KCD HANGZHOU
GROWING CLOUD NATIVE TOGETHER

CHINA OpenInfra Days

LLM-D：面向云原生的大模型部署与实践

船长(samzong)

DaoCloud AI Infra/LLMOps 产品负责人

2025/11/11

船长(samzong)

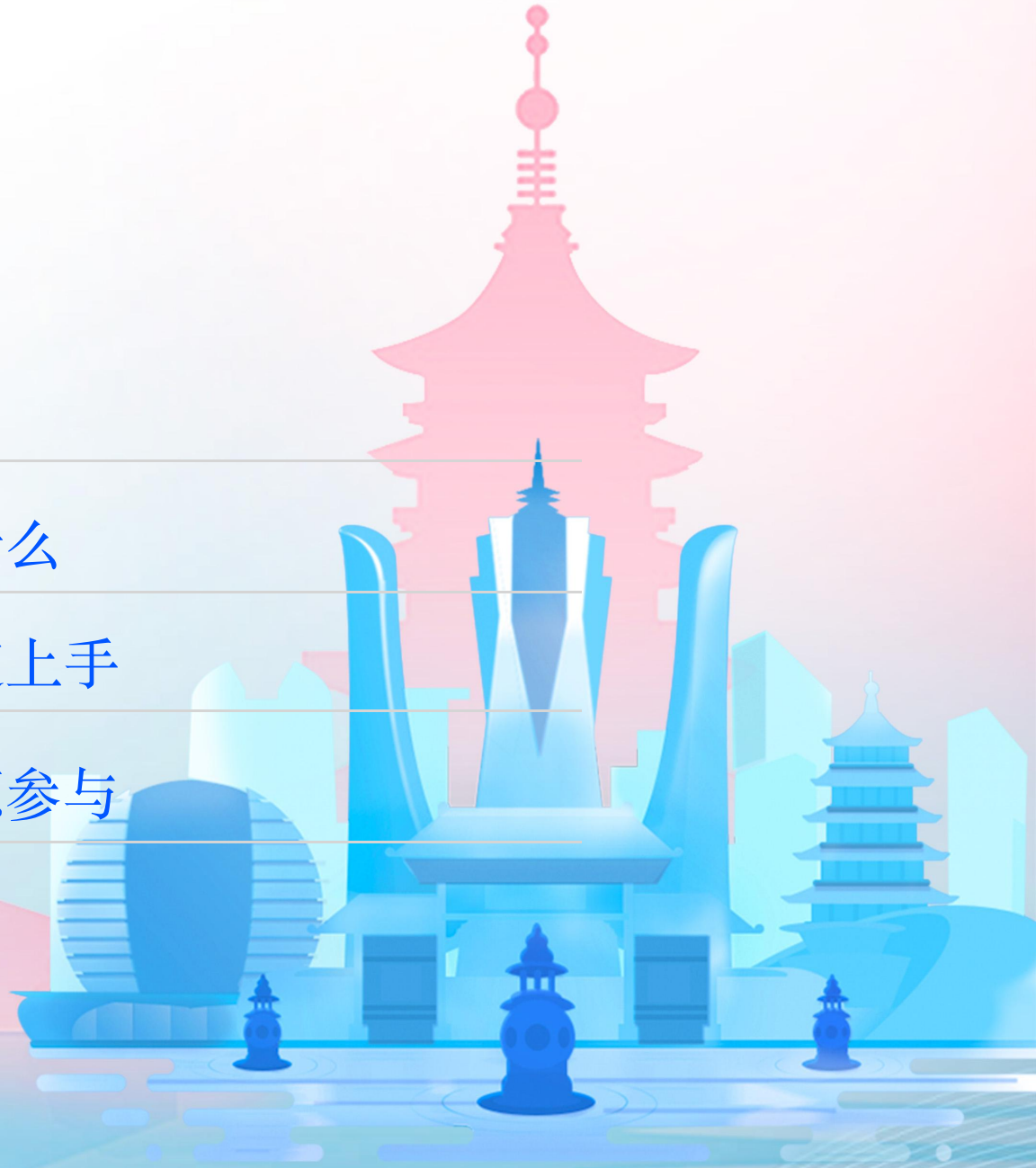
DaoCloud AI Infra/LLMOps 产品负责人

专注 AI Infra 和 LLMOps 的开源开发者，活跃于 Kubernetes、Kueue、Karmada、Istio、HAMi、llm-d 等项目。关注 GPU 调度、模型推理性能、分布式系统工程实践。喜欢做工具、写代码、造轮子，用开源方式探索更高效的 AI 基础设施。

