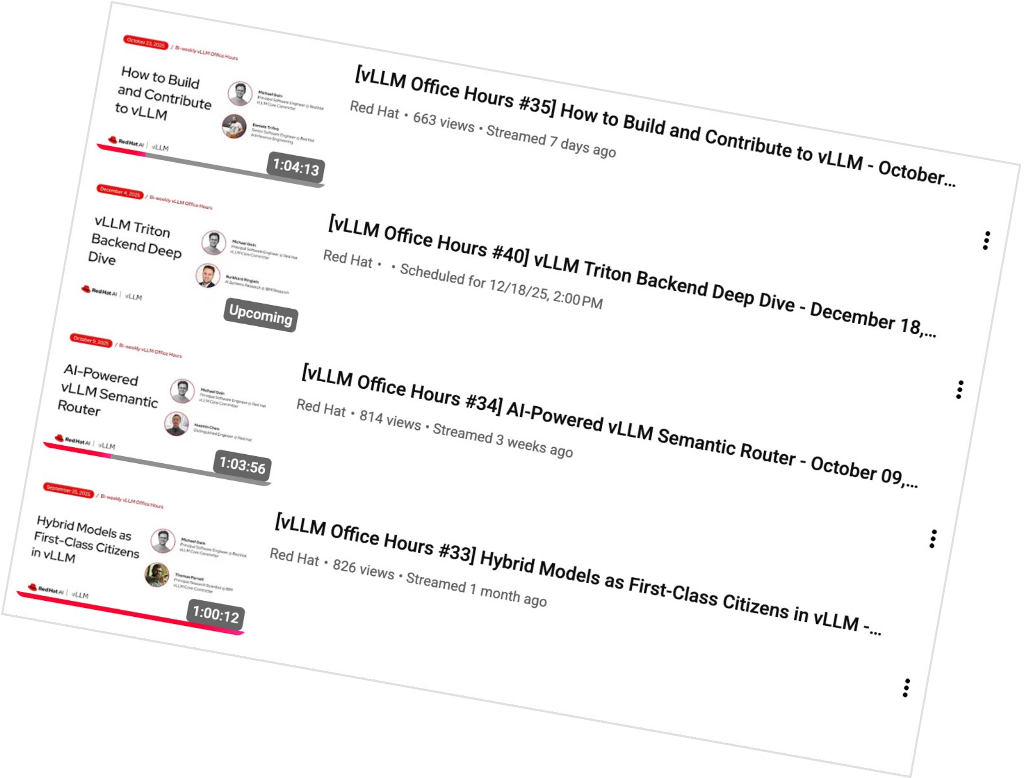


Demo


Deploy
llm-d
KVCache Aware Routing Demo
LLM-D Inference Demo • 3 Parallel Sessions
Chat1
System Prompt
This is application 1 system prompt, with a total of 10k words.
User Prompt
Explain the significance of KV-cache reuse in LLM inference.
Send
Chat2
System Prompt
This is application 1 system prompt, with a total of 10k words.
User Prompt
What is your favorite llm-d feature?
Send
Chat3
System Prompt
This is application 2 system prompt, also with a total of 10k words.
User Prompt
When will you become sentient?
Send

Join Bi-Weekly vLLM Office Hours [Virtual]

- Happening every other Thursday at 20:00 CET
  - Watch on-demand immediately!
- Hear the bi-weekly vLLM update
- Give feedback & ask questions
- Deep dive into cutting-edge topics to accelerate your vLLM inference



Sign up →

vLLM Red Hat

Red Hat Summit

Connect

Thank you

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat