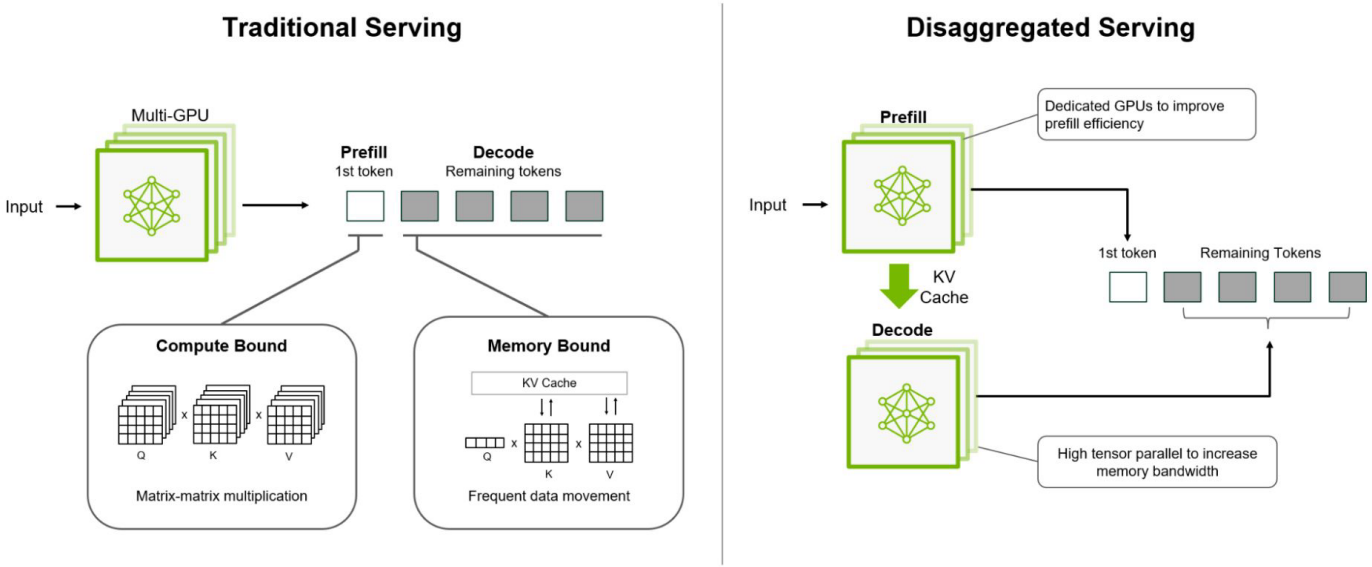
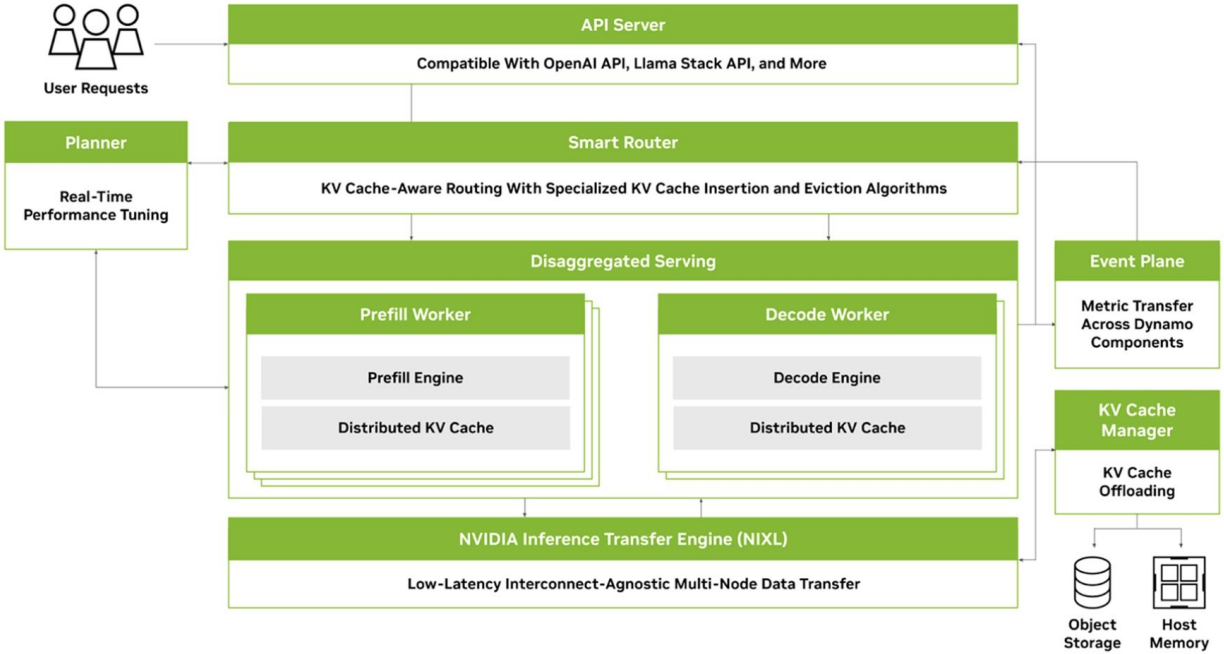
Beyond the Theory: What Actually Breaks When You Scale Your Disaggregated Pytorch Models

Ekin Karabulut & Ron Kahn
NVIDIA

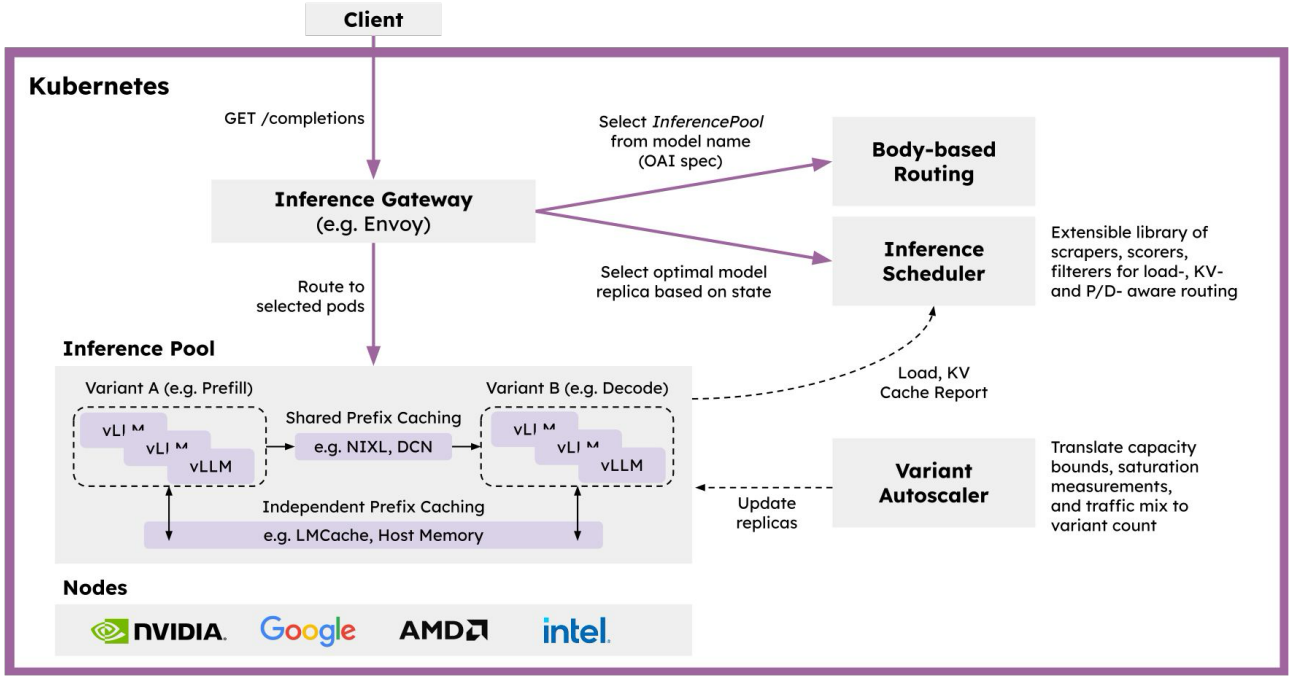
Why We Split Prefill and Decode Apart

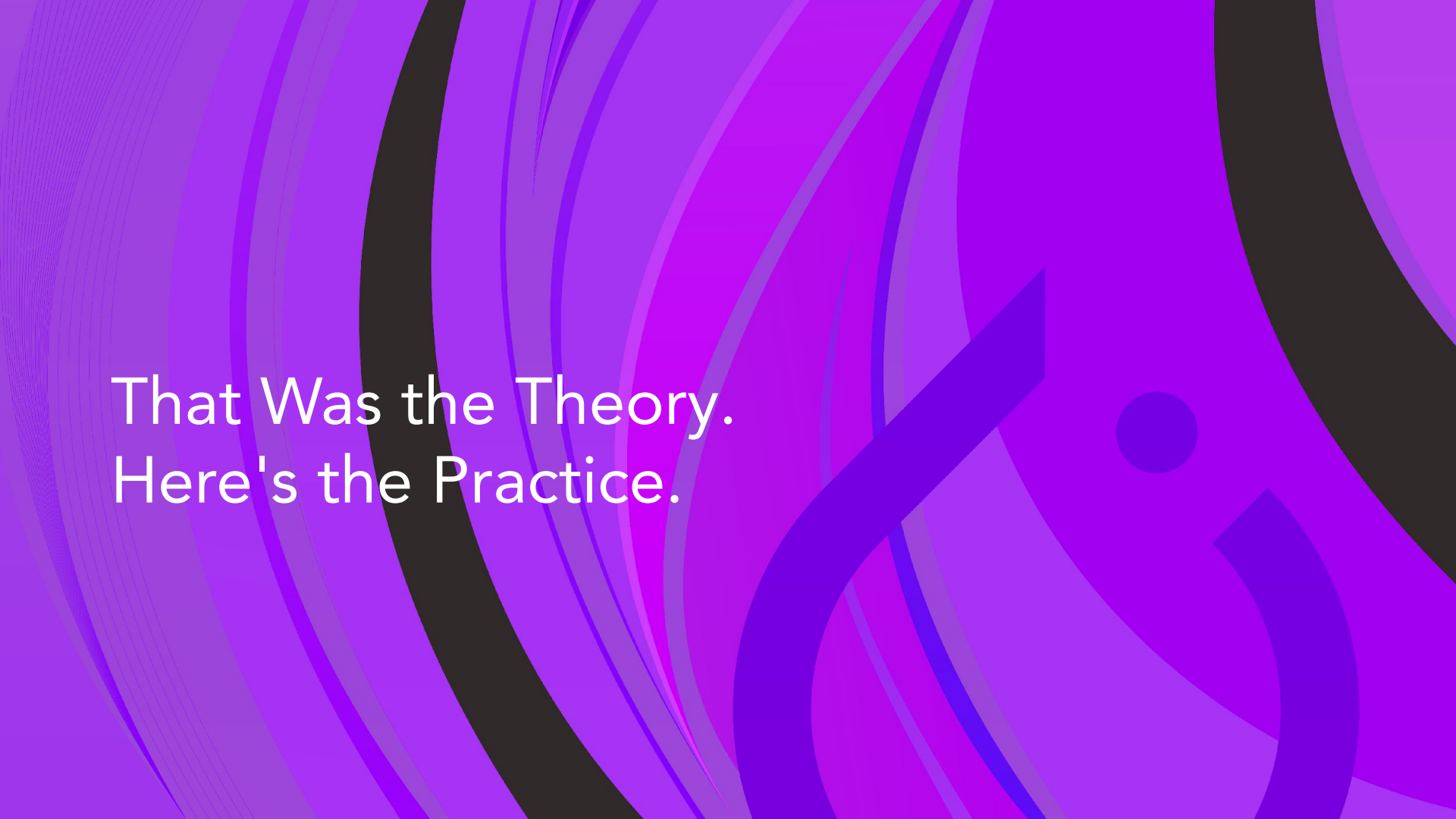


What Dynamo Looks Like Under the Hood



What llm-d Looks Like Under the Hood

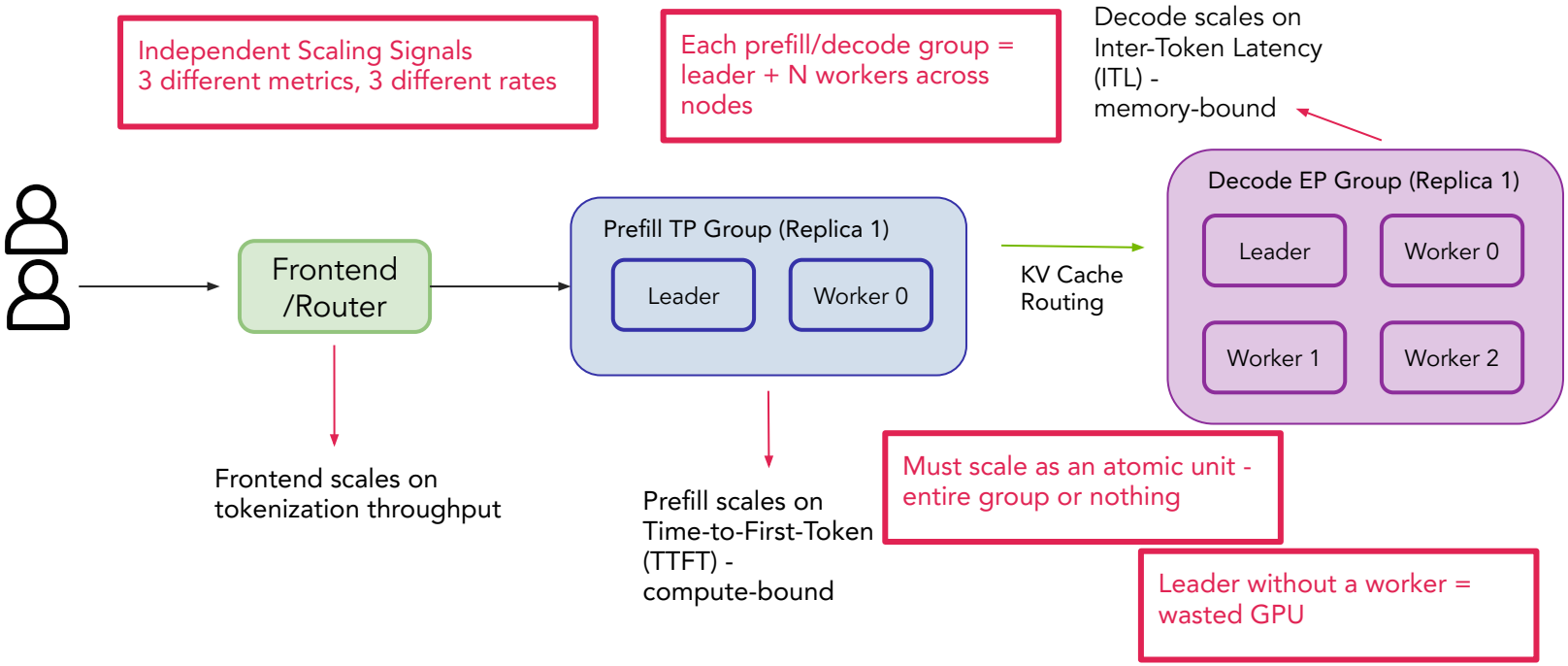



That Was the Theory.
Here's the Practice.

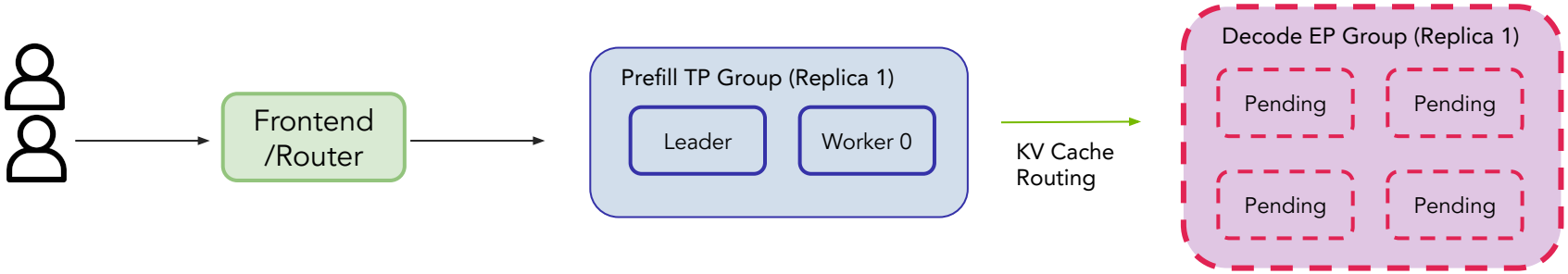
The Production Checklist Nobody Warned You About

1. Scaling: The unit of scaling isn't a pod, it's a coordinated group
2. Scheduling: Partial allocation leads to GPU deadlocks
3. Placement: The scheduler doesn't know your components talk to each other
4. Rolling Updates: v1 and v2 components can't safely coexist
5. Startup Ordering: Gang scheduling doesn't mean gang initialization

What You Need to Know Before We Break Things



The Deadlock Worth $$$$/hr



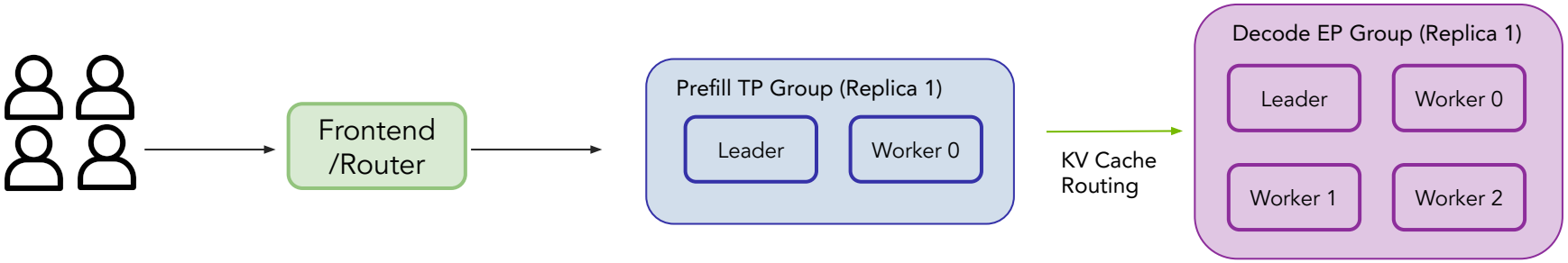
Not enough GPUs available.

Decode Group cannot schedule

We need gang-scheduling across components

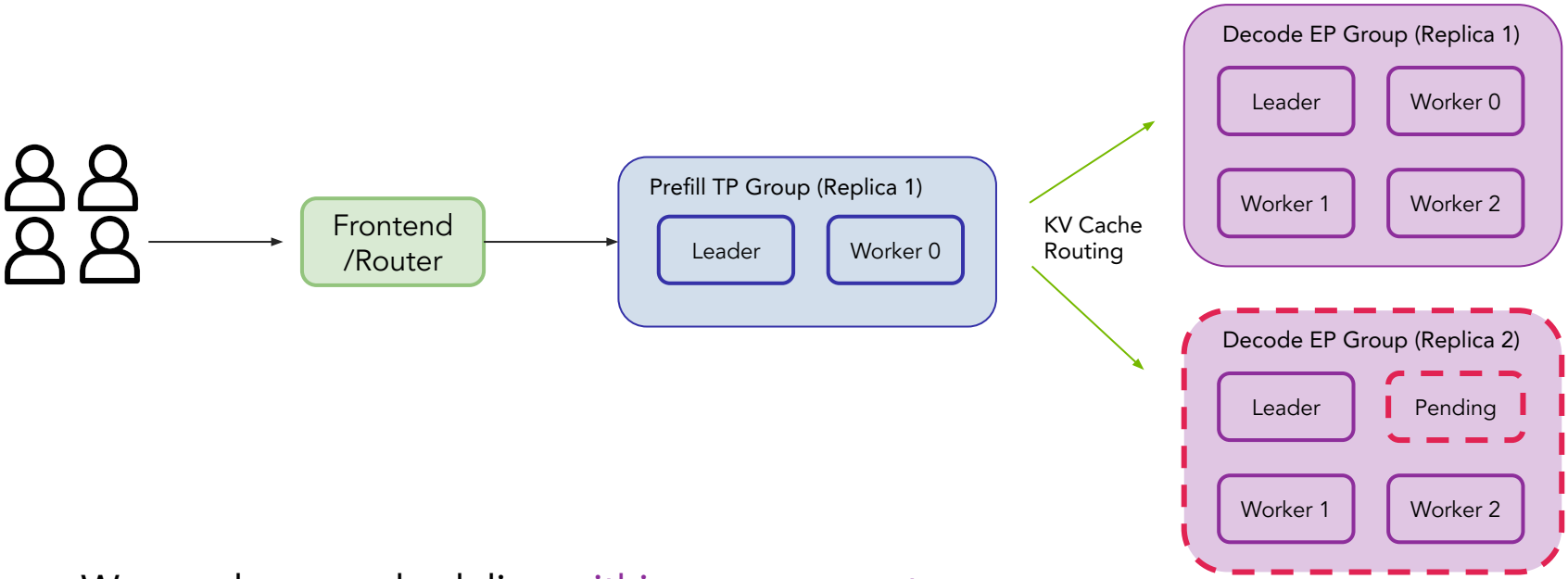
Cool Demo. Now Add Real Traffic

Inter token latency increased



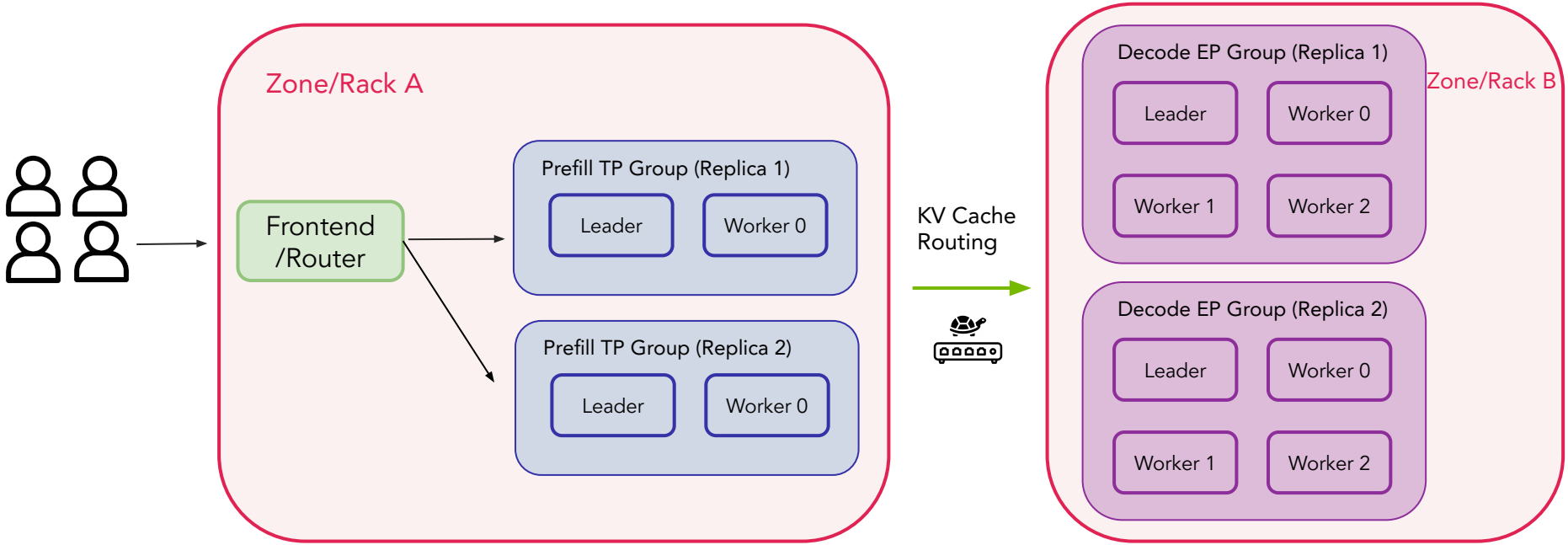
Scale up the decode group!

Cool Demo. Now Add Real Traffic



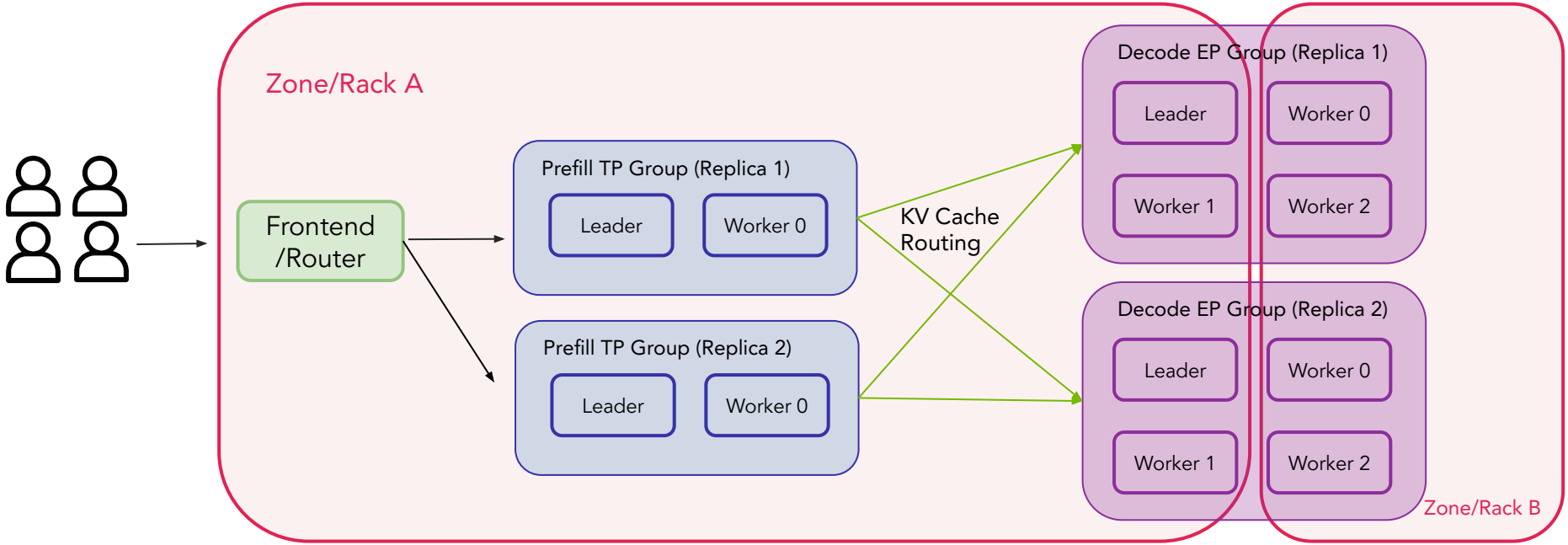
We need gang-scheduling within a component

The Scheduler Put Your Groups in Different ZIP Codes



We need Topology Awareness

The Scheduler Put Your Groups in Different ZIP Codes 2

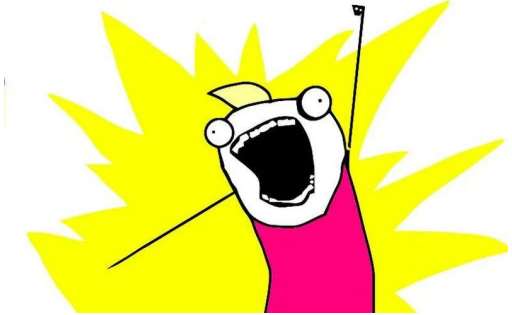


We need Topology Awareness

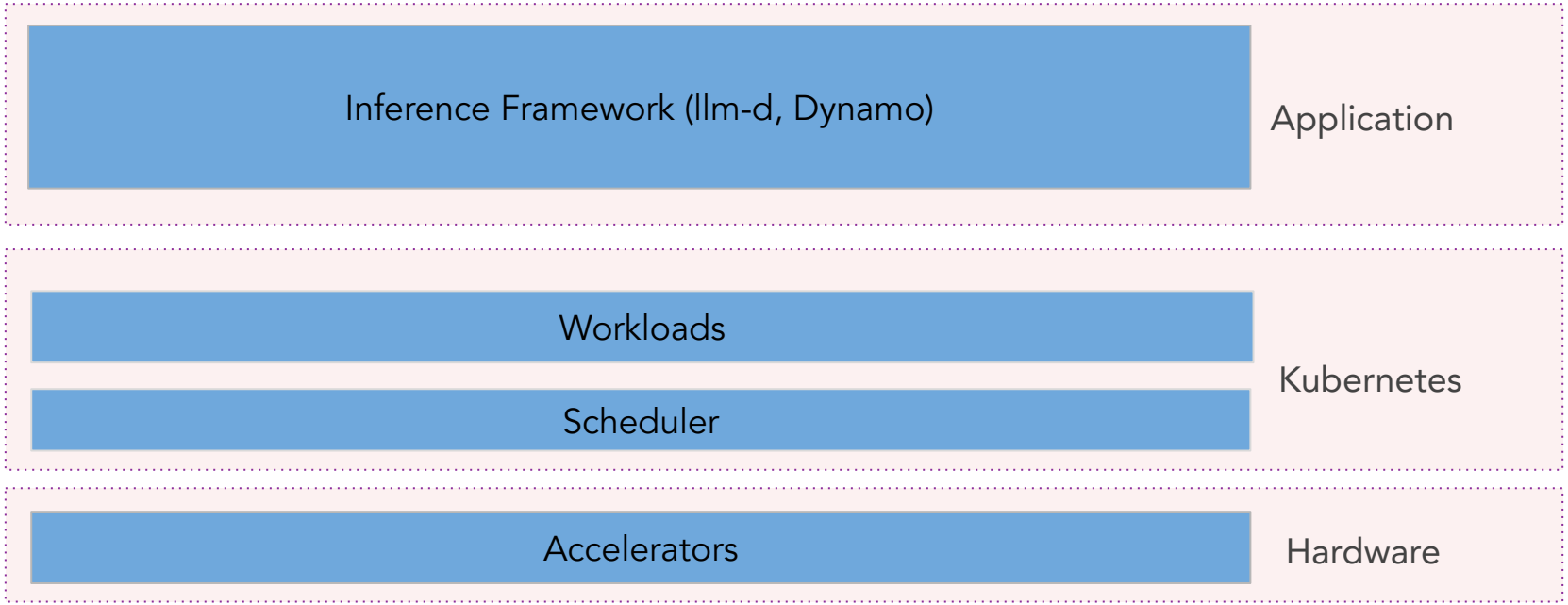
Slow Decode Group! Workers are in different planets

What we want

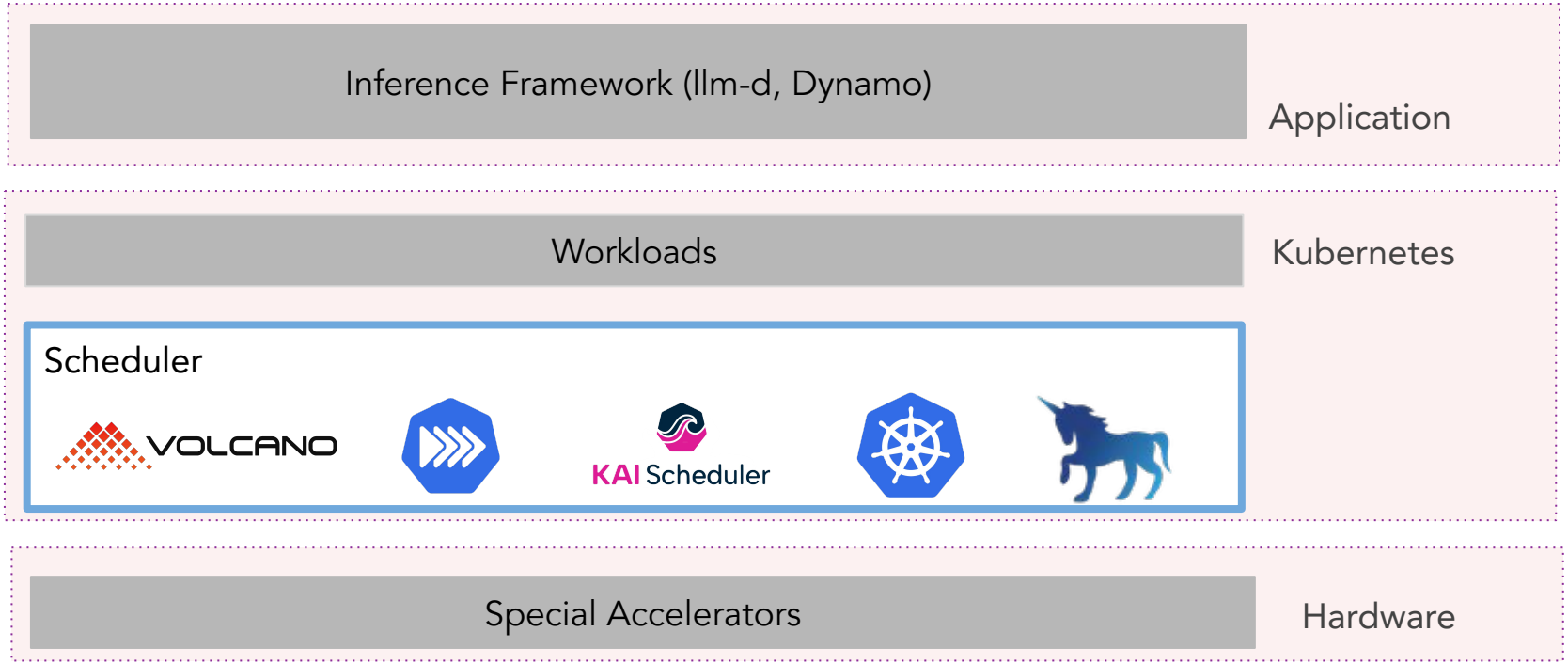
- Gang scheduling across components
- Gang scheduling within components
- Topology Aware Placement
- Multi-level autoscaling
- Smart way of rolling updates



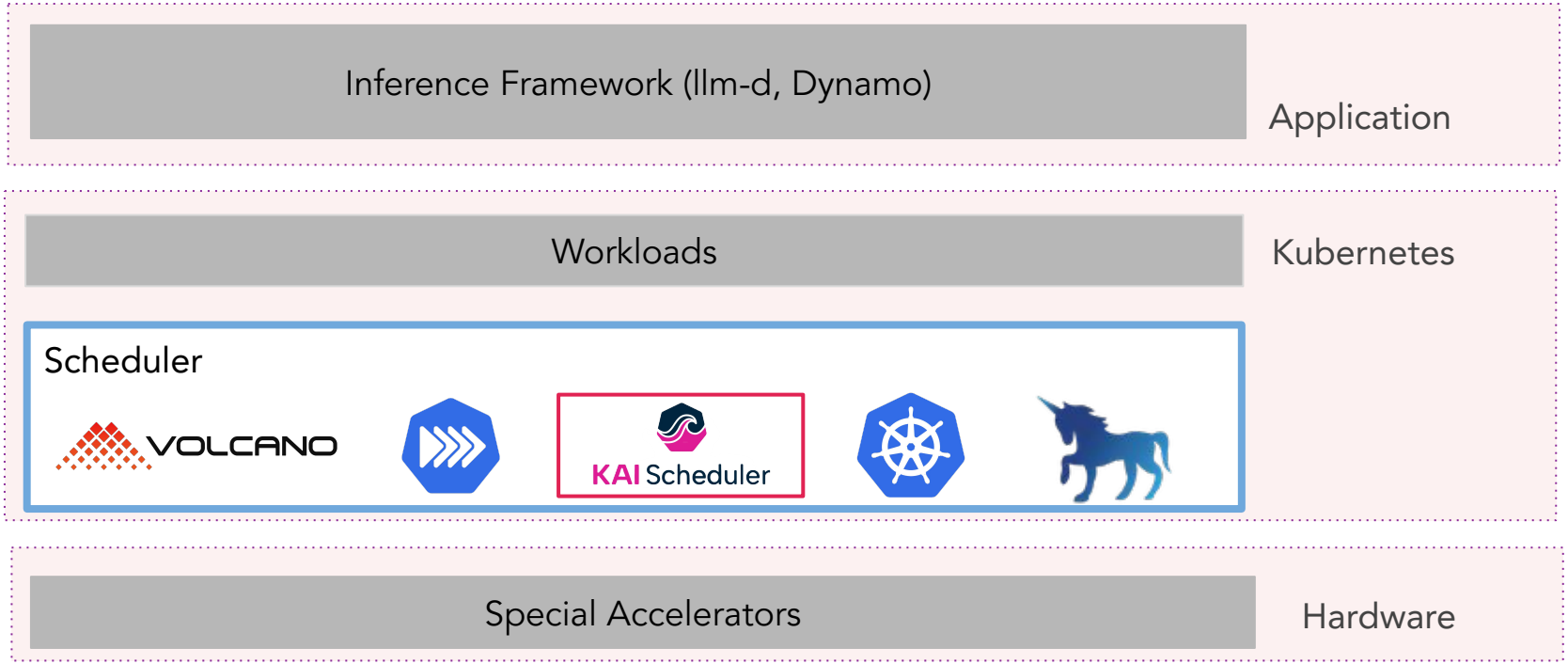
Disaggregated Inference Stack in Production



Scheduler Layer



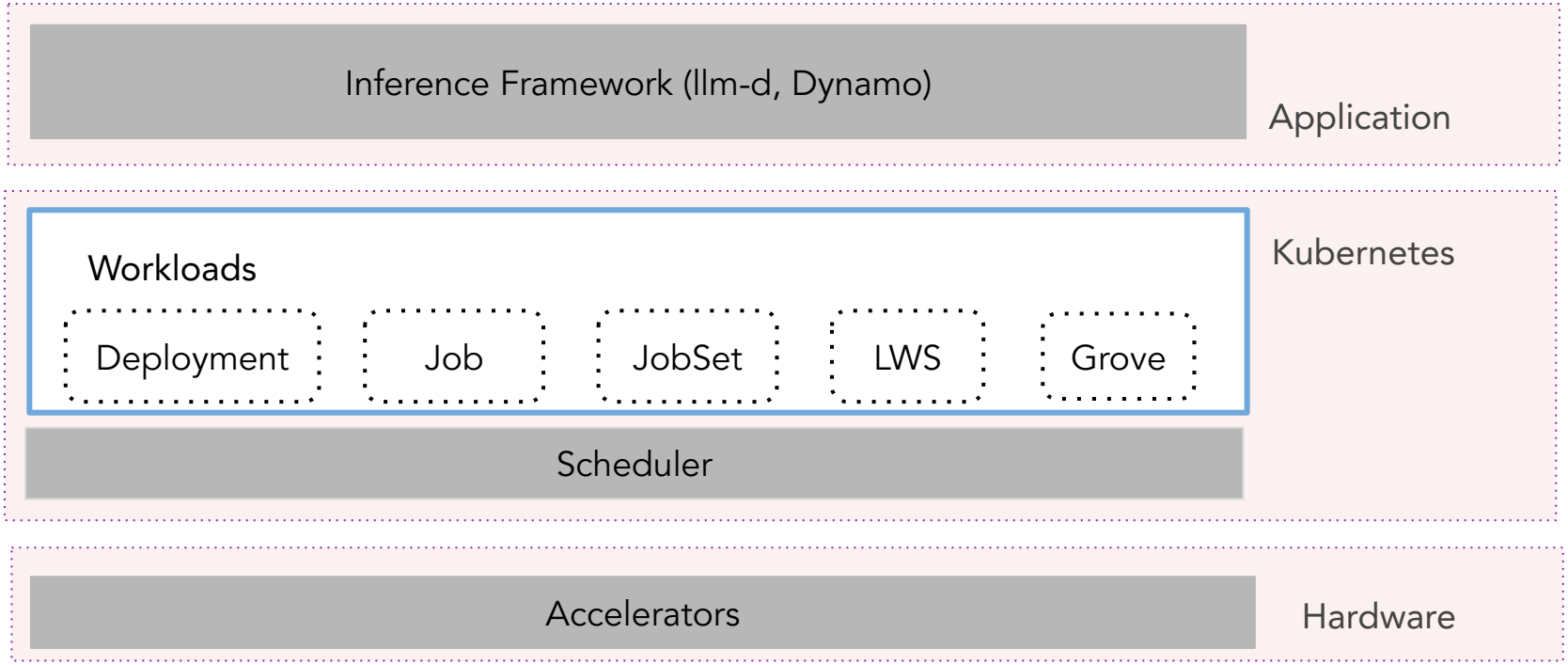
Scheduler Layer



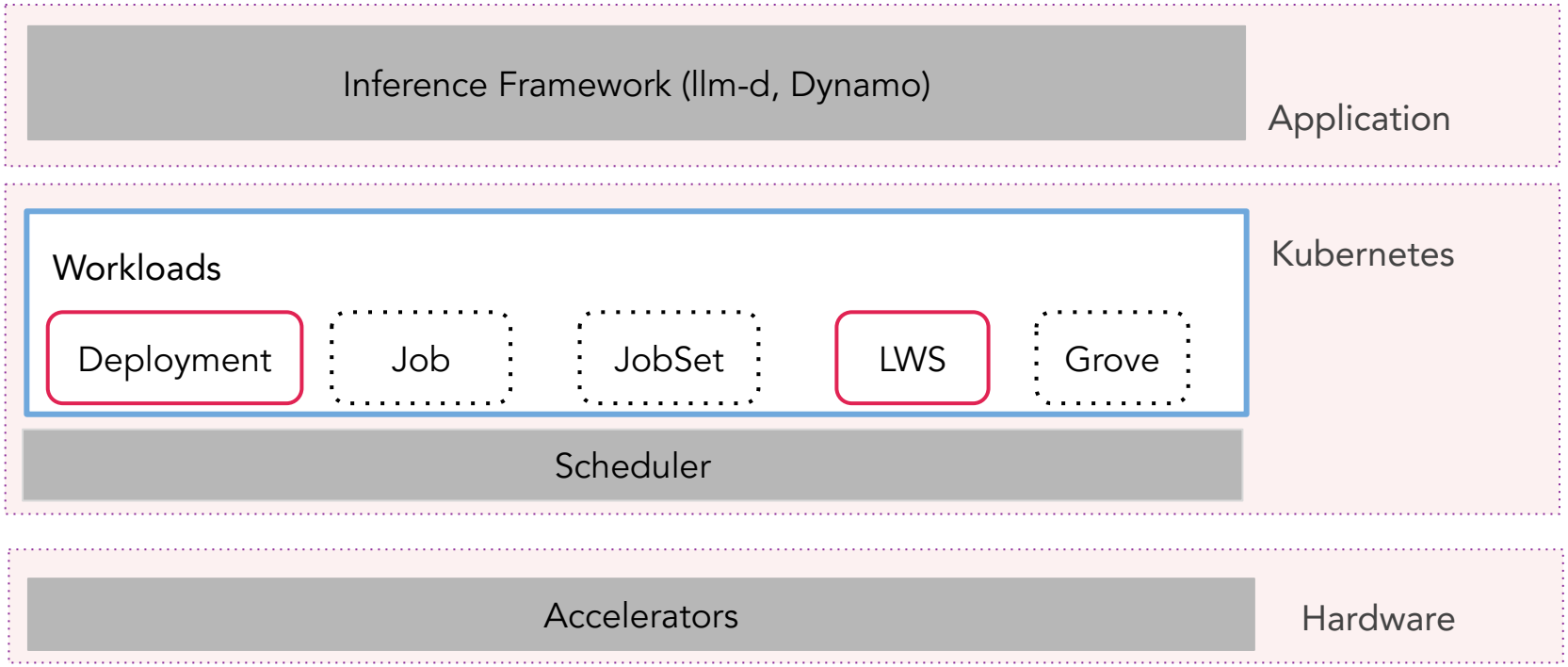
KAI: The Scheduler for AI Workloads

- Optimizes GPU allocation for AI/ML at scale
- Open-sourced by Run:ai, production-tested
- Gang scheduling across components
- Gang scheduling within components
- Topology-aware placement (TAS)
- GPU sharing, fairness, preemption, elastic scaling
- CNCF Sandbox project

Workloads Layer

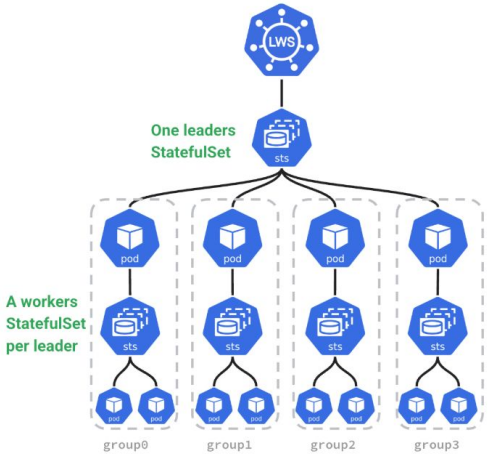


Workloads Layer

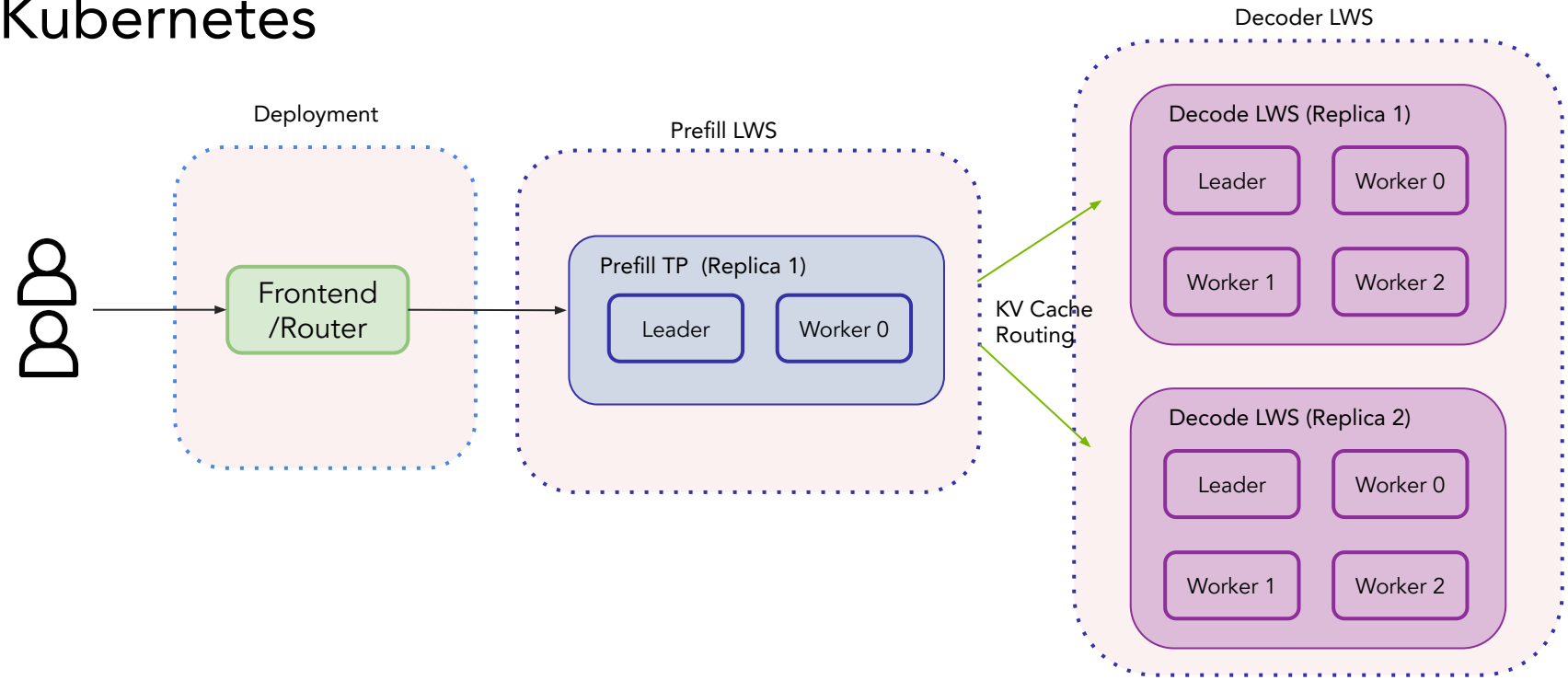


LeaderWorkerSet (LWS)

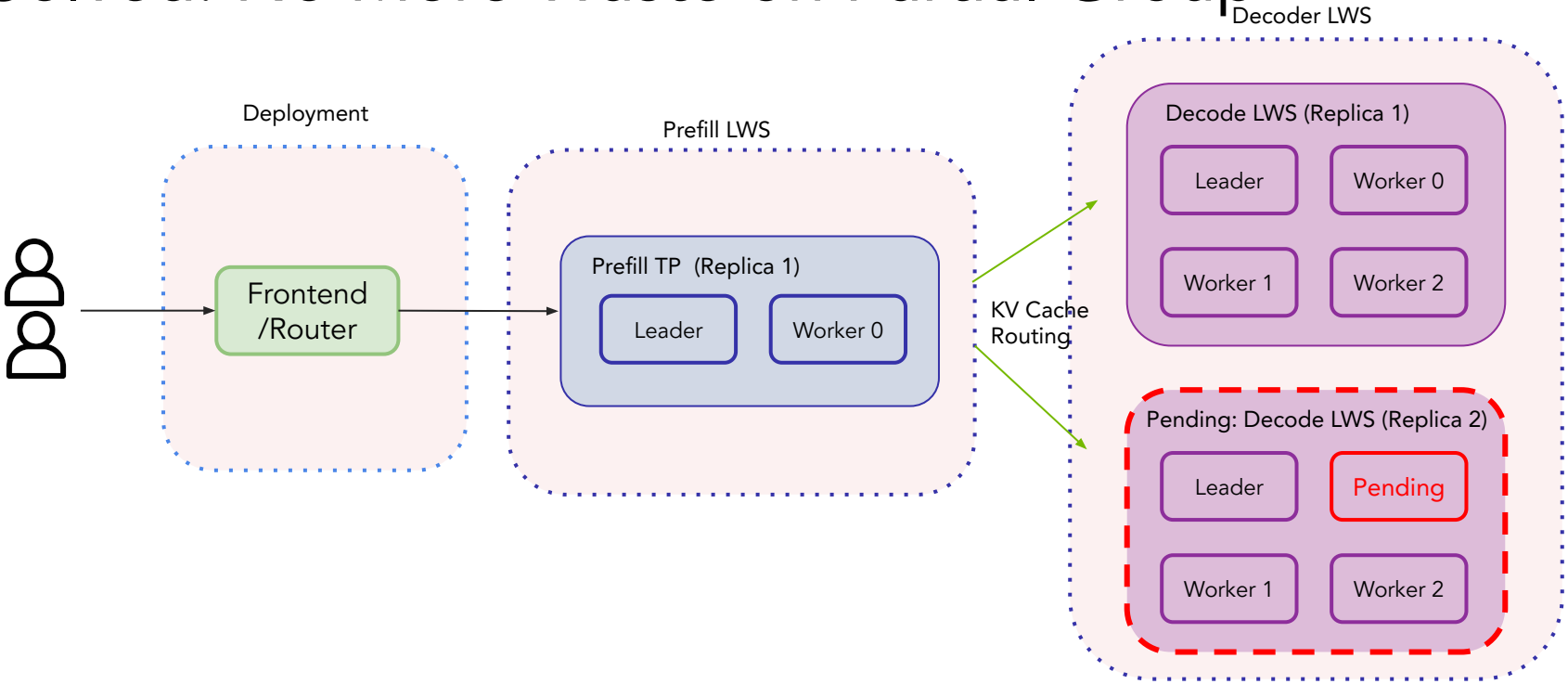
- Upstream Kubernetes project, community-backed
- Designed for multi-host inference
- One leader + N workers = one group
- Scale the whole group, not individual pieces
- Workers automatically discover the leader
- Gang scheduling + topology via supported scheduler (KAI)
- Represents a single component — not the whole app



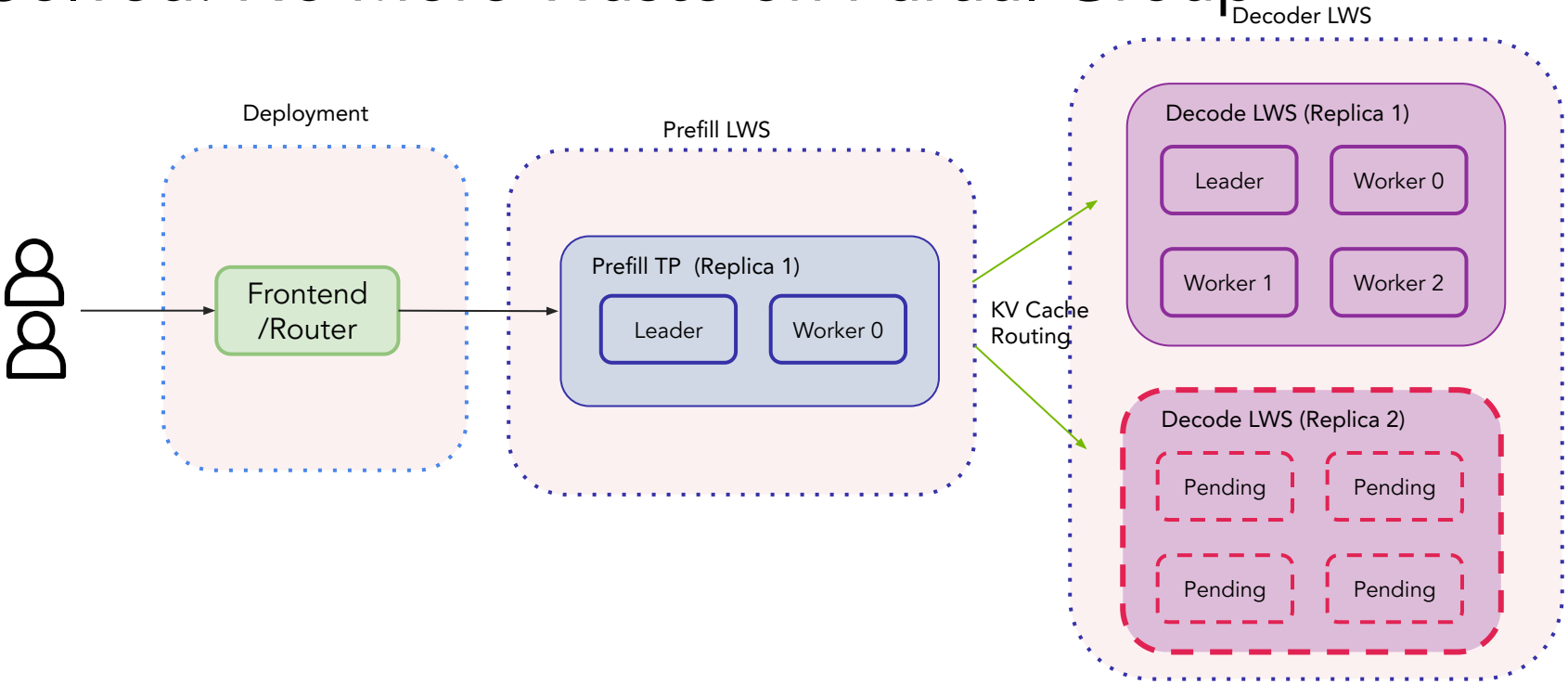
Disaggregated Inference using LWS and Deployment In Kubernetes



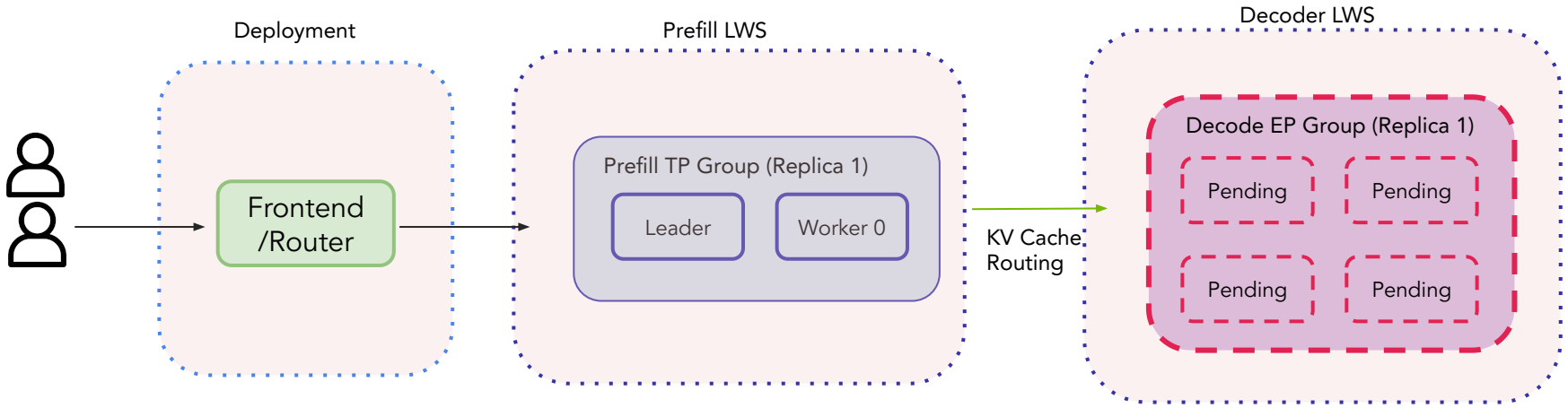
Solved: No More Waste on Partial Group



Solved: No More Waste on Partial Group



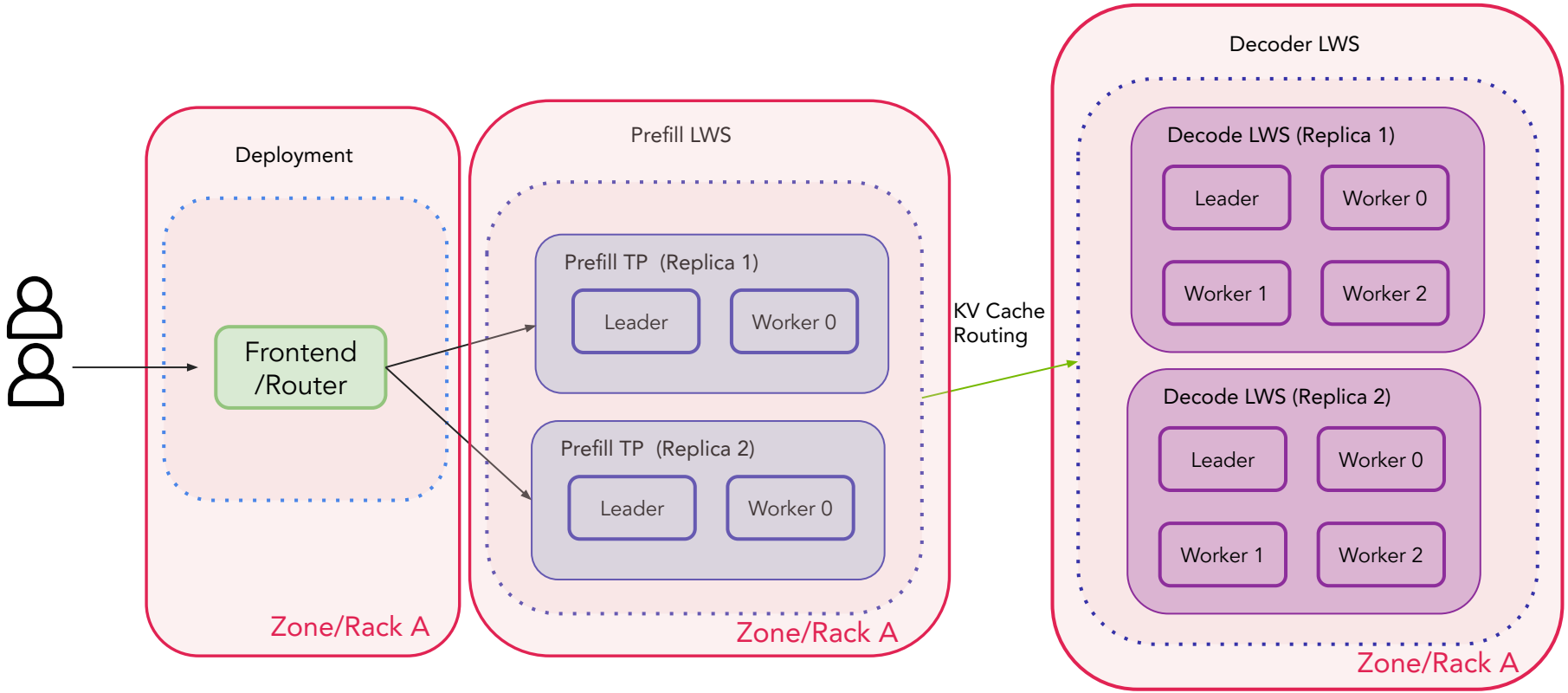
Challenge Remains - Cross component Gang scheduling



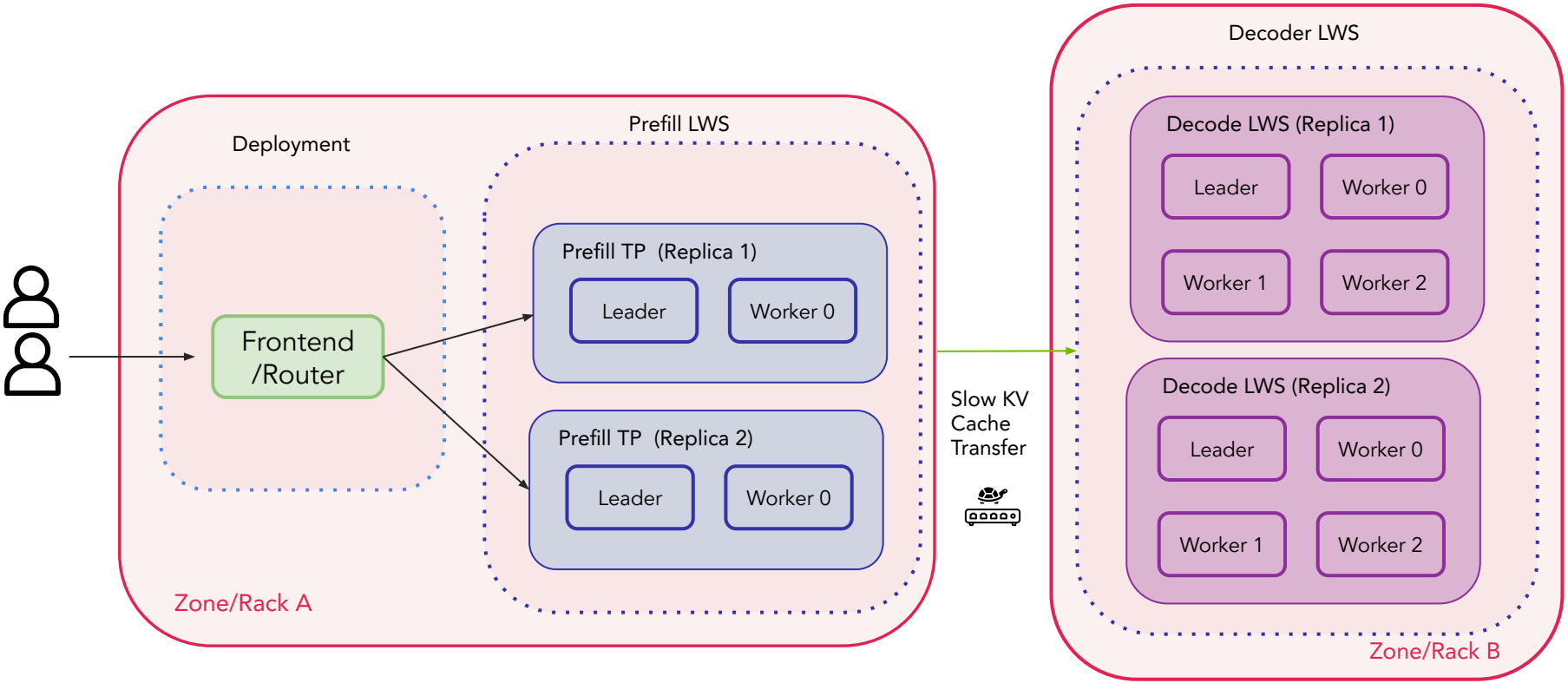
Not enough GPUs available.