KCD HANGZHOU

CHINA OpenInfra Days

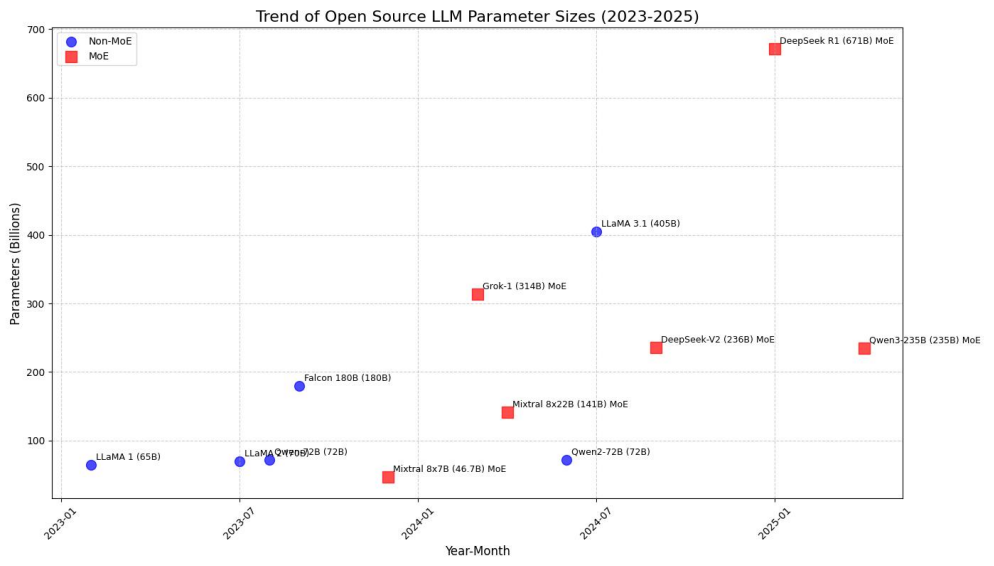
CONTENT
目录

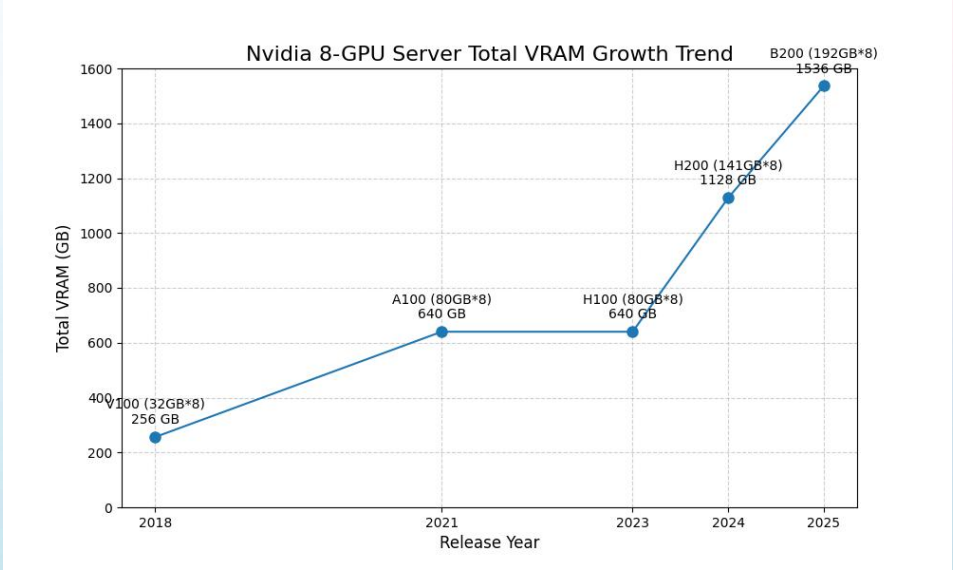


大模型和模型推理的发展

大模型推理已成为核心算力消耗环节。**模型规模（参数量、KV Cache 体积）持续增长，GPU 内存成为瓶颈。**

推理服务需同时满足**高 QPS 与低延迟**，但传统单体推理架构在**资源利用率和灵活性**上严重受限。





大模型和模型推理的发展

问题	现象	结果
资源耦合	Prefill 与 Decode 固定绑定同一 GPU	GPU 内存利用率低，易出现“Prefill 满载 / Decode 空闲”
KV Cache 传输低效	全量复制或同步延迟大	显著影响吞吐和延迟
缺乏统一调度	Prefill / Decode 由同一服务调度	无法根据请求动态调度 GPU 资源
缺乏异构适配	不同模型结构（Transformer、SSM、MLA）差异大	推理引擎难以通用，影响整体效率
系统封闭	vLLM、TGI 等采用紧耦合架构	难以跨节点、跨集群伸缩

单体式推理架构已无法支撑多模型、多租户、高并发的生产级负载。

为什么需要分布式推理（分体式）

将 Prefill 与 Decode 解耦为独立阶段。通过网络化 KV Cache 传输连接两者。实现独立扩缩容与动态调度。通过 Inference Gateway (IGW)。

需求	目的
Prefill / Decode 独立扩缩容，GPU 利用率最大化	提高资源利用率
支持非阻塞 KV Cache 流式传输（Push / Pull 模型）	降低延迟
天然支持多节点、多模型并行	提高扩展性
可嵌入 vLLM、TGI、SGLang 等引擎生态	推理引擎

vllm 部署的一些局限性

vLLM 的设计是高效服务引擎，其关注点主要是**单机或单节点多GPU环境**“优化 KV 缓存”、“连续批理”“PagedAttention”。

当我们把它用于分布式 / 分体式推理时，会遇**Prefill／Decode 拆分和调度、路由多节点、KV 缓存跨节点**。

```
vllm serve Qwen/Qwen3-Coder-480B-A35B-Instruct \
  --max-model-len 32000 \
  --enable-expert-parallel \
  --tensor-parallel-size 8 \
  --enable-auto-tool-choice \
  --tool-call-parser qwen3_coder
```

