THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA
Embedded Linux Conference

KV-Cache Centric Inference

Building an Open Source LLM Serving Platform Around State

Maroon Ayoub, IBM Research
Martin Hickey, IBM Research

Martin Hickey

GitHub: @hickeyma

- 30 yr tech career in enterprise and open source software
- Currently focusing on Cloud Native Computing and AI inference
- Core maintainer in LMCache
- Emeritus Helm core maintainer and TOC member
- Contributor to vLLM, llm-d, LMCache, Helm, Kubernetes, Open Telemetry, OpenStack
- Open source developer at IBM Research

GitHub: vMaroon

Maroon Ayoub

- Staff Research Scientist, IBM Research
- Co-lead of llm-d - an open source project for distributed LLM inference
  - KV-disaggregation SIG chair
- Focused on evolving agentic-inference
- Previously worked on Kubernetes, hybrid cloud, and distributed systems

LLM Scaling Inference Challenges

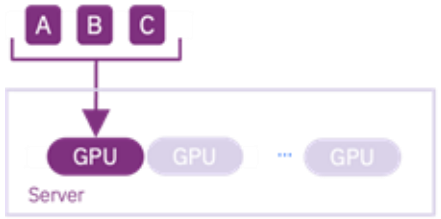
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
Embedded Linux Conference
NORTH AMERICA

The challenge of scaling inference

Distributed inference is essential for cost-effective scaling of AI, but introduces unique **operationalization challenges**

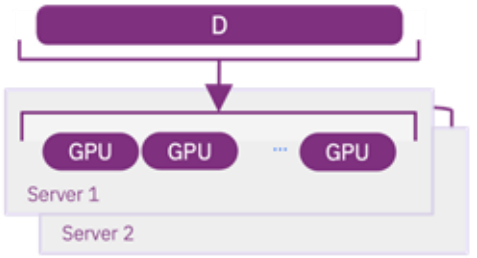
- Load balancing **variable, resource-heavy** and hardware-affinity nature of requests
- **Ensuring SLO** and maintaining low latency under multi-tenant traffic
- Leveraging and managing **heterogeneous hardware** for better cost-efficiency
- Managing distributed **KV cache state** at scale as key part in inference efficiency
- **Orchestrating** prefill and decode phases; and multi-node serving

Multiple small models
(e.g., granite-8b, mistral-7b, ...)

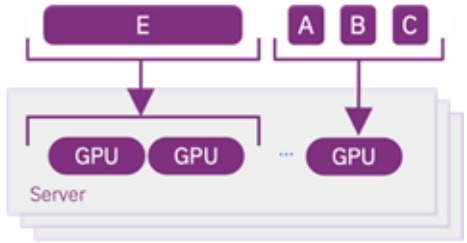


Single large model
(e.g., lama3-70b, llama4,

)



Mix of small and large models



KV cache management

KV cache reuse is essential to make prefix-heavy workloads **cost-effective**

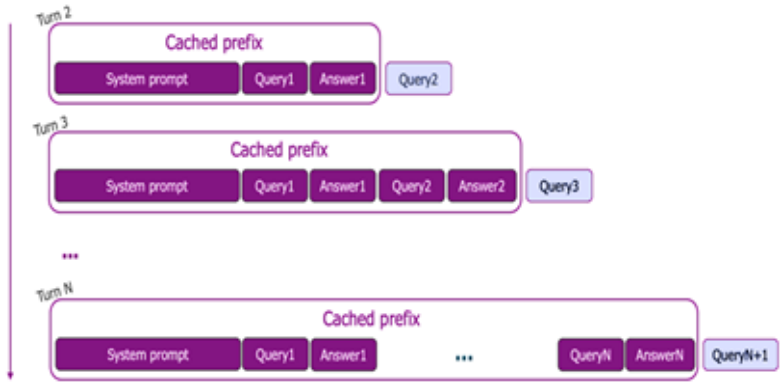
KV cache size grows linearly with context length and batch size

E.g., Llama 3.1 8B (fp8), 32K–128K context length results in 16–64 GB KV-cache size increase

KV cache grows to hundreds of terabytes for chatbot and agentic AI

Conversational AI

Large input-to-output ratio

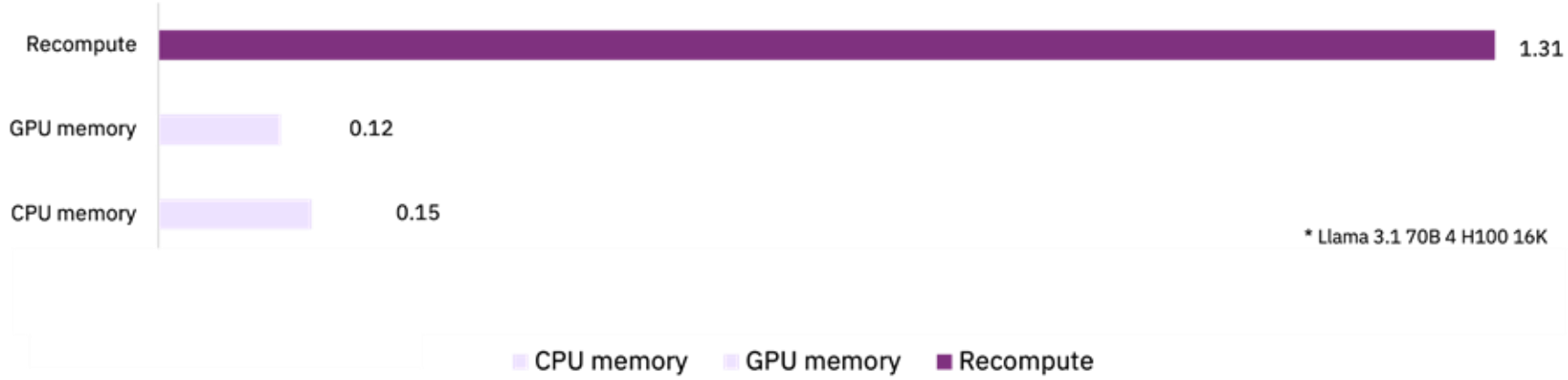


Agentic workflows

Very large input-to-output ratio (100:1)

Inference Time - TTFT (sec)

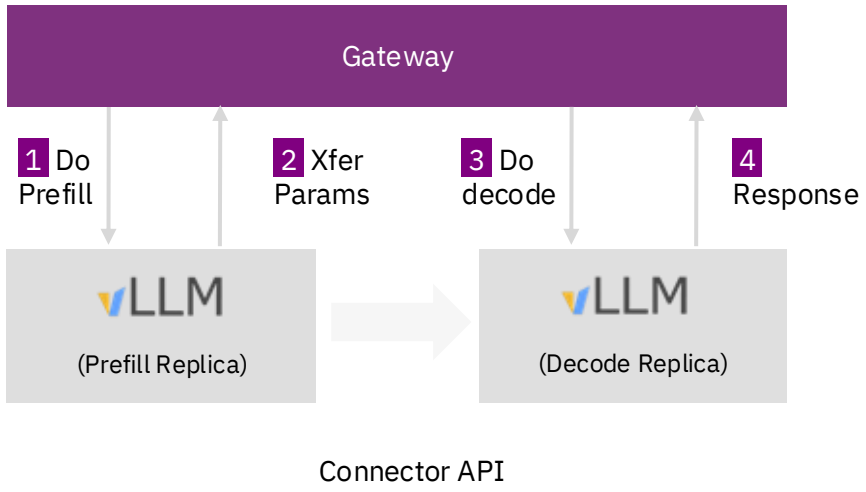
Inference speed-up with KV reuse



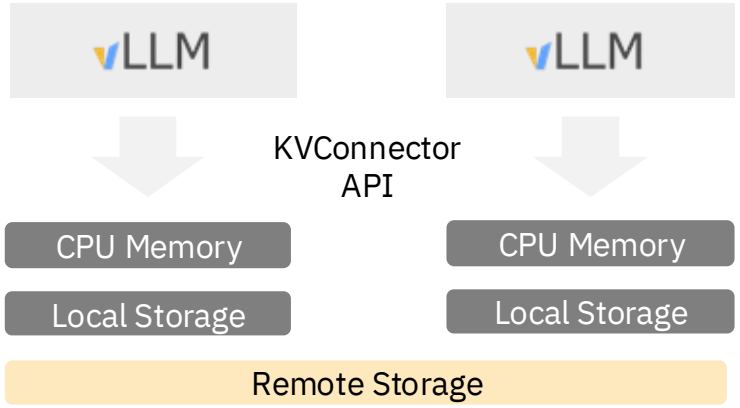
Pipeline imbalance: prefill and decode behave very differently

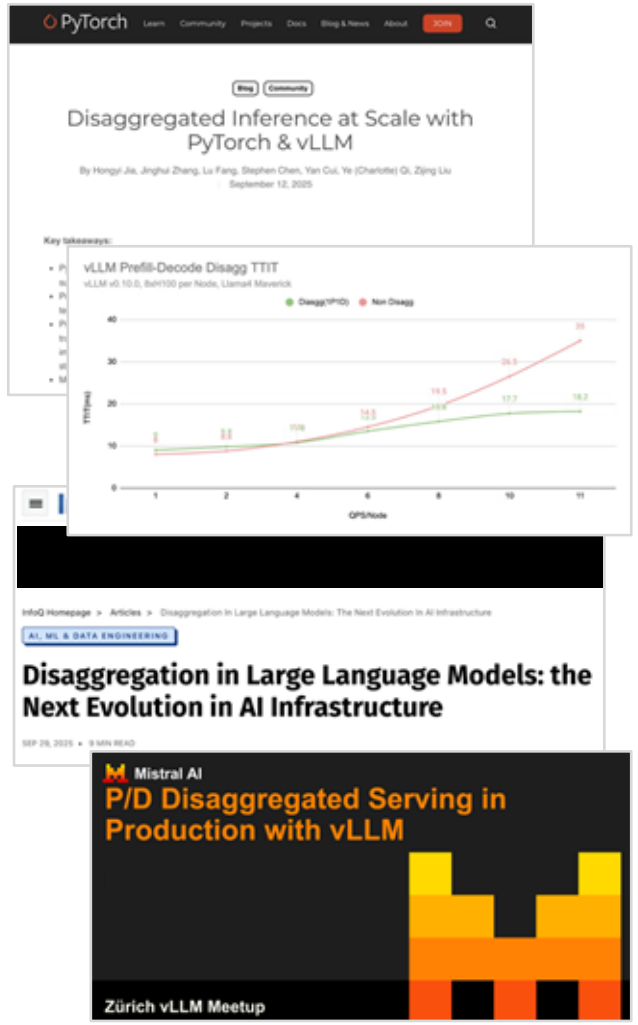
P/D disaggregation, from exploration to production operation

Prefill/Decode Disaggregation



KV Cache Offloading





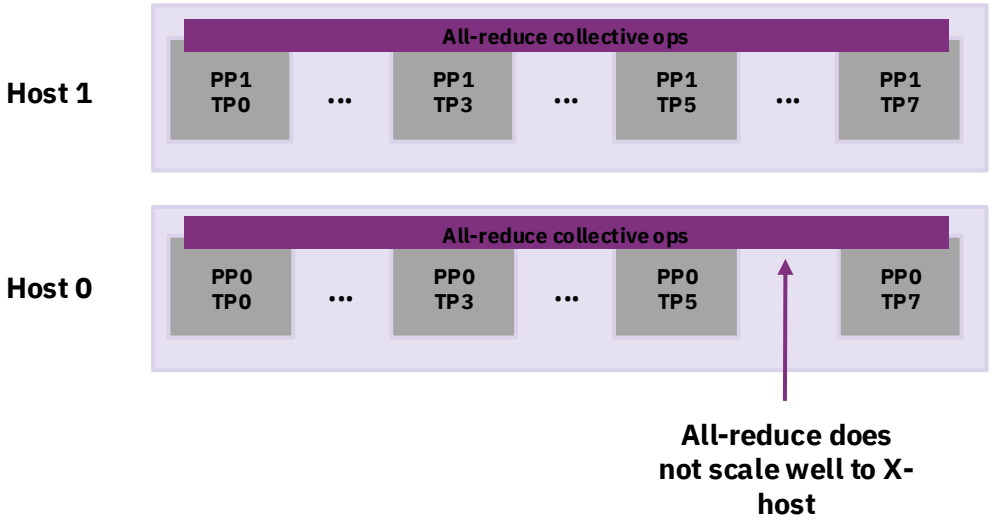
Key challenges on implementing and standardizing the optimizations/APIs needed for **disaggregated serving orchestration**

The communication wall: scaling MoE across nodes

Sparse MoE architectures scale to multi-node: this enables running **very large models** efficiently

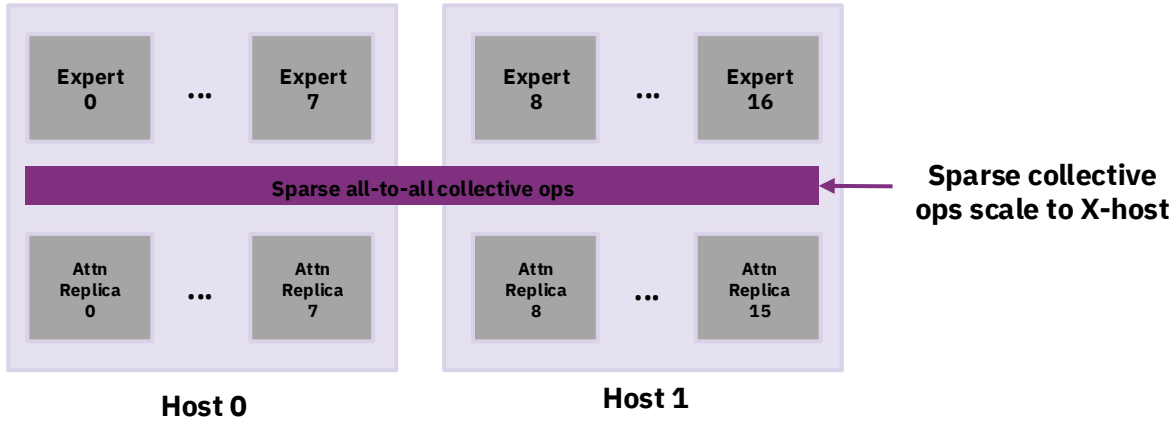
Dense: Tensor + Pipeline Parallel

e.g. Llama-405B



Sparse MoE: Data + Expert Parallel

e.g. DeepSeek-R1



Key challenges is on building a platform that can orchestrate these distributed systems behaviors with **operational simplicity**

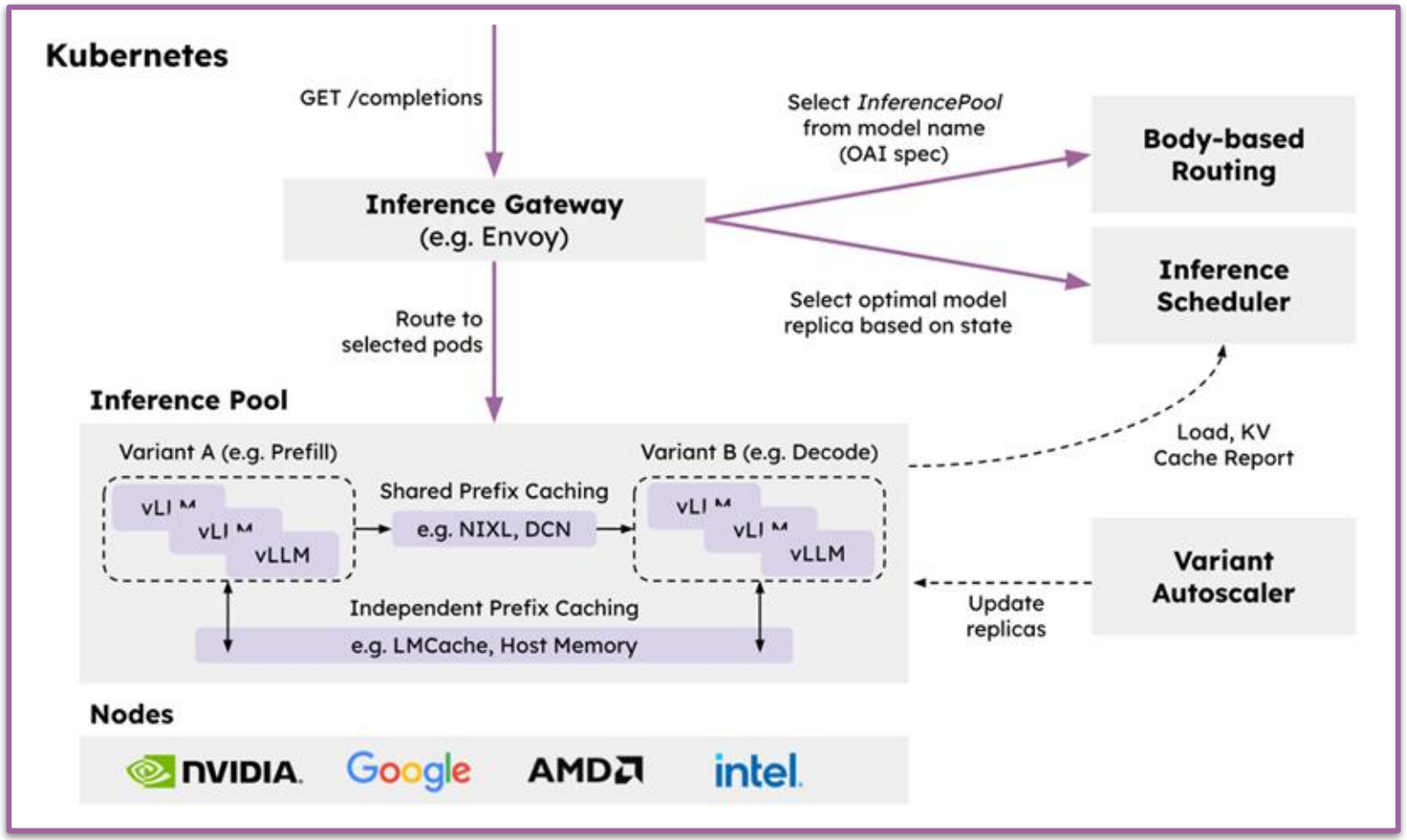
Introducing llm-d

High Performance distributed scalable LLM inference platform

github.com/llm-d/llm-d

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA
Embedded Linux Conference

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

What are the Key llm-d Capabilities for Enhanced Distributed LLM Workloads?

https://llm-d.ai/docs/guides

01

Intelligent Inference Scheduling

Prefix-aware + load-aware routing
via pluggable EPP scorers

02

KV Cache Management

GPU → CPU → Remote Storage tiers
with global prefix index

03

Prefill / Decode Disaggregation

Separate compute-optimized prefill
from bandwidth-optimized decode fleets

04

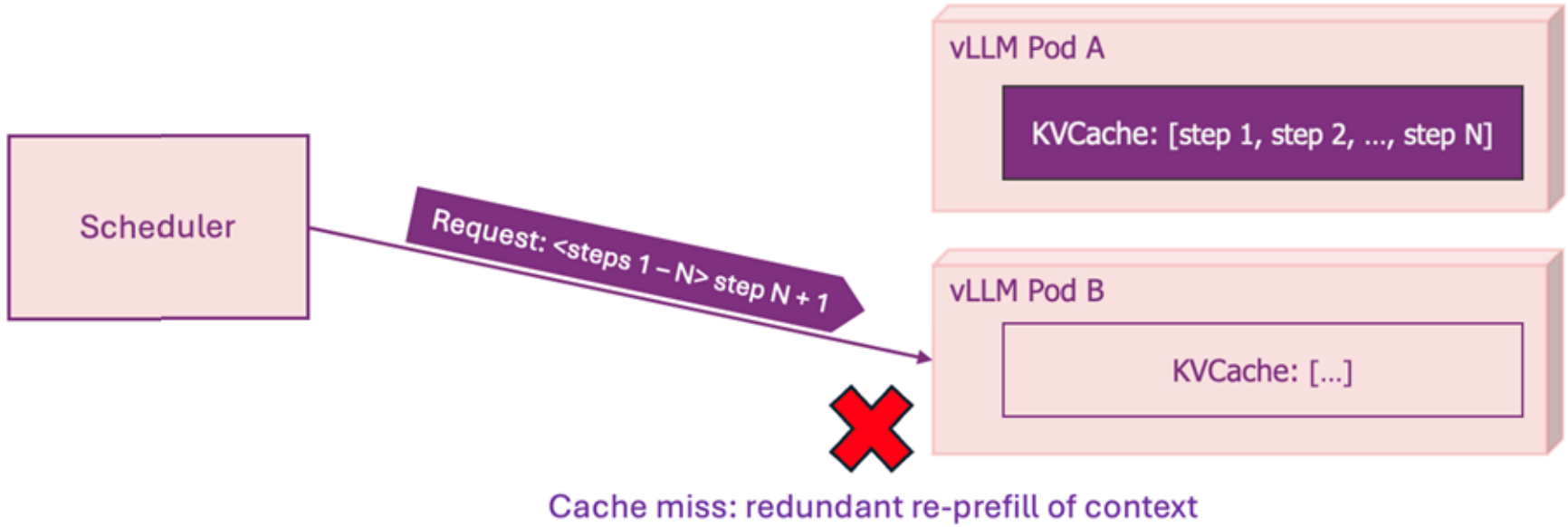
Wide Expert Parallelism

Kubernetes-native MoE routing
with DP-aware EPP

Where Standard Load-Balancing Fails

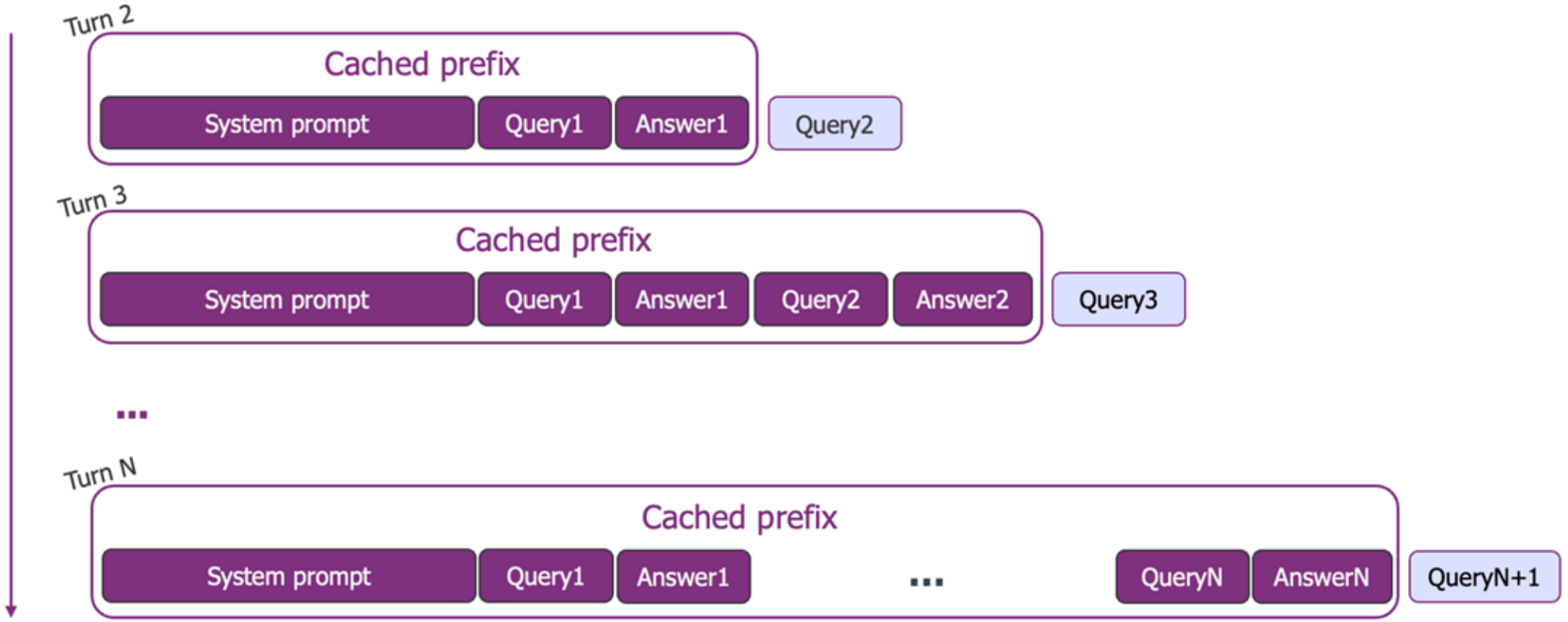
Production AI inference involves multiple vLLM instances and replicas

Naive load-balancing distributes traffic evenly across the pods -> destroying cache optimizations



Prefix-Cache Reuse in Production

Multi-Turn Chat



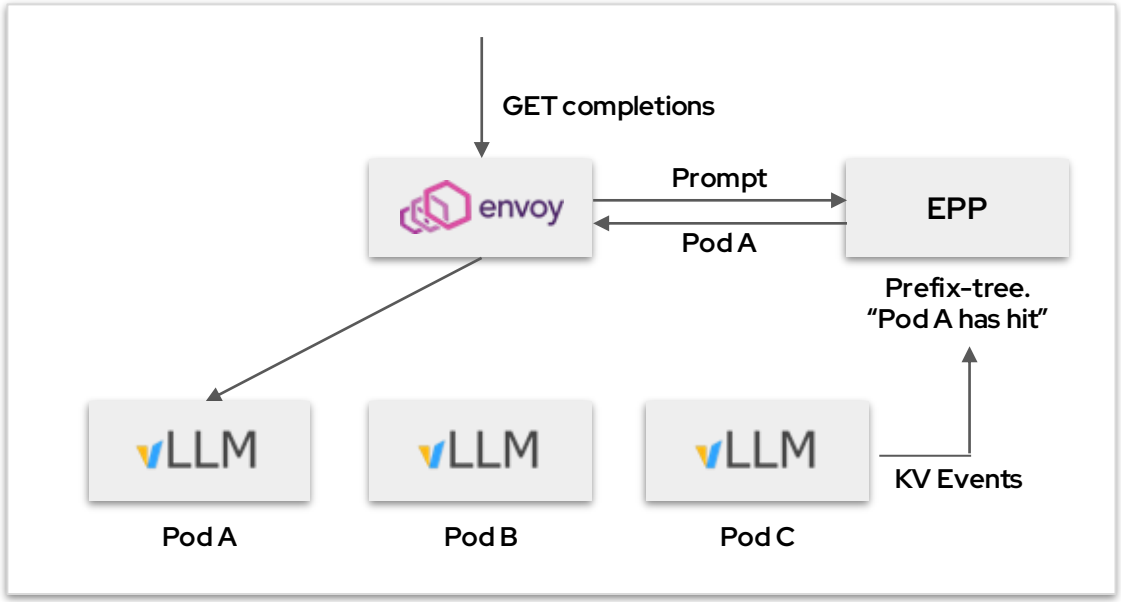
Prefix-Cache Reuse in Production

Linear Agents – Multi-Turn Tool Use

Intelligent Inference Scheduling

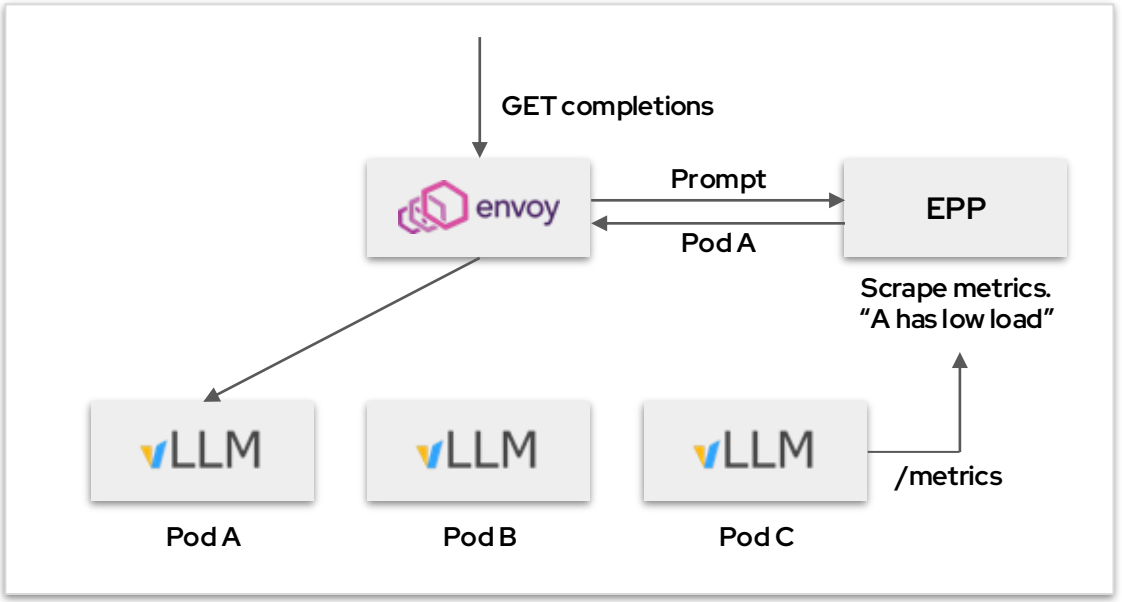
vLLM-aware load-balancing enables smarter request routing that improve SLOs

Prefix-Aware Routing



Dramatically increase prefix-cache hit rate

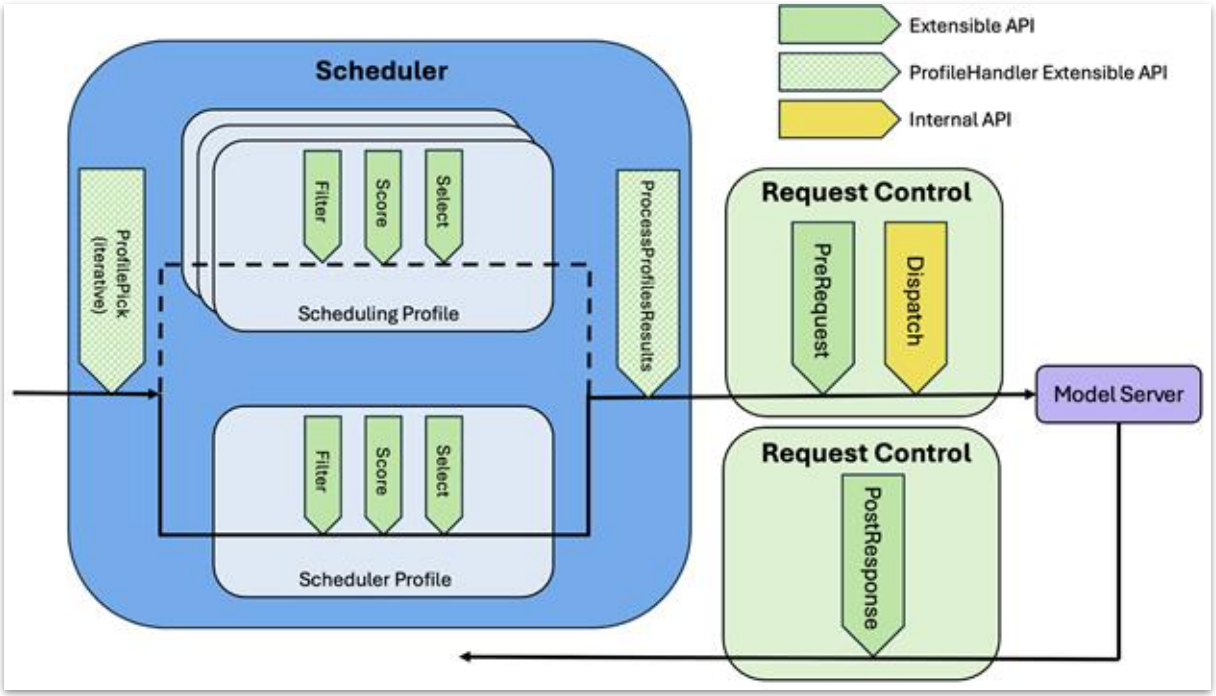
Load-Aware Routing



Load-balancing based on actual replica state

Intelligent Inference Scheduling

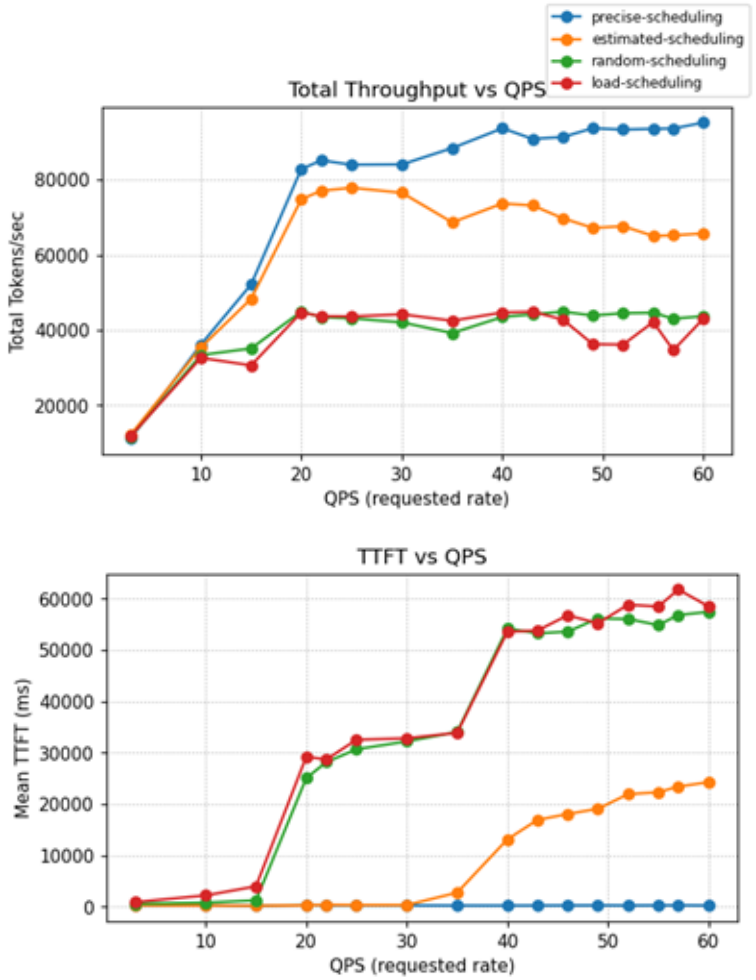
llm-d's pluggable architecture enables first-class composing and extending



- **Scorers** → Apply weights to signals
  - Configure relative weights
  - Add custom scorers as needed
- **Filters** → filter specific pods
- **Request Control** → queue requests until the server load is ready

Intelligent Inference Scheduling

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

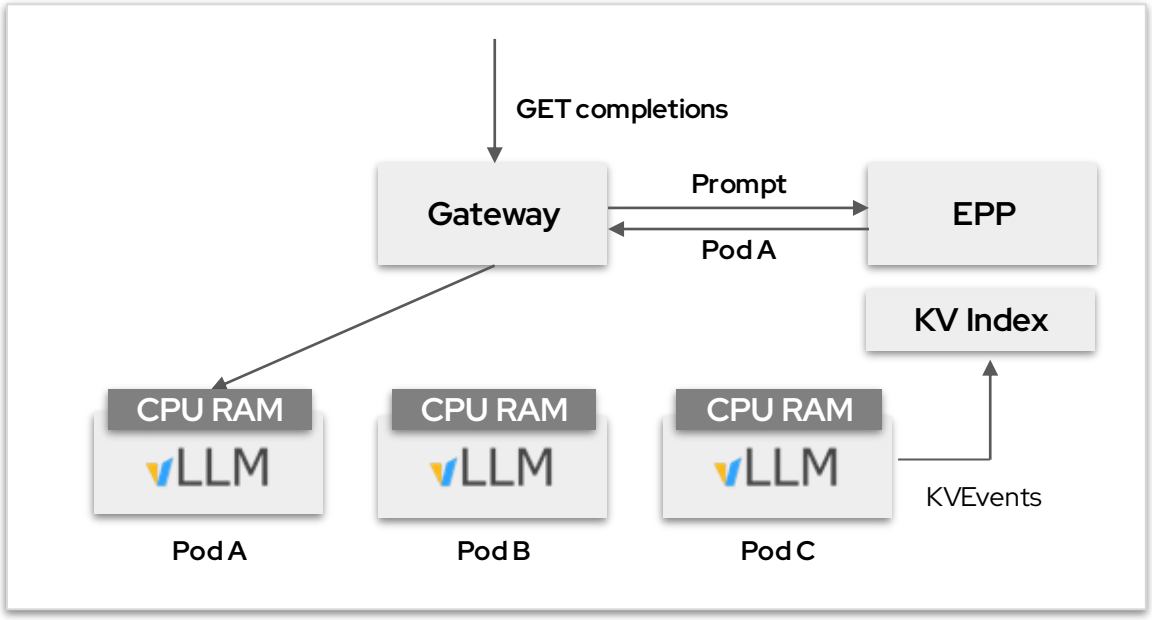


- llm-d provides different levels of **prefix-cache aware scheduling** capabilities in our **guides**
- In a workload that demands only 73% of a 16 H100 GPUs KV-cache capacity, **optimal performance is achieved**:
  - **x57 faster response time** than naive scheduling
  - **x2 throughput** than load-aware scheduling

KV Cache Management

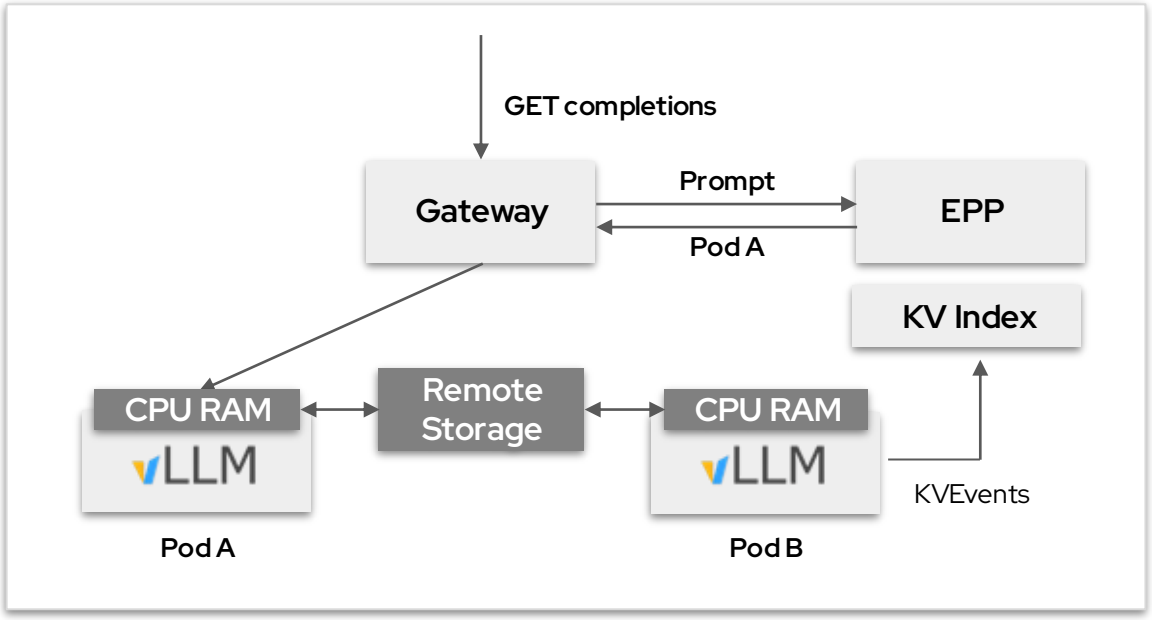
Leverage all system resources to maximize prefix cache hit rate within the cluster

