A CNCF INCUBATION PROJECT

HAMi and Viettel Cloud: From Project to Production

Fractional GPU Virtualization for Multi-tenant AI Workloads

Reza Jelveh
Solution Architect, Dynamia AI - Makers of HAMi
github.com/rezajelveh linkedin.com/in/rezajelveh

The Anh Nguyen
Software Engineer, Viettel Networks - CNCF Kubestronaut
github.com/ntheanh201 linkedin.com/in/ntheanh201

DYNAMIA.AI
Creators of HAMi

Part 1: The Problem

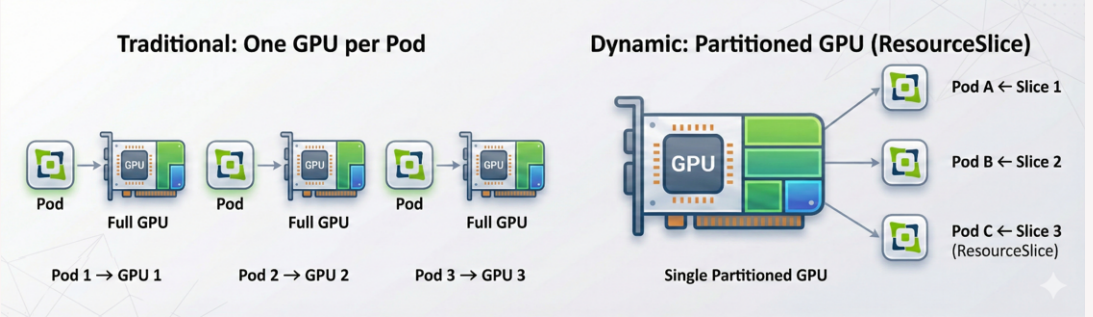
GPU underutilization, atomic allocation, multi-tenant contention

The Problem

Atomic GPU allocation wastes silicon

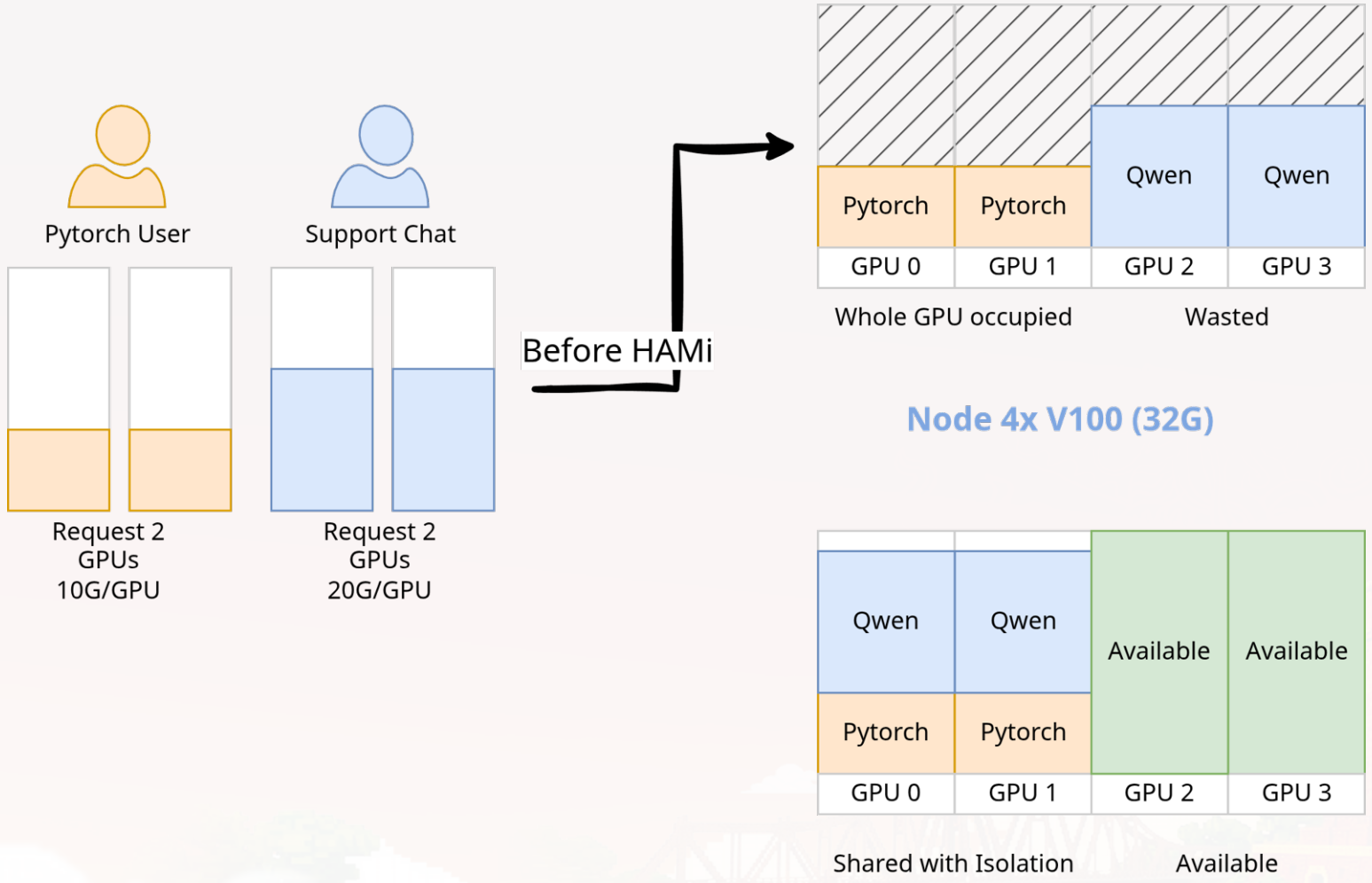
Kubernetes allocates GPUs atomically. DRA + HAMi fix this.

- 1 GB task blocks an 80 GB device
- Over-provisioning is the default: idle silicon
- DRA can request GPUs but requires MIG to slice
- Why HAMi if DRA slices? We'll get to that



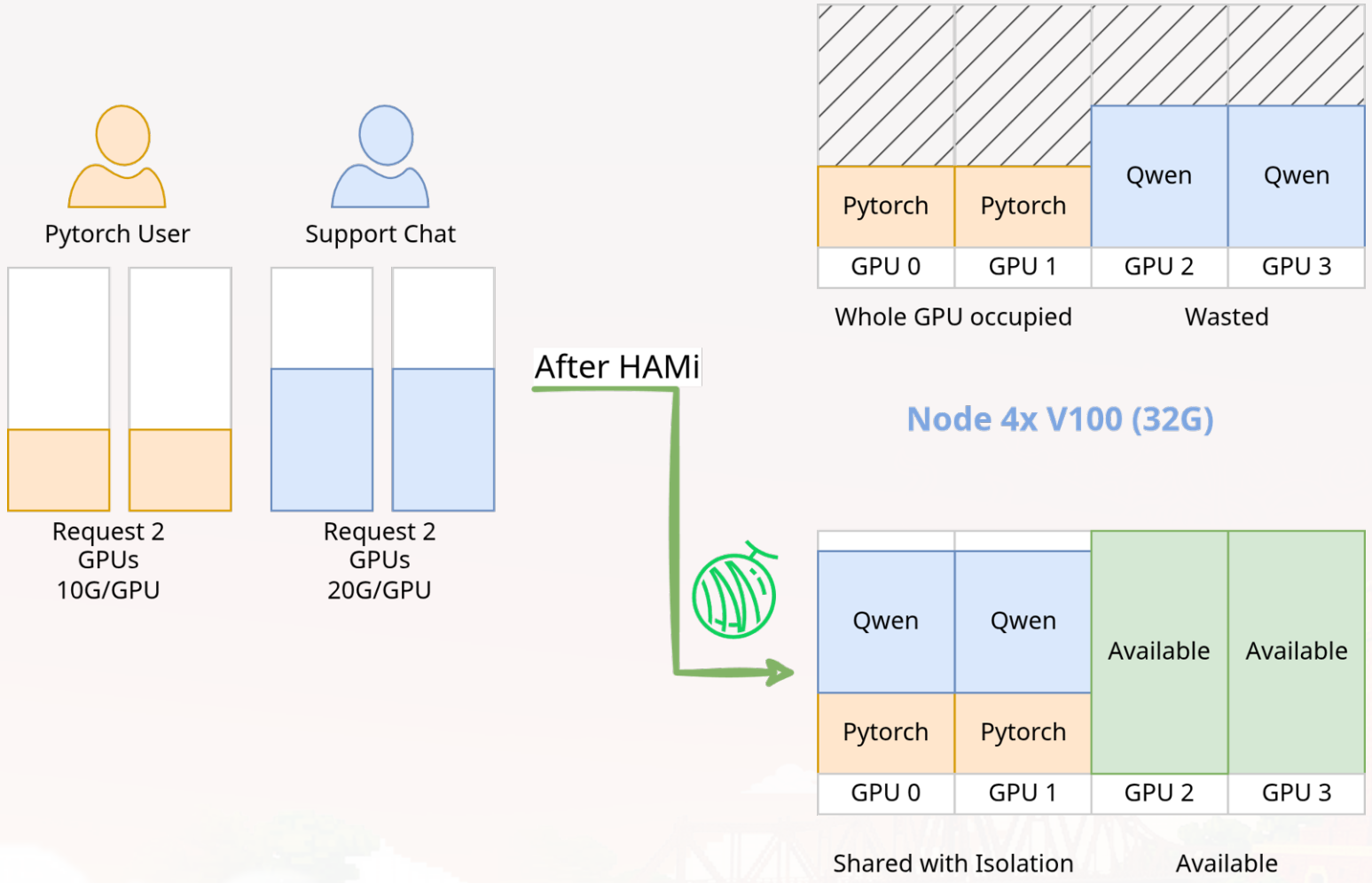
What is HAMi

Static allocation, one GPU per task



What is HAMi

Fractional vGPUs, multiple tasks per device



The GPU Challenge

What breaks, what we need

Problem

- GPUs are scarce, allocated whole
- Vendors locked in, supply tight
- Utilization stuck at ~30%
- No central observability
- Fragmented inference workloads

Requirements

- Hardware agnostic: one API, any accelerator
- Fractional GPU: 1MB slices, multiple tasks per device
- Advanced scheduling: binpack, spread, topology-aware

Unified observability, 50% GPU utilization, 10x workloads running, 10x GPU availability. AMD MI355X: 80% of B200 perf at ~1/3 the cost. Not everyone needs Vera Rubin.

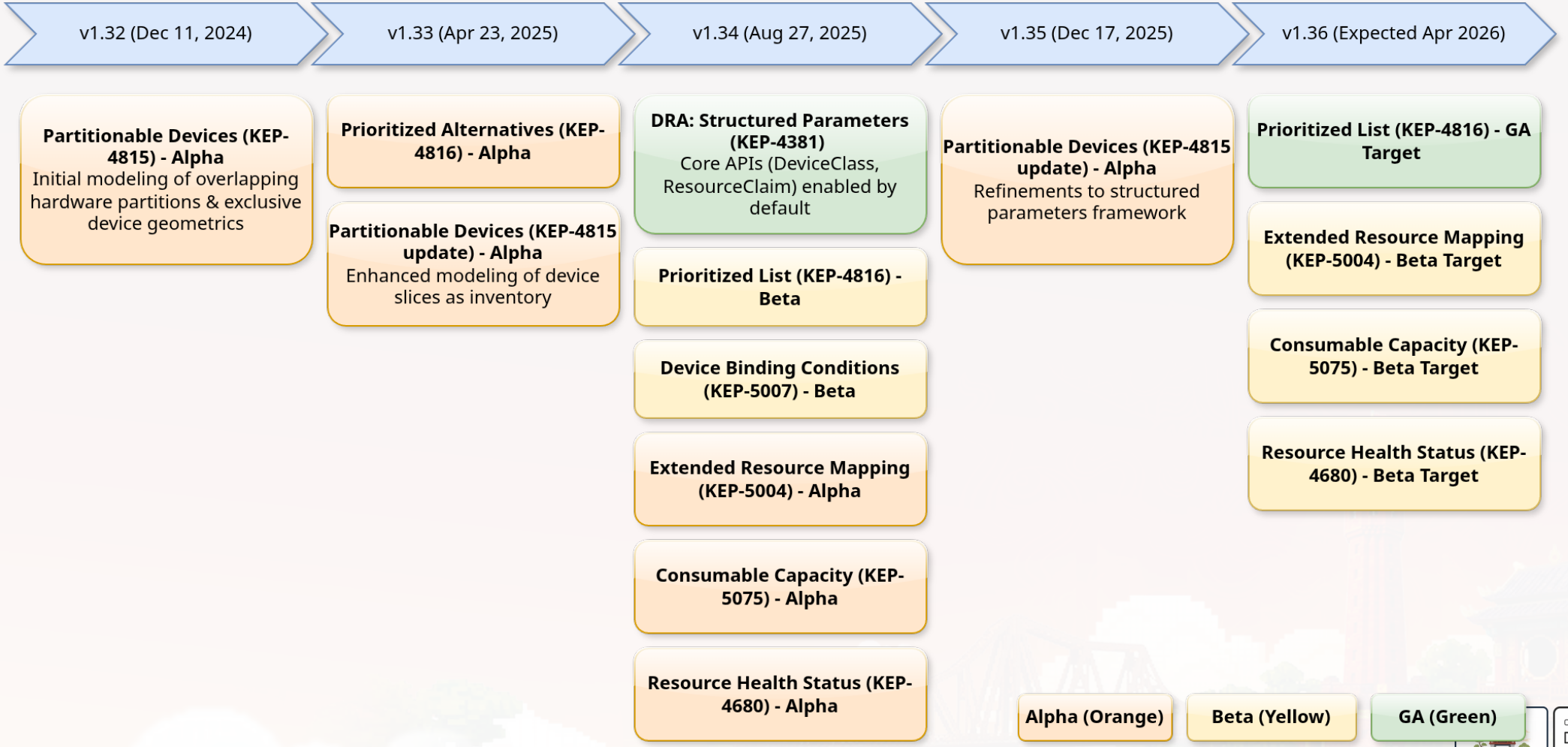
DYNAMIA.AI
Creators of HAMi

Part 2: The Solution

Fractional GPU allocation and scheduling

DRA Feature Timeline

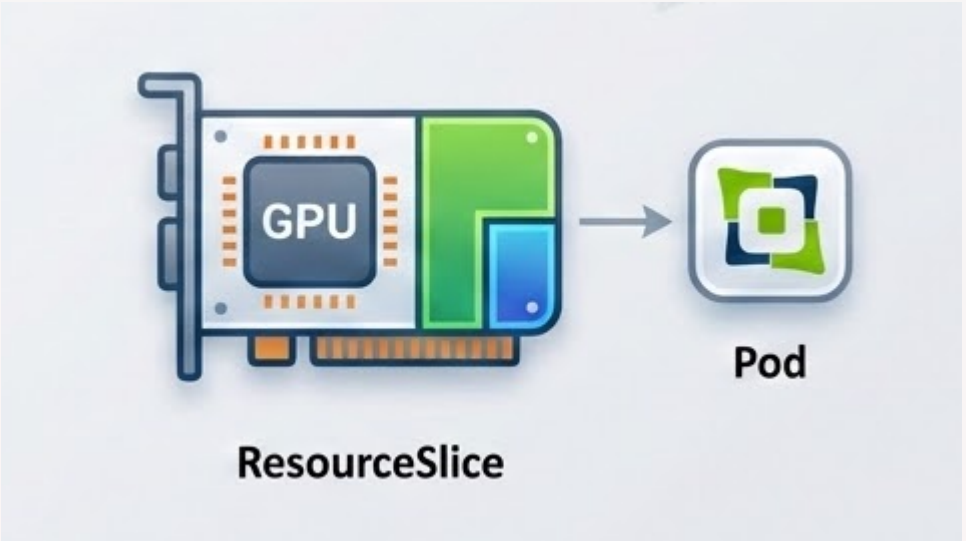
KEPs and their development status: we won't cover this today



DRA: ResourceSlice

Per-node device inventory

- **ResourceSlice:** lists all available devices on a node with their attributes
- DRA drivers publish slices; scheduler reads them to match claims to devices
- Tied to a node via `nodeName`, supports topology and NUMA attributes



DRA: ResourceClaim

A standardized way to request hardware: not just GPUs. Stable in K8s 1.34.

- **DeviceClass:** groups devices with identical resource models. **ResourceClaim:** a workload's ticket to hardware. **ResourceClaimTemplate:** reusable blueprint, auto-creates a claim per Pod.

HAMi Capabilities

Six things HAMi brings to GPU scheduling

Heterogeneous Management

Manage GPU, NPU, MLU, and other accelerators in one workflow.

Hard Isolation

Slice memory and compute with hard isolation at runtime.

Advanced Scheduling

Binpack, spread, and topology-aware placement policies.

Kubernetes Native

Kubernetes-native APIs, DRA, and CDI support.

Resource Isolation & QoS

Memory and core quotas for fair, stable sharing.

Unified Monitoring

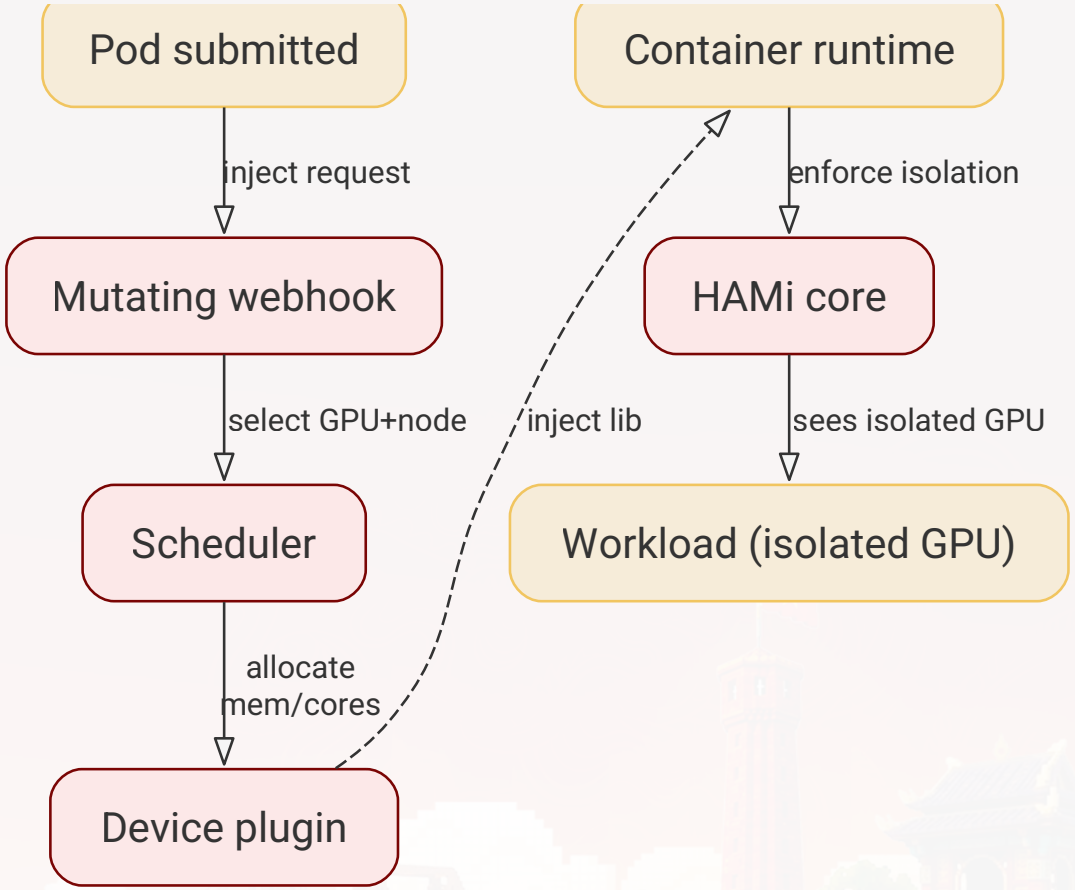
Consistent metrics and visibility across vendors.

How HAMi Works

Pod creation to GPU isolation

HAMi intercepts pod creation and GPU allocation through seven stages:

- **Mutating webhook:** injects device request into pod spec
- **Scheduler:** selects GPU and node via HAMi scheduler plugin
- **Device plugin:** allocates GPU memory and compute cores
- **Container runtime:** injects HAMi-Core library into container
- **HAMi core:** enforces isolation in-process via symbolic hijacking



GPU Sharing

Dynamic fine-grained device slicing

- **NVIDIA, Ascend, Cambricon, Hygon, Iluvatar** supported
- **Fine-grained:** as small as 1MB device memory, 1% computing cores
- **Transparent to tasks:** no code changes required
- **Hard resource isolation** inside containers
- One API across vendors: same YAML, any accelerator

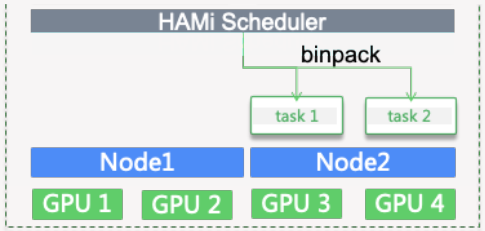
```yaml
# Ascend 910C: 8GB + 20% compute
resources:
  limits:
    huawei.com/Ascend910C: "1"
    huawei.com/Ascend910C-core: "20"
    huawei.com/Ascend910C-memory: "8192"
```

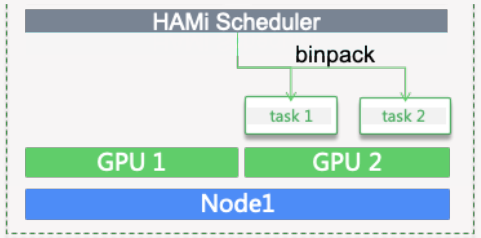
```yaml
# NVIDIA: 3GB on any GPU
resources:
  limits:
    nvidia.com/gpu: 1
    nvidia.com/gpumem: 3000
```

Scheduling Policies

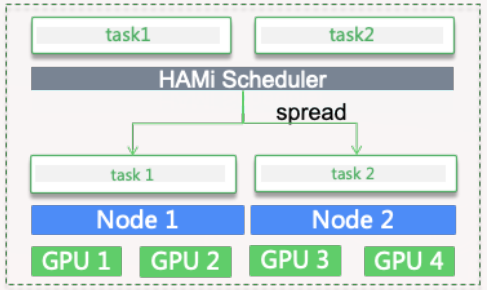
Binpack & Spread

- **Node binpack** frees whole machines: reduces cost, helps cluster autoscaler
- **Node spread** isolates faults: HA across zones, blast radius control
- **GPU binpack** prevents fragmentation: frees entire GPUs for training
- **GPU spread** protects tail latency: reduces HBM and NVLink contention
- Advanced scheduling works with standalone HAMi; DRA mode can use Yunikorn or Volcano

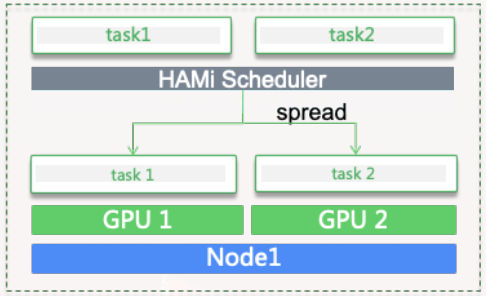




Node Spread



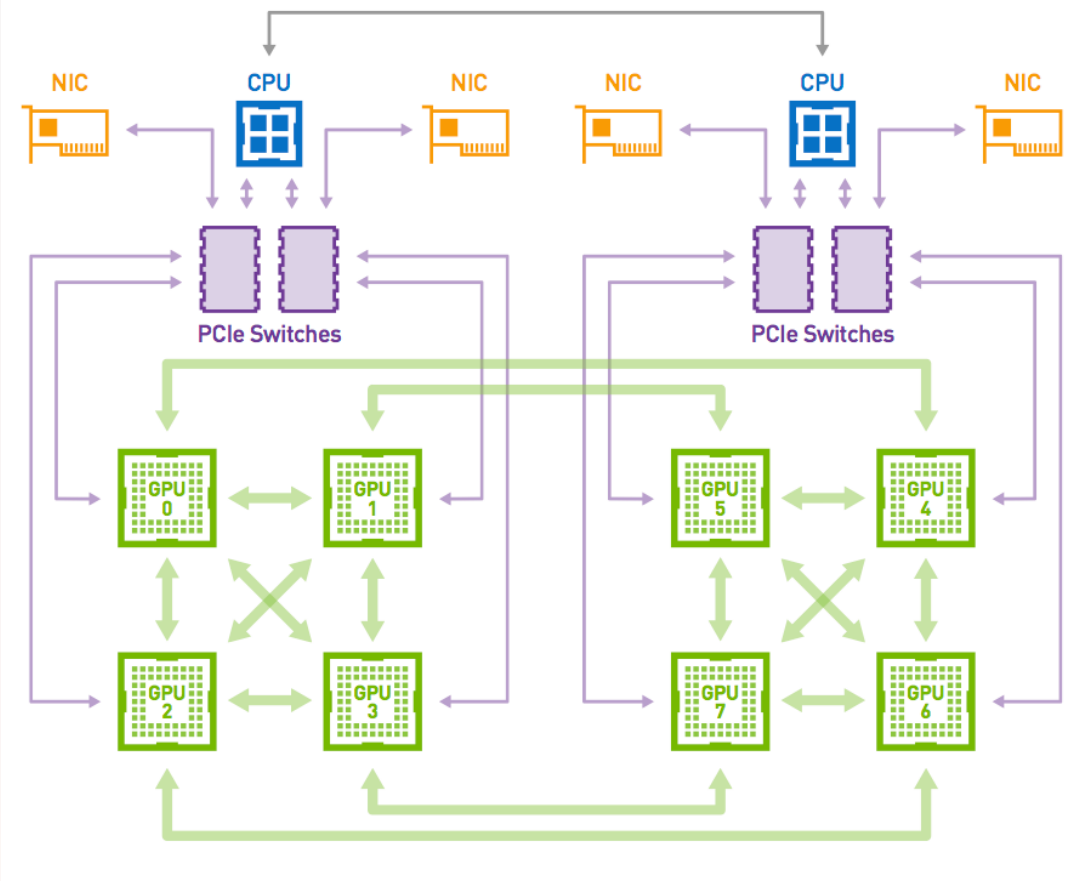
GPU Spread



Scheduling Policies

Topology-Aware

- **NVLink 3 (A100):** 600 GB/s, 12 links
- **NVLink 4 (H100/H200):** 900 GB/s bidirectional across 18 links
- **NVLink 5 (B200/B300):** 1.8 TB/s, 14x PCIe 5.0
- **NVLink 6 (Rubin):** ~3.6 TB/s target
- **PCIe 5.0 x16:** 128 GB/s. **PCIe 6.0:** 242 GB/s
- **HAMi topology policy:** prefers NVLink (NVIDIA), HCCS (Ascend), and other high-speed interconnects, avoids PCIe bridge pairs



GPU Sharing Approaches

MIG vs HAMi vs HAMi+DRA vs NVIDIA DRA

Capability	MIG	HAMi	HAMi+DRA	NVIDIA DRA
Pre-configured templates	✗	✓	✓	✗
Dynamic MIG repartition	✗	✓	✓	✗
Symbolic hijacking (1MB slice)	✗	✓	✓	✗
Multi-vendor	✗	✓	✓	✗
Advanced scheduling	✗	✓	✗	✗

MIG needs preconfigured GPU profiles. HAMi creates dynamic MIG partitions based on workload, and also uses symbolic hijacking for slices as small as 1MB. NVIDIA DRA is MIG-only: no repartitioning, no hijacking, no multi-vendor, no advanced scheduling (yet).