8xH200 (or H20) GPUs (141GB × 8)

kubernetes 一个好的选择

大规模 LLM 推理服务需要异构 GPU、缓存路由、请求形态识别与资源弹性扩缩，可监控扩缩与故障自愈，而这些正是 K8s 成熟具备的分布式调度系统的能力。

同时 Kubernetes 内也拥有大量利用开源项目来支撑。

- Kueue
- HAMi
- GAIE

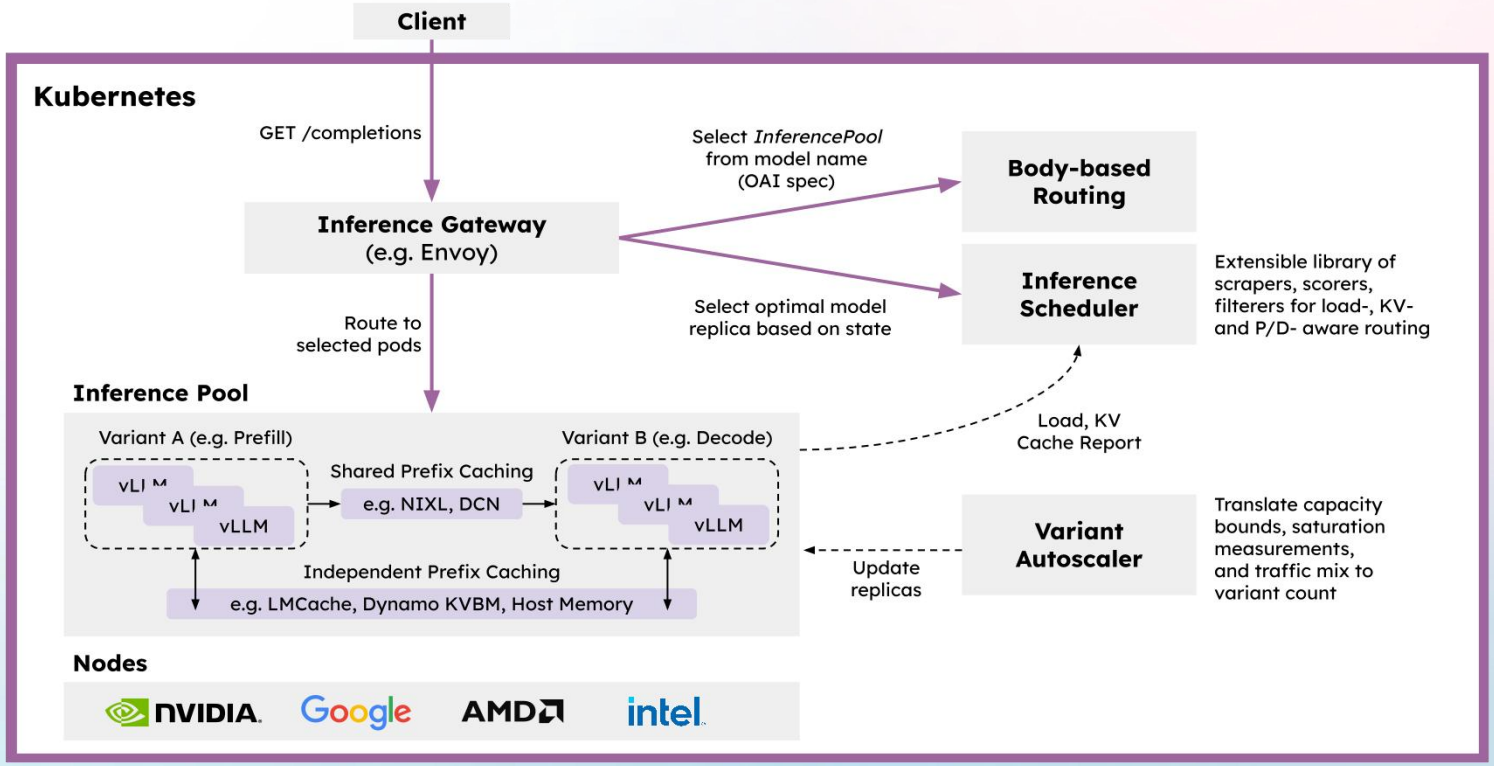
NVIDIA Dynamo

vllm-project/product-stack

LLM-D

llm-d 云原生分布式推理框架

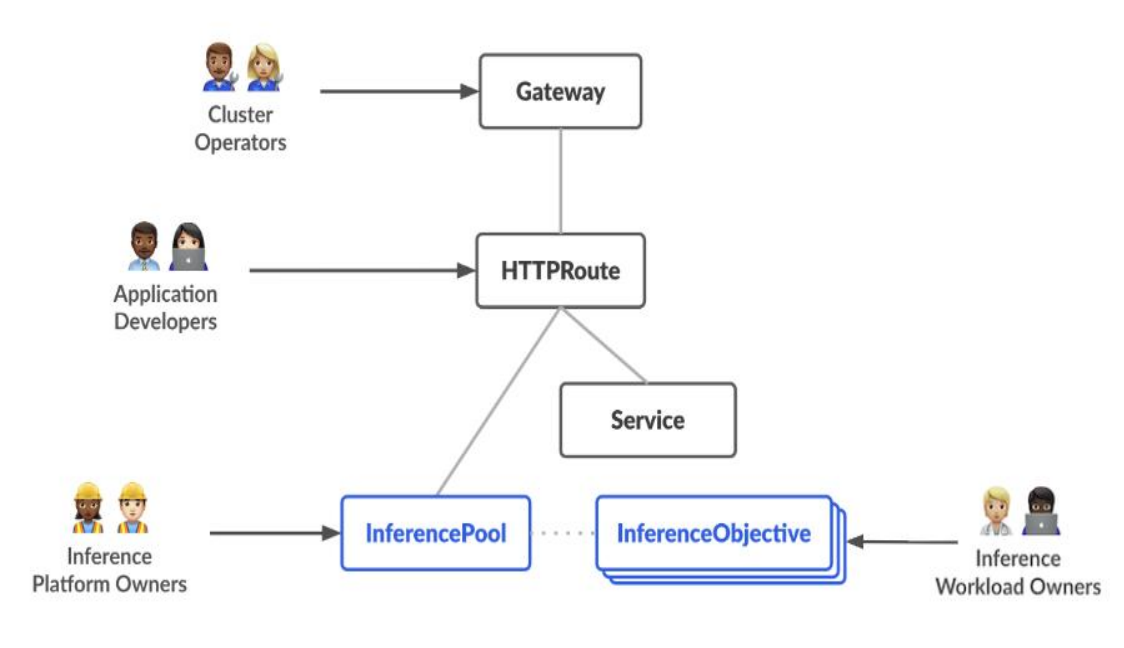
- 开源、Kubernetes 原生、兼容多种加速器的推理框架
- 明确且易用的推荐实践路径
- 可复现的基准测试
- 具备 LLM 感知能力的推理网关和自动扩缩容
- 遵循开放标准，具备高度可定制和可组合性



Inference Gateway Extension（IGW）

IGW 是 llm-d 扩展 Gateway API 的组件，把 request-level routing 和 Prefill/Decode 调度能力拉到 API 层。它提供：

用来感知请求体、前缀缓存与分体式需求，而不是只看 HTTP header。



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

ModelService

Helm Chart 实现 大模型与 LoRA 的声明式部署与管理，
简化推理服务的生命周期，统一 llm-d 部署方式：

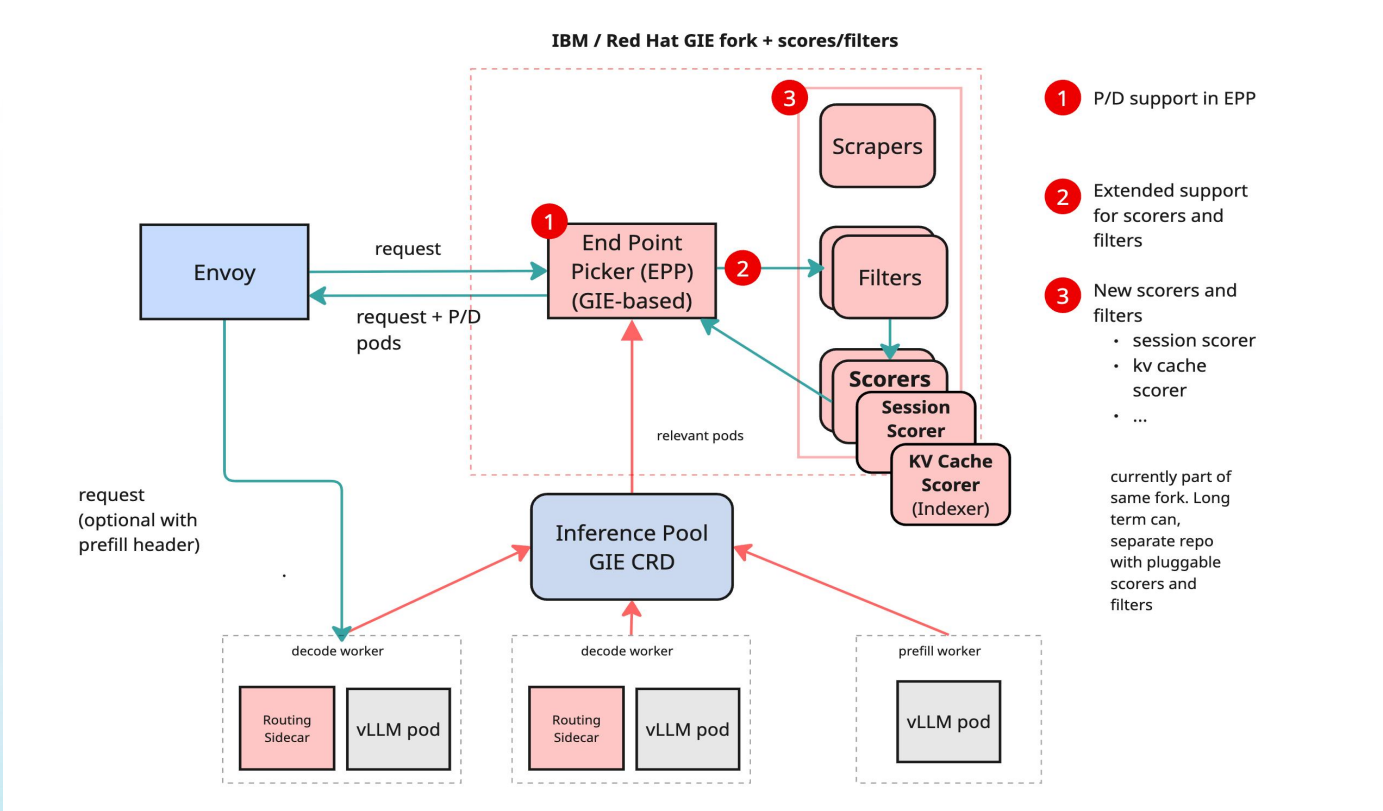
- 快速、可复现的推理部署
- 支撑分布式性能调优与实验
- 为 llm-d 提供标准化 Inference 入口

```
(.venv) ~/git/llm-d-project (main ×) helm repo add llm-d-modelservice https://llm-d-incubation.github.io/llm-d-modelservice/

"llm-d-modelservice" has been added to your repositories
(.venv) ~/git/llm-d-project (main ×) helm repo update llm-d-modelservice
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "llm-d-modelservice" chart repository
Update Complete. ⎈Happy Helming!⎈
(.venv) ~/git/llm-d-project (main ×) helm search repo llm-d-modelservice
NAME                                    CHART VERSION   APP VERSION     DESCRIPTION
llm-d-modelservice/llm-d-modelservice   v0.3.3          v0.2.0          A Helm chart for ModelService in llm-d
(.venv) ~/git/llm-d-project (main ×)
```