North/South KV Management



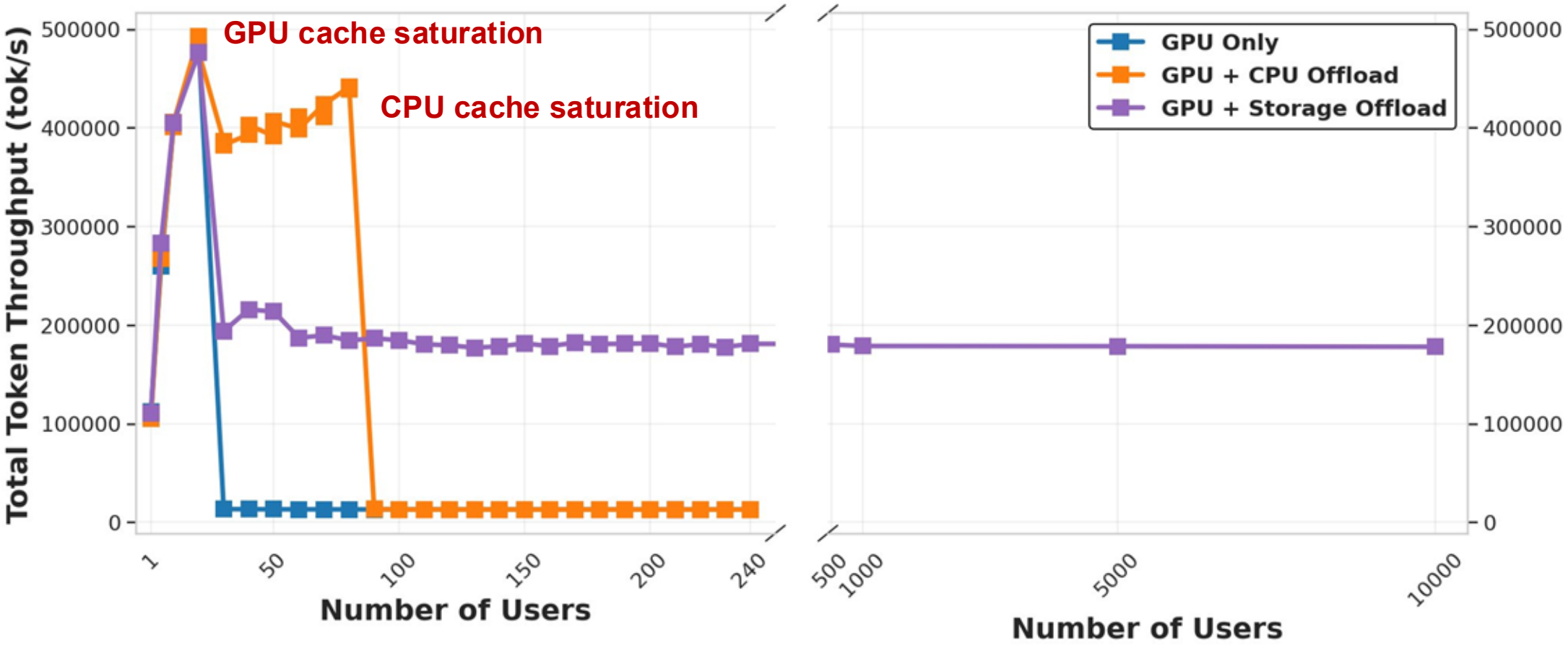
Offload KV caches to local memory with global index

East/West KV Management

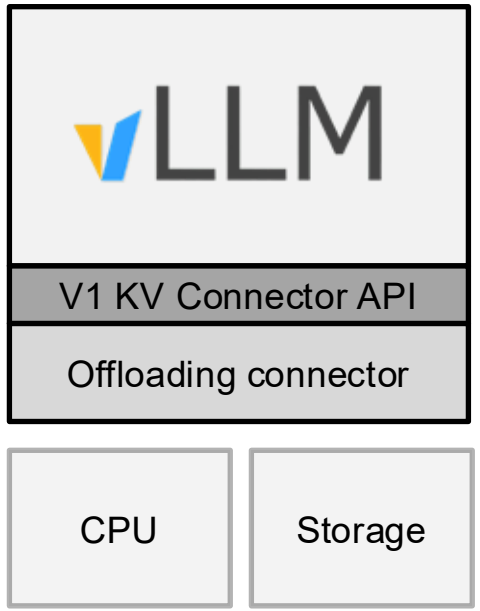


Offload KV caches to global shared remote storage

KV-Offloading Performance

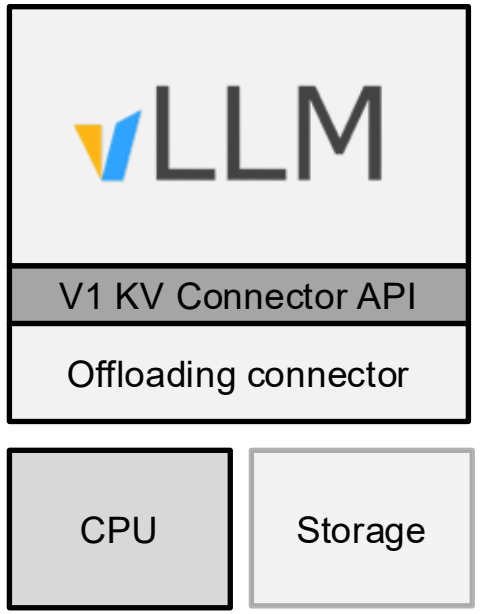


The KV Offloading Connector



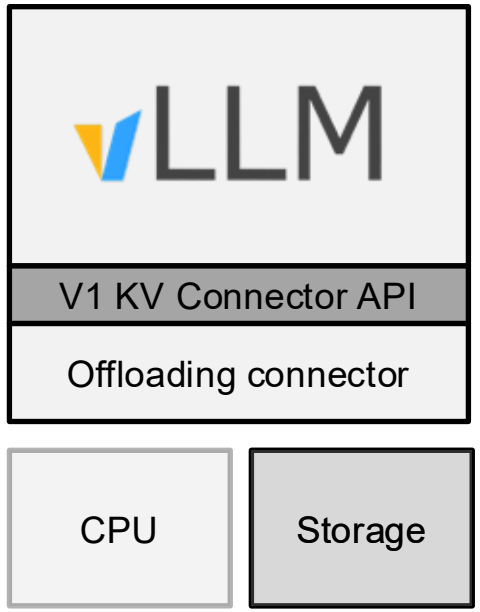
- **Bridges the vLLM V1 KV-Connector API** into a simple, unified offloading interface for **pluggable backends**
- Enables KV cache to live **beyond GPU memory** across CPU and storage tiers
- Improves **throughput** and **stability** by reducing GPU memory pressure

