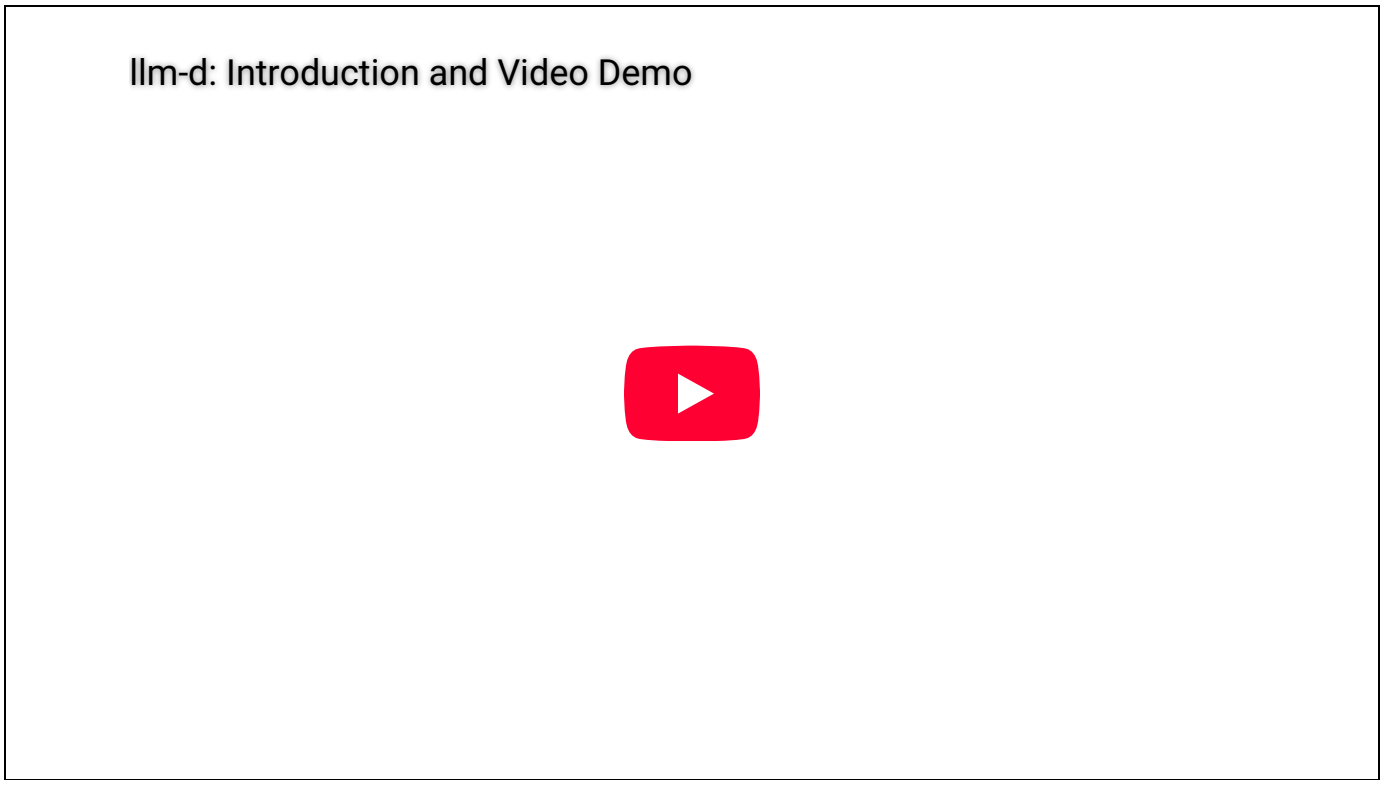
Overview of llm-d architecture

`llm-d` is a Kubernetes-native distributed inference serving stack - a well-lit path for anyone to serve large language models at scale, with the fastest time-to-value and competitive performance per dollar for most models across most hardware accelerators.

With `llm-d`, users can operationalize GenAI deployments with a modular solution that leverages the latest distributed inference optimizations like KV-cache aware routing and disaggregated serving, co-designed and integrated with the Kubernetes operational tooling in Inference Gateway (IGW).