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

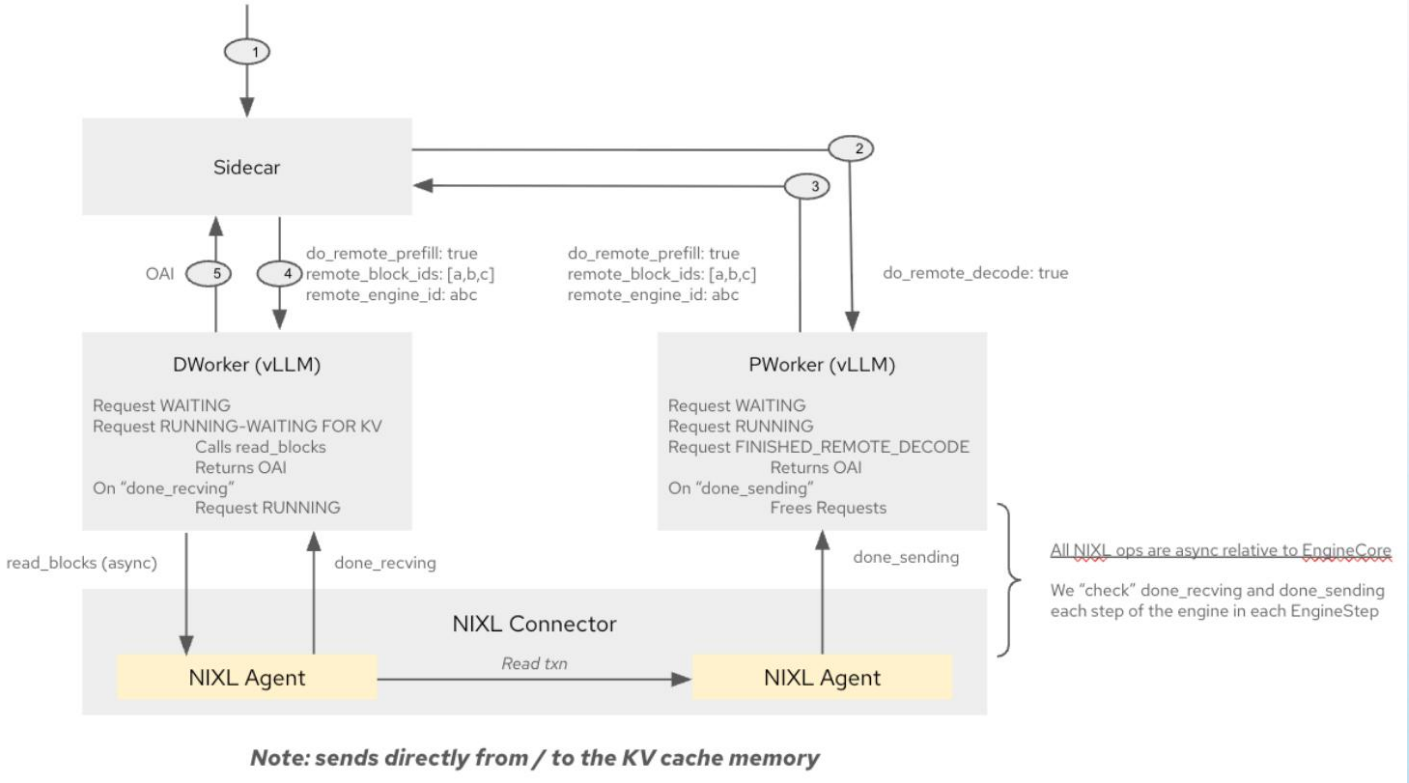
P/D Inference Scheduler

- EPP (extends the GAIE)
  - https://github.com/llm-d/llm-d-inference-scheduler/blob/main/docs/architecture.md
- 负责协调 Prefill 与 Decode 流程
- 基于推理 Pod 的指标，采用前缀缓存感知与负载感知的智能路由。
- 支持可配置阈值：当 prompt 长度超过预设限制时，自动触发远端 Prefill。
- 通过 decode-sidecar 代理实现请求的编排与路由。