CPU Offloading Backend



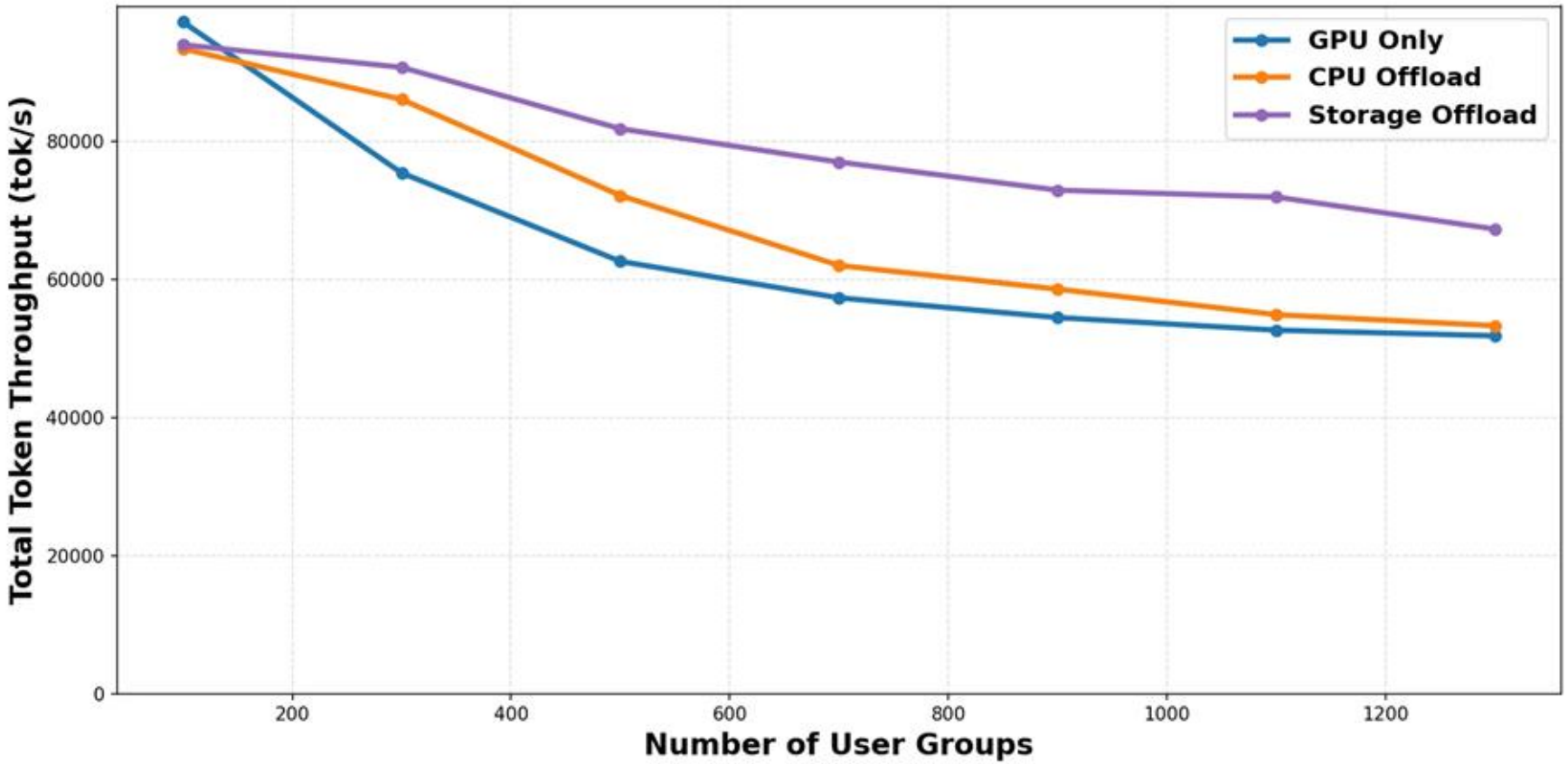
- KV blocks are **copied asynchronously** from GPU -> pinned CPU memory **using DMA**
- **CPU memory acts as a high-capacity KV cache tier** extending the number and lifespan of token sequences
- **When a request resumes, KV blocks are streamed back just-in-time** instead of recomputing attention from scratch

Storage Offloading Backend



- KV blocks are **copied asynchronously** from GPU -> shared storage using the same offloading connector (C++)
- **Storage acts as a cluster-wide KV cache pool** extending cache capacity beyond any single node's GPU or RAM
- **Any POSIX filesystem is immediately supported**