DYNAMIA.AI
Creators of HAMi

Part 3: Viettel Cloud

What we measured, what broke, and what we learned

KCD VIETNAM
OpenInfra DAYS VIETNAM

Who We Are

Viettel Cloud, the AI Platform, and the GPU-sharing slice we'll cover

The Anh Nguyen

Software Engineer, Viettel Networks

- Technical lead, **Viettel AI / GPU Platform**
- **CNCF Kubestronaut**
- **NVIDIA Certified Professional**, AI Ops
- **CNCF Glossary** VN Lead, **SIG Docs VI** approver
- **Kubernetes & HAMi** member

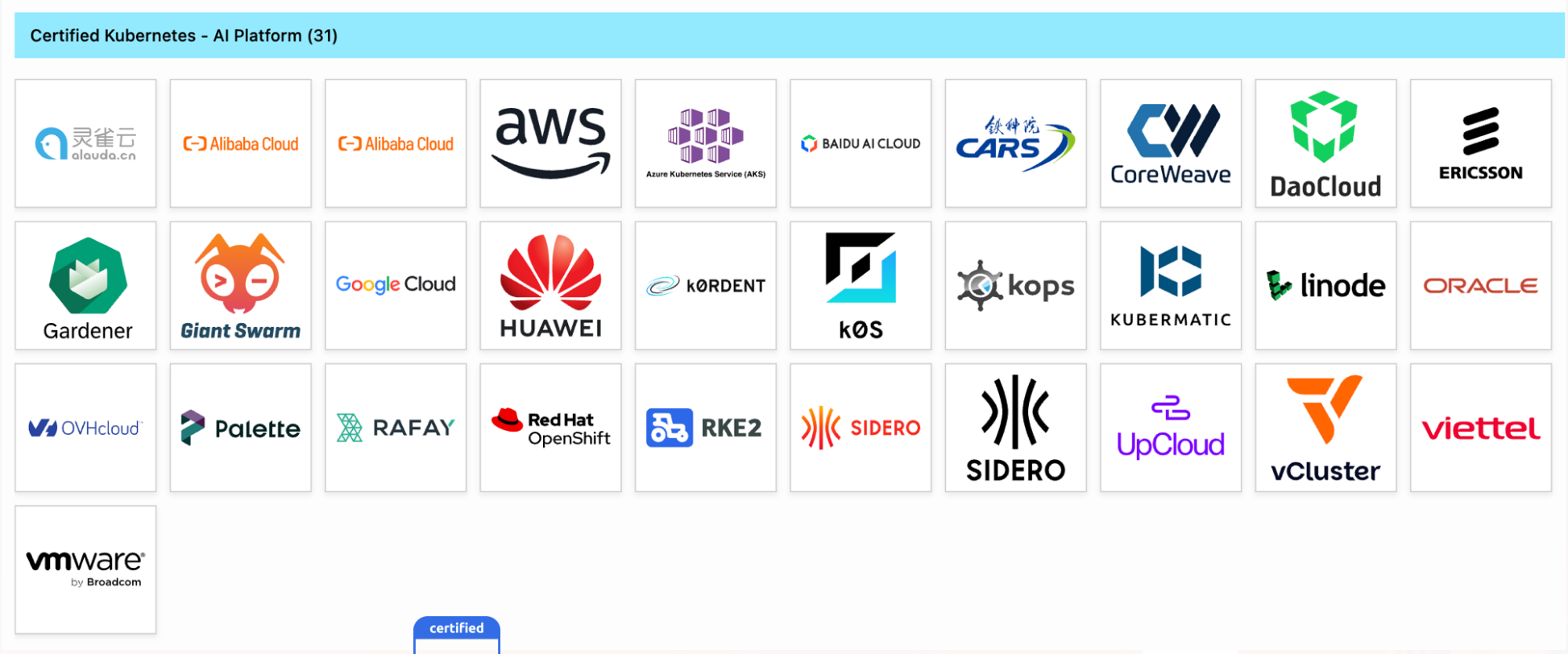
Viettel Cloud

Cloud platform of Viettel Group, Vietnam's largest telco.

- **Viettel Cloud AI-Native** - the AI / GPU platform
- **AI Notebooks with fractional GPU** - dev & training
- Also **inference serving**
- **H200** pool (141 GB), plus **L40, L40S, A30, A6000, ...**

Certified Kubernetes AI Platform

31 platforms in the world have it. We are one.



certified AI Platform

Our **AI Platform** - **first & only in Vietnam**, **#20** worldwide.

What We Run Today

HAMi in our clusters

- **HAMi v2.9** on **both clusters**: the H200 pool, and the mixed-GPU one
- Our own engineers use it every day
- The memory limit is real: pod sees **2 GB, not 141 GB**
- Lives next to **Slinky, Inference, KEDA**
- Full monitoring, down to **per-pod GPU metrics**

This is all a user writes:

```yaml
resources:
  limits:
    nvidia.com/gpu: 1
    nvidia.com/gpumem: 2000   # MB - a hard limit
    nvidia.com/gpucores: 30   # % - best effort
```

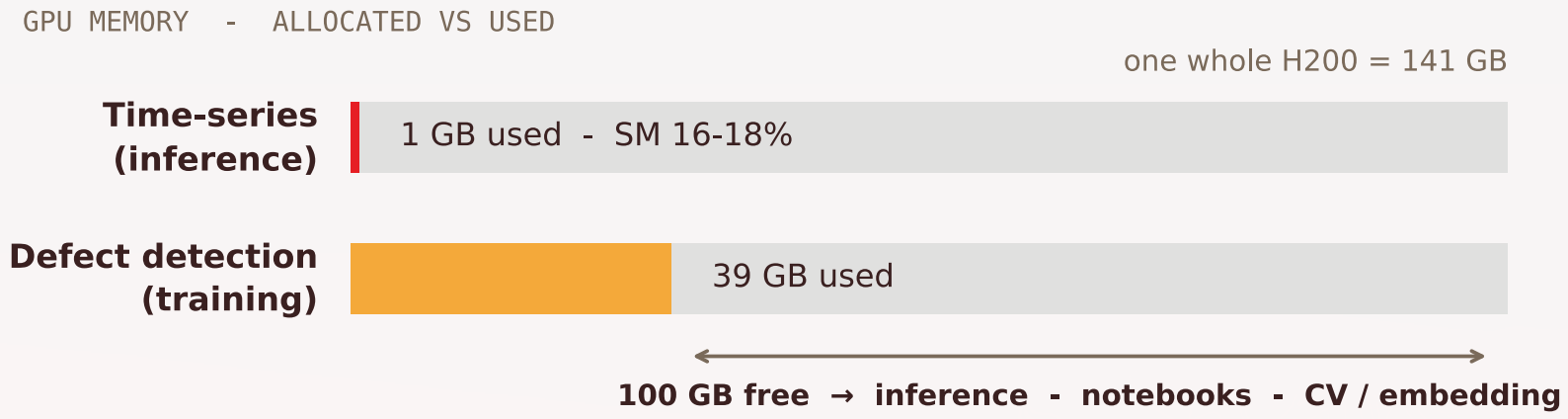
Our researchers do not write that - they pick the GPU size in our **AI Notebooks**, built on **Kubeflow Notebooks**.

Numbers in this talk: mostly the **H200 pool**, a few from **L40**.

The Problem We Measured

Asking for a GPU is not the same as using it

Two real jobs. Each one held **a whole H200, 141 GB**:

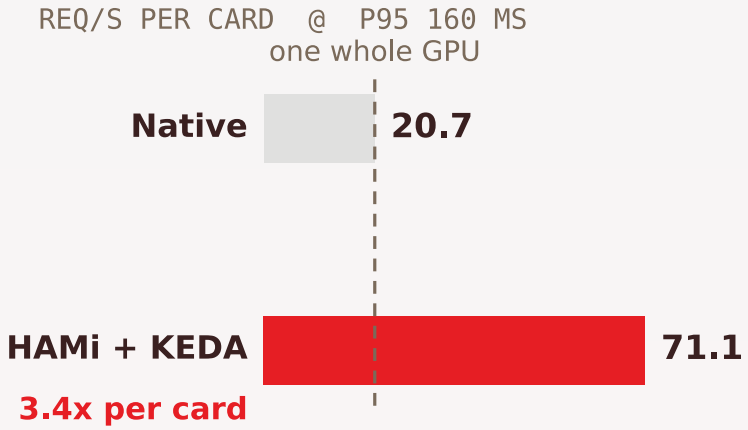


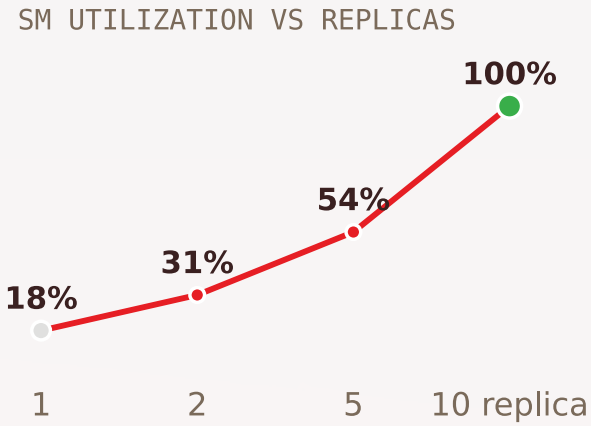
- The time-series job leaves **140 GB and 80% of the GPU** doing nothing
- Detection training took **39 GB** - it still left most of the card unused
- **Many of our GPUs have no MIG:** L40, L40S, A6000, ...

> Kubernetes gives you one choice: `nvidia.com/gpu: 1`. The whole card, or nothing.

The Result: 3.4x More Work Per GPU

Fix the SLA, scale replicas, fill the empty GPU





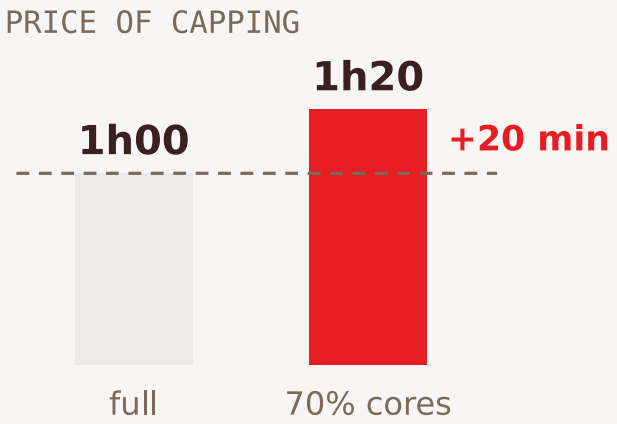
One pod uses only **18% of the card**. Pack **10 replicas** on it and the GPU hits **100%** - same latency.