Decode Group cannot schedule

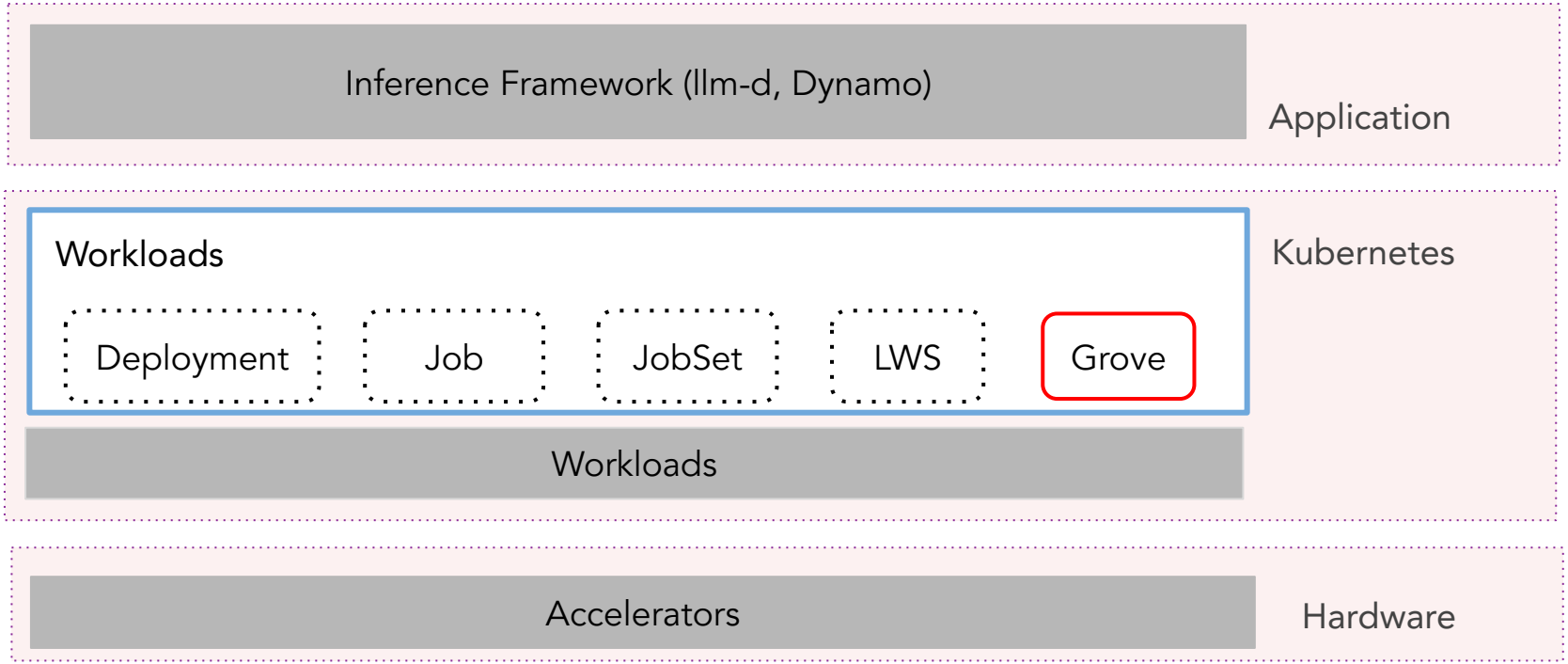
Solved: No more Spread Apart in Group!



Challenge Remains - Cross component Topology Aware Placement



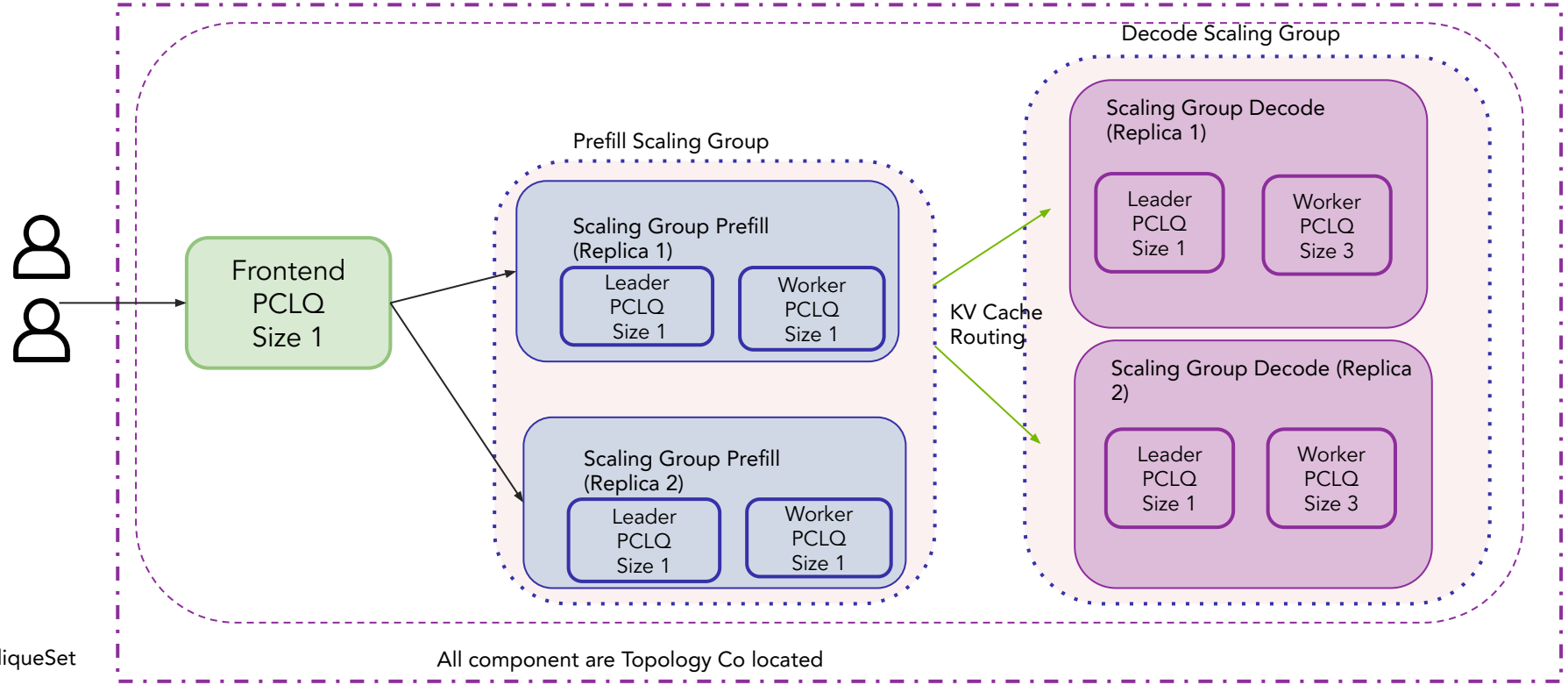
Workloads Layer



Grove: Describe Your AI Workload

- Flexible API for any AI workload — single instance to data center scale
- Built to Describe a disaggregated inference
- Multi-level gang scheduling semantics
- Built-in topology-aware placement API
- Multi-level autoscaling + coordinated rolling updates
- Works with KAI, more schedulers on the roadmap
- Open source, a modular component used by NVIDIA Dynamo
- Well-lit path doc in the making with llm-d

Disaggregate Inference using Grove



What to Take Home

1. Disaggregated inference is not a traditional service (applies to agentic pipelines as well)
    a. The unit of scaling is a coordinated group
    b. Prefill and decode have independent scaling signals and hardware profiles
    c. Components have data dependencies (KV cache) that the infrastructure must respect
2. You need at least three layers working together
    a. Inference framework (Dynamo, llm-d): the runtime
    b. Workload API (Grove, LWS): defines the workload & its components
    c. Scheduler (KAI, Kueue, Volcano): enforces placement and gang scheduling constraints
3. Choose your tools depending on your use case & setup

Check out the Projects and Get Involved!

Dynamo Github Repo

KAI Github Repo

Llm-d Github Repo

Grove Github Repo

LWS Github Repo


Thank you!
PyTorch
CONFERENCE
EUROPE 2026