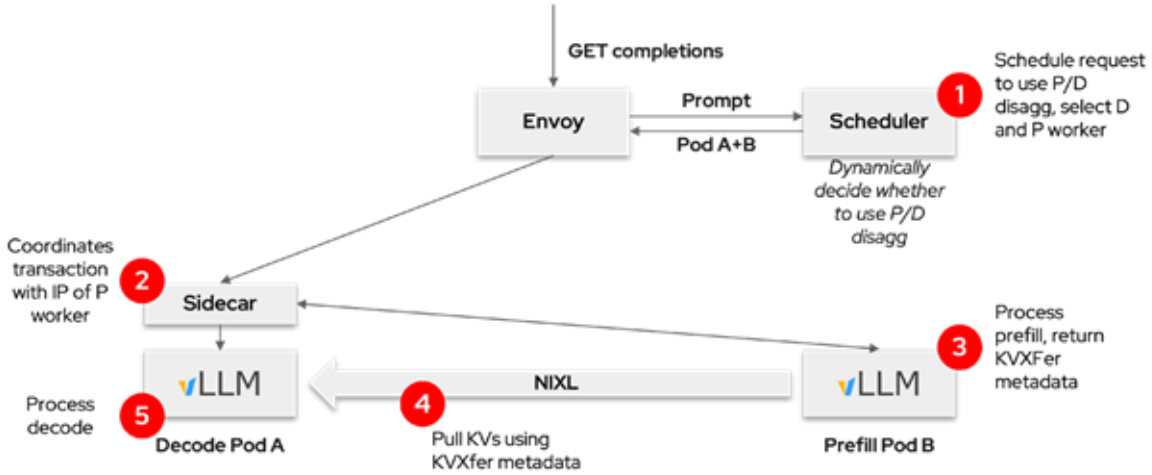
Offloading Performance

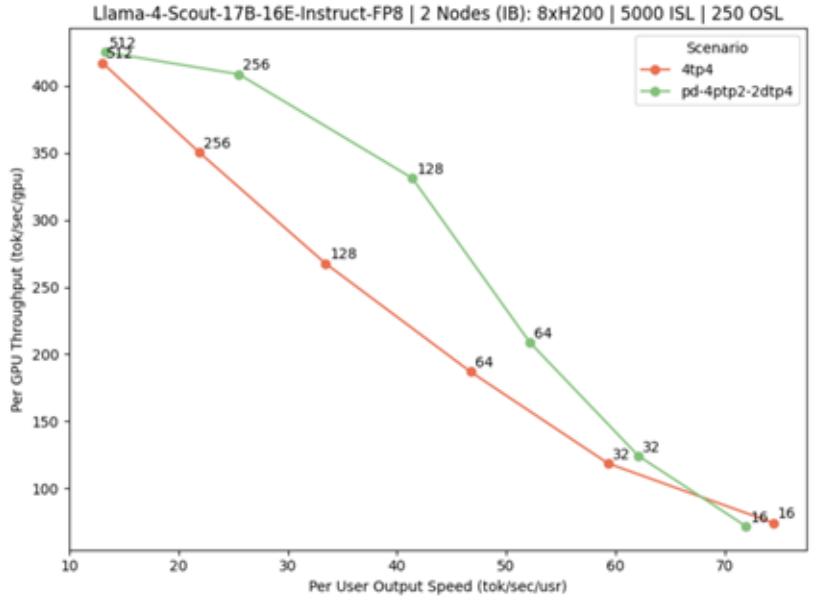


llm-d blog: Native KV Cache Offloading to Any Filesystem with llm-d

Prefill/Decode Disaggregation

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



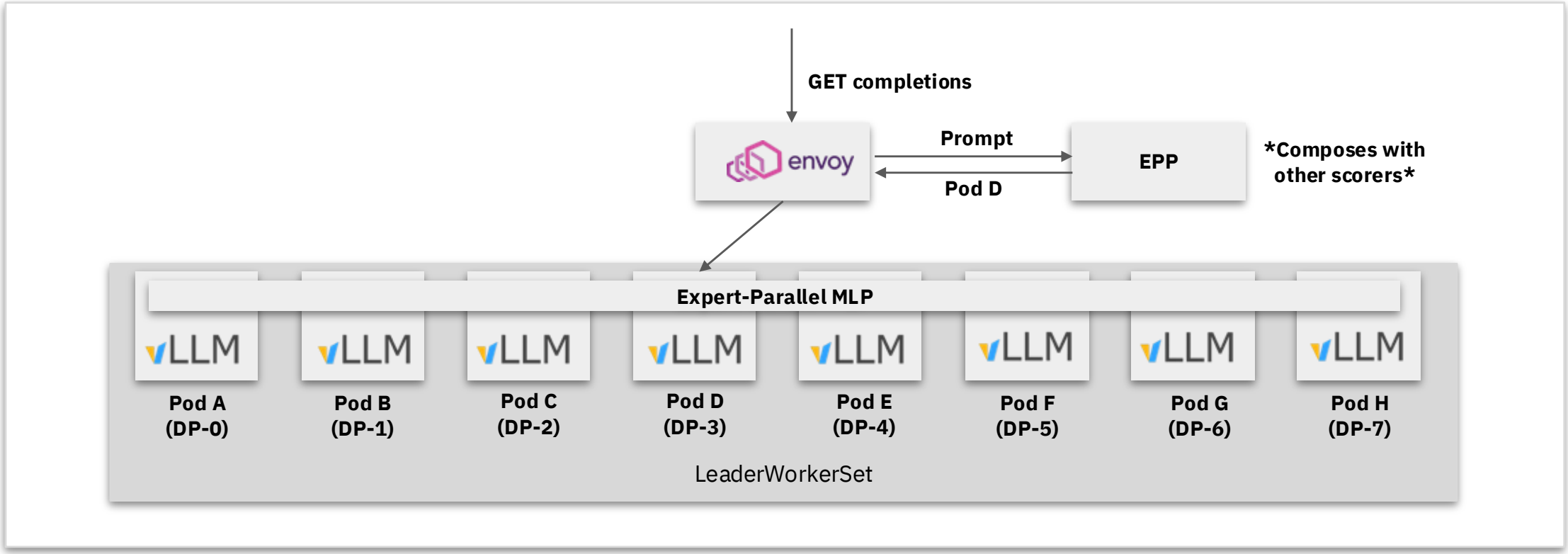


- Separate prefill and decode
- Allows specialization
- Dynamically decided
- Easier to meet TTFT and ITL SLO requirements

- Best for “medium” server load
- Typical: small prefillers, large decoders
- P/D benefits are most clear when jointly optimizing efficiency and interactivity

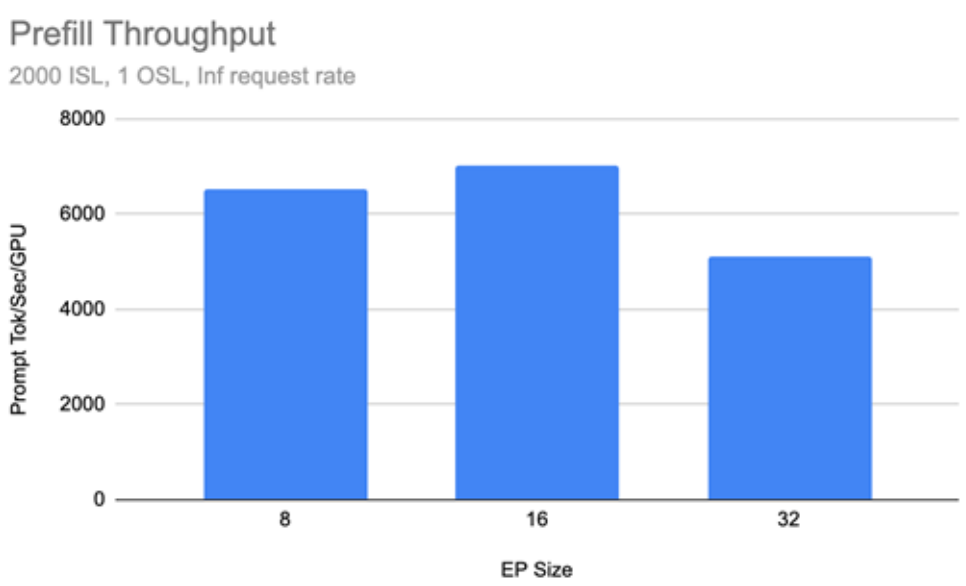
Wide Expert Parallelism

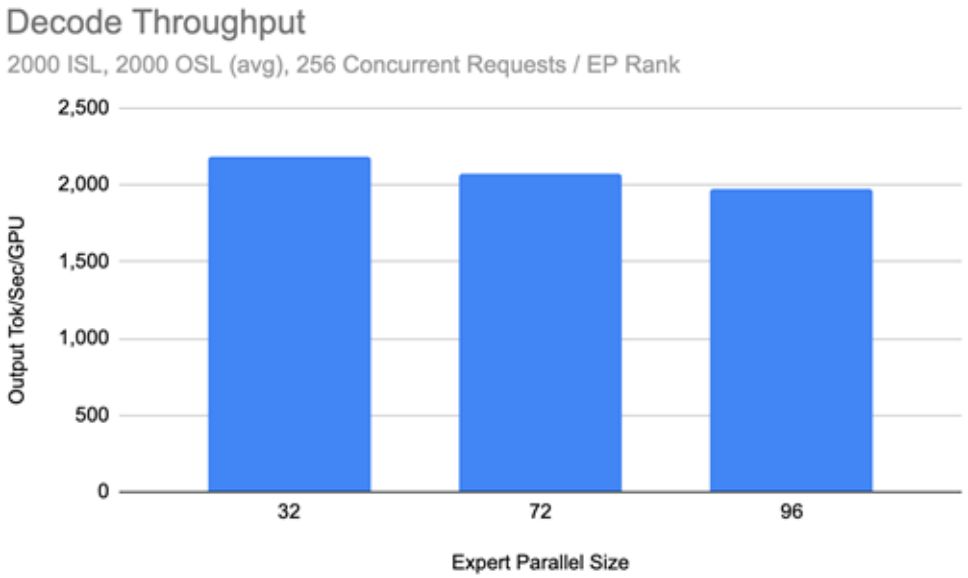
llm-d's K8s-native design integrates EP implementation directly with the broader system, including DP-aware load balancing



Wide Expert Parallelism

Reached **2.2k tokens/s per H200 GPU** for DeepSeek-R1 in community benchmarks on H200 clusters





State-of-The-Art performance from deep GPU and scheduler optimizations across the full inference stack

https://llm-d.ai/blog/llm-d-v0.3-expanded-hardware-faster-perf-and-igw-ga

To Conclude

Key Takeaways

01

KV-Cache Hit Rate Is the #1 Production Metric

A 10× cost gap separates cached from uncached tokens. Standard load balancing destroys this. It's not a nice-to-have — it's your inference margin.

02

llm-d Delivers Real Wins Out of the Box

Prefix-aware scheduling alone: 57× faster TTFT, 2× throughput. P/D disaggregation, expert parallelism, and KV offloading compound gains further.

03

Semantic KV-Cache Is the Next Frontier

Smart (semantic-aware) instead of just raw tokens .

Join the llm-d Community!

llm-d https://llm-d.ai/

Open source. Kubernetes-native.
Your contributions matter.

GitHub

github.com/llm-d/llm-d

github.com/llm-d/llm-d-kv-cache

github.com/llm-d/llm-d-inference-scheduler

Slack (llm-d workspace)

#sig-kv-disaggregation

#sig-inference-scheduler

Slack (vLLM workspace)

#feat-v1-cpu-offloading

#feat-kvcache-offloading

https://llm-d.ai

Join Community Slack
llm-d.ai/slack

Thank You!

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Embedded Linux Conference