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

Routing Sidecar



Routing Sidecar 是与 Decode Pod 同部署的轻量代理，它在 Prefill/Decode 整个生命周期中负责：

1. 接受来自 Inference Gateway Scheduler 的路由决策，维护 decode worker 的 KV chunk metadata。

2. 在需要远端 Prefill 时充当桥梁，发送 `max_tokens=1` 触发 prefill；完成后把 KV transfer 结果同步到 decode 端，并流式返回首个 token。

3. 在非 PD 模式下保持零开销，整体上让调度器感知的流量和实际的 token 生成保持一致。

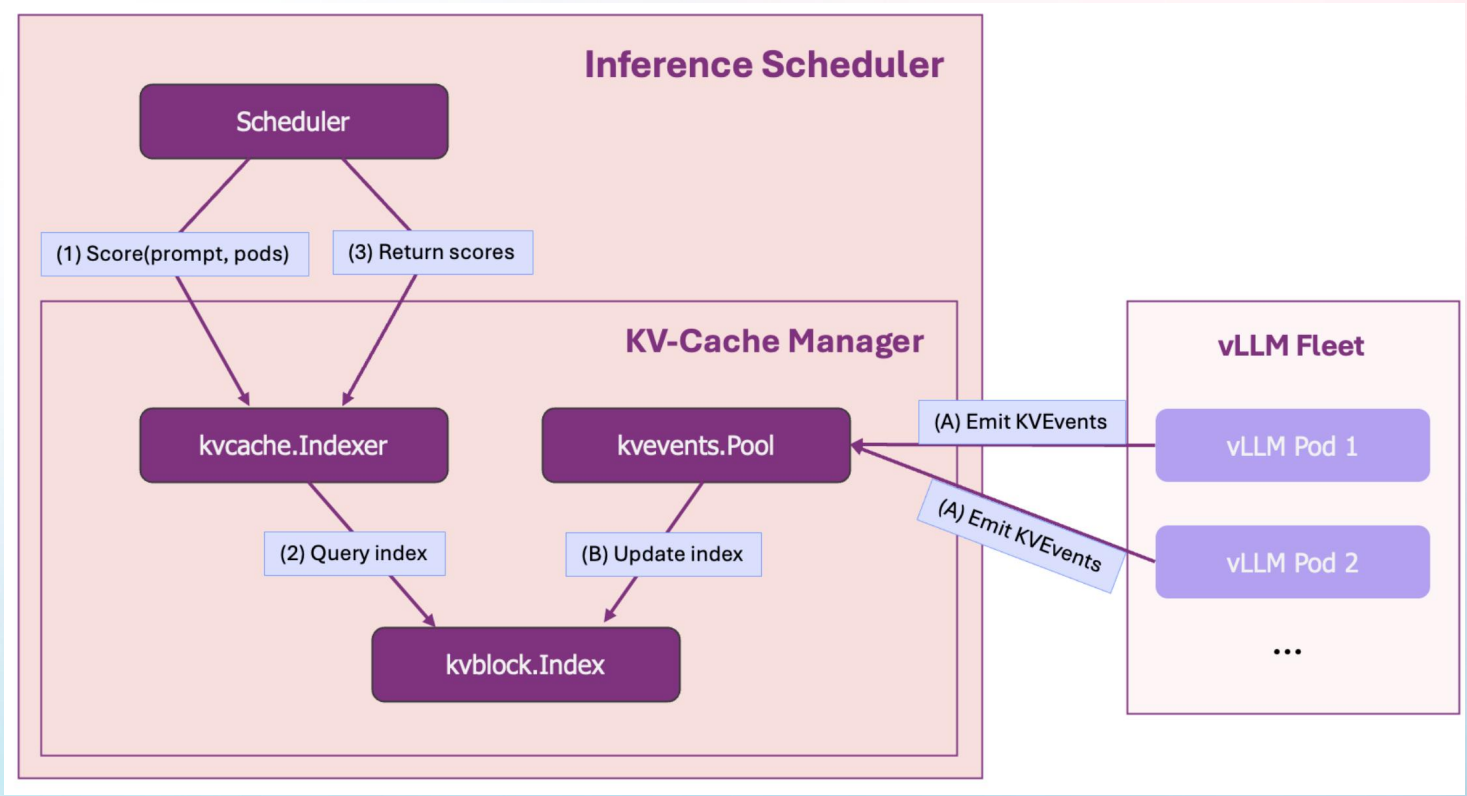
https://github.com/llm-d/llm-d-routing-sidecar

KV-Cache Manager

跨节点 KV 转移通过 NIXL RDMA、LMCache、或 GPU-direct（NVLink/GPUDirect）完成，其核心原则是 **非阻塞、按需加载、预分配**：

- Prefill worker 在生成 KV cache 之后，立刻将 chunk metadata 用 scoring plugin 广播给调度器/sidecar，不再同步等待 decode 端 ack，避免了 blocking push 导致的 compute stranded。

- Decode worker 在收到 KV chunk id 后，采用非阻塞 pull 模式从 Prefill worker 拉取数据，也可以预分配内存（例如 Dynamo 方案）避免重复的 malloc。



部署（llm-d-infra）

llm-d-infra 的设计倾向于组合化：你可以只部署 GIE、InferencePool、ModelService chart（该 chart 于 2025 年 6 月 10 日获准接受）或 LeaderWorkerSet 组件，无需再依赖单一的 llm-d-deployer chart。

这样可以根据团队硬件（例如 H200、TPU、Intel XPU）灵活选配组件，并保持与 llm-d 社区的最新实践同步。

```
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo add llm-d-infra https://llm-d-incubation.github.io/llm-d-infra/
helm install my-llm-d-infra llm-d-infra/llm-d-infra
```