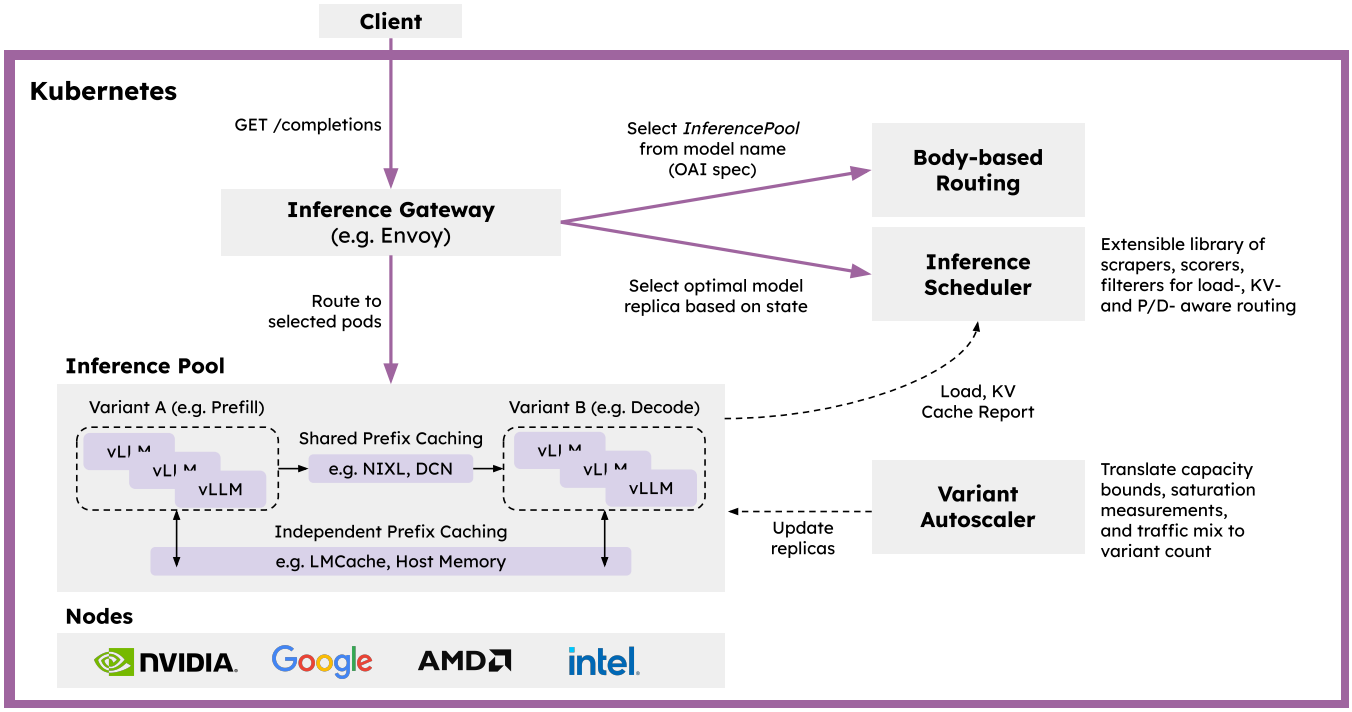
Built by leaders in the Kubernetes and vLLM projects, `llm-d` is a community-driven, Apache-2 licensed project with an open development model.

Video Demonstration



Architecture

`llm-d` adopts a layered architecture on top of industry-standard open technologies: vLLM, Kubernetes, and Inference Gateway.



Key features of `llm-d` include:

- **vLLM-Optimized Inference Scheduler:** `llm-d` builds on IGW's pattern for customizable "smart" load-balancing via the Endpoint Picker Protocol (EPP) to define vLLM-optimized scheduling. Leveraging operational telemetry, the Inference Scheduler implements the filtering and scoring algorithms to make decisions with P/D-, KV-cache-, SLA-, and load-awareness. Advanced teams can implement their own scorers to further customize, while benefiting from other features in IGW, like flow control and latency-aware balancing. See our Northstar design

- **Disaggregated Serving with vLLM:** `llm-d` leverages vLLM's support for disaggregated serving to run prefill and decode on independent instances, using high-performance transport libraries like NVIDIA's NIXL. In `llm-d`, we plan to support latency-optimized implementation using fast interconnects (IB, RDMA, ICI) and throughput optimized implementation using data-center networking. See our Northstar design

- **Disaggregated Prefix Caching with vLLM:** `llm-d` uses vLLM's KVConnector API to provide a pluggable cache for previous calculations, including offloading KVs to host, remote storage, and systems like LMCache. We plan to support two KV caching schemes. See our Northstar design
    - *Independent* (north-south) caching with offloading to local memory and disk, providing a zero operational cost mechanism for offloading.
    - *Shared* (east-west) caching with KV transfer between instances and shared storage with global indexing, providing potential for higher performance at the cost of a more operationally complex system.
- **Variant Autoscaling over Hardware, Workload, and Traffic** (🚧): We plan to implement a traffic- and hardware-aware autoscaler that (a) measures the capacity of each model server instance, (b) derive a load function that takes into account different request shapes and QoS, and (c) assesses recent traffic mix (QPS, QoS, and shapes) Using the recent traffic mix to calculate the optimal mix of instances to handle prefill, decode, and latency-tolerant requests, enabling use of HPA for SLO-level efficiency. See our Northstar design

For more, see the project proposal

Getting Started

`llm-d` can be installed as a full solution, customizing enabled features, or through its individual components for experimentation.

Deploying as a solution

llm-d's deployer can be used to install it as a solution using a single Helm chart on Kubernetes.

Tip See the guided experience with our quickstart.

Experimenting and developing with llm-d

llm-d repo is a metaproject with subcomponents that can be cloned individually.

To clone all the components:

```
git clone --recurse-submodules https://github.com/llm-d/llm-d.git
```

Tip As a customization example, see here a template for adding a scheduler scorer.

Releases

Visit our GitHub Releases page and review the release notes to stay updated with the latest releases.

Contribute

Guidelines

- See our project overview for more details on our development process and governance.

Connect

- We use Slack to discuss development across organizations. Please join via this inviter link and access the workspace here.
- We host a weekly standup for contributors on Wednesdays at 1230pm ET. Please join by adding the shared calendar
- We use Google Groups to share architecture diagrams and other content. Please join: Google Group

License

This project is licensed under Apache License 2.0. See the LICENSE file for details.