* llm-d-deployer 于 **2025 年 7 月 25 日** 被标记弃用。

版本更新 v0.3

增强对专用硬件后端（如 TPU、XPU）的支持。

建立度量与可观测性（metrics & observability）机制。

Inference Gateway v1.0 GA：作为稳定的智能路由原语，
支持缓存感知、负载 & 前缀感知路由。

llm-d 的贡献参与（仓库和孵化）

llm-d：

核心组织，包含所有走向生产环境关键路径的代码。

- 遵循 API 变更与废弃（Deprecation）流程
- 所有重大变更需提交项目提案（Project Proposal）
- 如后续需要中间层仓库（midstream repos），会在本组织内纳入管理

llm-d-incubation：

实验阶段或尚未完全支持的组件。

- 将 llm-d 内的代码范围限定为最小可用项目（Minimum Viable Project）
- 提供更低门槛、更灵活的场所孵化项目
- 毕业的组件会**迁移至 llm-d 组织**

```
(.venv) ~/git/llm-d-project (main ×) tree -L 1 llm-d llm-d-incubation
llm-d
├── llm-d
├── llm-d-benchmark
├── llm-d-deployer
├── llm-d-inference-scheduler
├── llm-d-inference-sim
├── llm-d-kv-cache-manager
├── llm-d-model-service
├── llm-d-pd-utils
├── llm-d-routing-sidecar
└── llm-d.github.io
llm-d-incubation
├── hermes
├── ig-wva
├── llm-d-ci
├── llm-d-fast-model-actuation
├── llm-d-infra
├── llm-d-modelservice
└── workload-variant-autoscaler
```

Hermes

Hermes 是一个为 LLM-D 推理工作负载 设计的集群配置扫描与自测生成工具，用于分析 RDMA 能力 GPU 集群 的网络与拓扑结构。它支持自动检测和测试高性能互连（如 InfiniBand 与 RoCE），可在多种环境中运行，包括 CoreWeave、GKE、OpenShift 以及通用的 Kubernetes 集群。

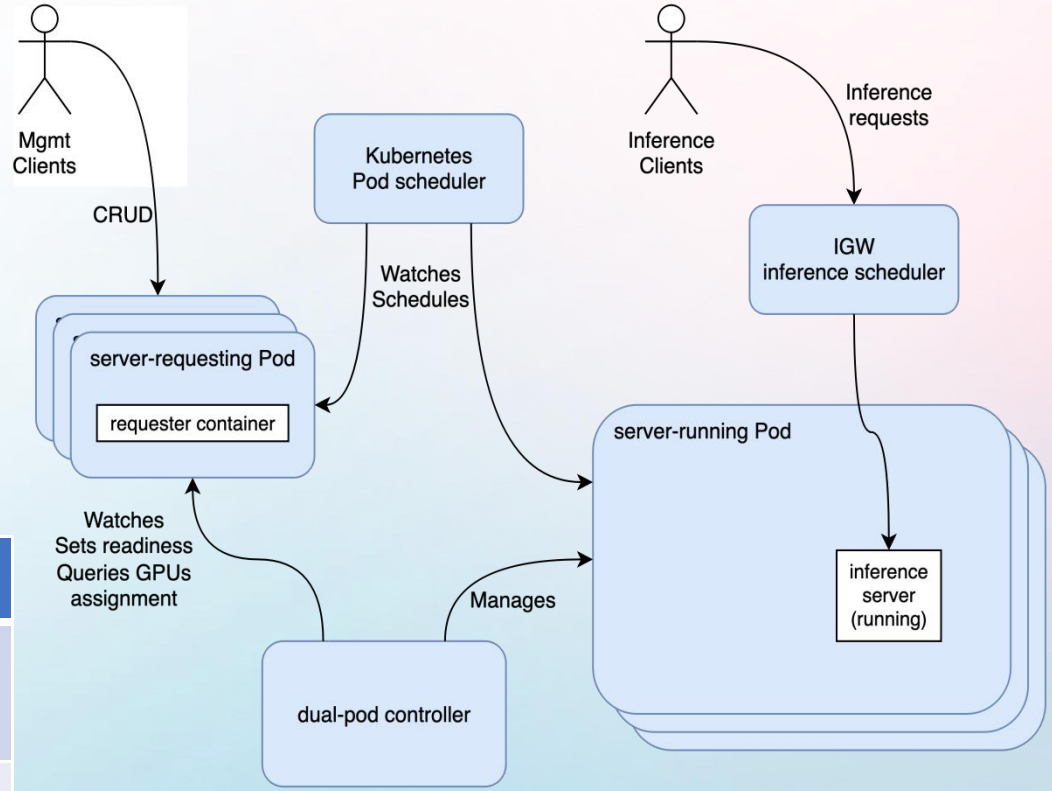
```
# macOS (Apple Silicon)
curl -LO https://github.com/llm-d-incubation/hermes/releases/download/v0.1.0/hermes-aarch64-apple-darwin.tar.gz
tar xzf hermes-aarch64-apple-darwin.tar.gz
sudo mv hermes /usr/local/bin/

# macOS (Intel)
curl -LO https://github.com/llm-d-incubation/hermes/releases/download/v0.1.0/hermes-x86_64-apple-darwin.tar.gz
tar xzf hermes-x86_64-apple-darwin.tar.gz
sudo mv hermes /usr/local/bin/

# Linux (x86_64)
curl -LO https://github.com/llm-d-incubation/hermes/releases/download/v0.1.0/hermes-x86_64-unknown-linux-gnu.tar.gz
tar xzf hermes-x86_64-unknown-linux-gnu.tar.gz
sudo mv hermes /usr/local/bin/
```

llm-d-fast-model-actuation

原生 Pod 不支持动态 GPU 管理与命令变更。vLLM 推理需要 sleep/wake 与 model swapping 的灵活机制

Dual-Pod 将“用户视角的 Pod”与“实际运行的 Pod”解耦。动态加载/卸载模型、模型热切换、GPU 动态绑定。

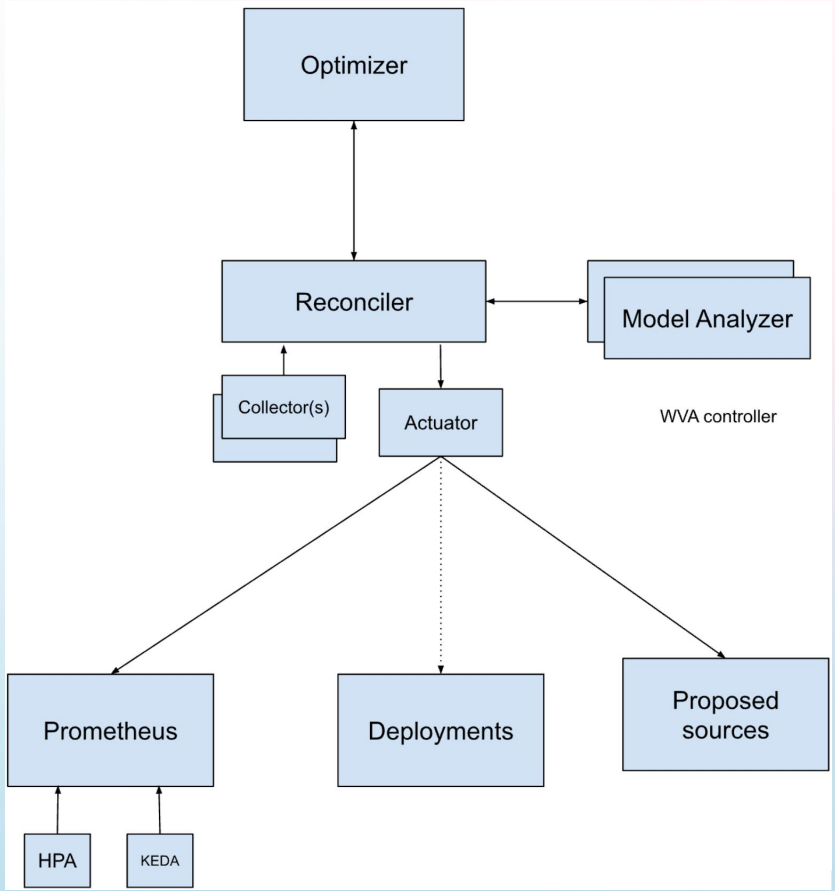
概念	角色	描述
Server-Requesting Pod	用户创建	描述期望的推理服务（stub pod）
Server-Running Pod	控制器创建	实际运行模型推理进程（vLLM 实例）



Workload-Variant-Autoscaler

GPU 感知型自动伸缩控制器，根据推理负载和服务等级（SLO）实现智能算力分配与成本优化。

- 智能自动伸缩（Intelligent Autoscaling）
- 自动分析推理请求速率、GPU 利用率和模型性能，动态调整副本数量与 GPU 类型。
- 成本最优化（Cost Optimization）
- 在满足延迟和吞吐目标的前提下，最小化 GPU 成本与资源浪费。
- 性能建模（Performance Modeling）
- 基于排队论（M/M/1/k, M/G/1）预测模型的延迟与吞吐表现，从而优化分配决策。
- 多模型支持（Multi-Model Support）
- 支持多模型、多优先级、多服务等级（如 Premium / Standard）的差异化资源调度。



```
~/git/llm-d-project/llm-d/llm-d (main ×) ls
CODE_OF_CONDUCT.md   docs                 LICENSE             patches             sccache.config.toml  SECURITY.md
CONTRIBUTING.md      guides               Makefile            PR_SIGNOFF.md       scripts              SIGS.md
DCO                  helpers              ONBOARDING.md       PROJECT.md          SECURITY_CONTACTS.md THREAT-MODEL.md
docker               hooks                OWNERS              README.md           SECURITY-INSIGHTS.md
~/git/llm-d-project/llm-d/llm-d (main ×) vim helpers/kind-testing/kind-config.yaml
~/git/llm-d-project/llm-d/llm-d (main ×) gst
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        scripts/setup-llmd-sim.sh

nothing added to commit but untracked files present (use "git add" to track)
~/git/llm-d-project/llm-d/llm-d (main ×) kind get clusters
```

```
~/git/samzong/samzong (main ✓)
```

```
~/git/llm-d-project/llm-d/llm-d (main ×)
```

```
Last login: Thu Nov 13 13:28:12 on ttys023
~/git/llm-d-project/llm-d/llm-d (main ✓)
```

KCD HANGZHOU

CHINA OpenInfra Days

Thanks, Join us.

https://llm-d.ai

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