- **HAMi does not make one pod faster** - it fills the GPU that was sitting empty
- Bigger batches did not help either: **32 to 512**, GPU still 7-10%
- Same 71 req/s (H200): **1 GPU instead of 3.4 → ~71% fewer cards**, **~51% less power**, **~$7k/month** at an example $4/GPU-h

One Card, One Day

One slice guaranteed, on purpose - and what it costs

Workload: an AI Notebook training YOLO11 - ~17k images, batch 64 → **39 GB** used. Batch is sized for **accuracy**, not to fill the card - so the spare **~100 GB is genuinely free**, not just idle. We fence **~30%** and cap the notebook to **70% of the cores**.



- **Memory is fenced.** The notebook sees only its own slice - it cannot OOM the neighbours
- **Cores are capped** with `force` - the freed share is **reserved**, not "maybe"
- **The price:** ~20 minutes, on purpose - *a small model; a big one costs much more*

The Rest Follows The Load

One notebook fixed all day - inference flexes by the hour

- 🔒 **The notebook never moves.** Its slice is fixed all day → training stays stable and guaranteed
- 🔄 **KEDA hands the freed space off through the day.** The morning **burst fades** → cores go to **LLM <20B** → one time-series instance stays for scattered load → at night, spare **auto-scales more notebooks** for extra overnight training. *Never idle.*

But It Does Not Always Help

HAMi fills the *empty* part of a card - a full one has nothing to give

✓ It helps when the card is empty

Inference: one pod = **~18% SM**, card **82% idle**.

- Pack ~10 → SM hits **100%** → **3.4x more work**

✕ Not when it is already full

Training / big LLM: one instance **already saturates** the card.

- Big LLM: one **11.6k** tok/s **>** four small **7.3k**
- Same training job **× 4**: **0.77x** - packing *loses*

Careful what you promise your users:

What you ask for	What you get
gpumem - memory	A real limit. Your neighbour is safe
gpucores, default	Only a hint. 10% and 90% ran the same speed
gpucores, force	A real cap. 70% cores → run takes 1h20, not 1h

What It Really Took

The hard part is not installing HAMi

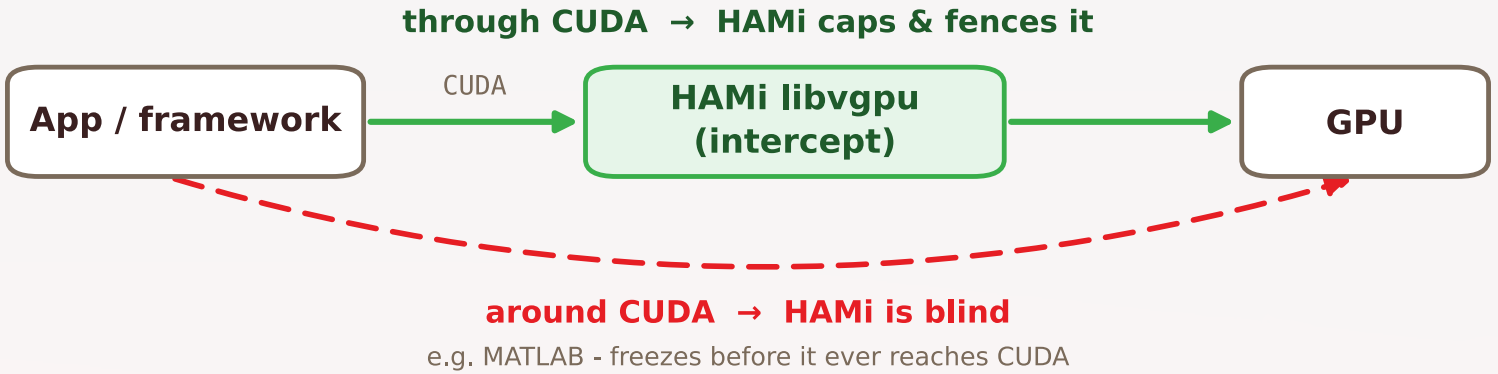
Obstacle	Solution
HAMi's privileged access conflicts with Kyverno policies	Exclude hami-system from the pod-security policies
nvidia.com/gpu is ambiguous - claimed by both the NVIDIA device plugin and HAMi	Operator device plugin off, CDI off
Resource-pool overlap - fractional, full-GPU, and Slurm (Slinky)	Own pool, own scheduler, and a taint
HAMi 2.9 metrics refactor - Grafana dashboards mismatch, go empty	Use hami_*. We fixed the docs upstream

It is an operations project, not a one-line install.

Where The Magic Breaks

Useful to know before you debug for two days

HAMi sits in front of every CUDA call. **Go around CUDA, and HAMi cannot see you.**



- 📦 **Alpine / musl images crash.** libvgpu - HAMi's `LD_PRELOAD` interposer - needs **glibc**; on Alpine/busybox it can't find `libdl.so.2` (exit **127**). Use a glibc CUDA image - Debian or Ubuntu
- ✕ **Runtimes that skip CUDA hang.** MATLAB freezes *before* it ever reaches a CUDA call
- ✓ **What does not break:** even multi-GPU - P2P, all-reduce, tensor parallel + CUDA graphs on NCCL 2.28 - all stays inside CUDA

What Is Next For Us

From one pool to a real service

- **More teams** inside Viettel, with quota per team
- **More pools** - the H200 and mixed-GPU pools today, the rest of the fleet next
- **More workload types** on the same card - agents, bigger LLMs, batch jobs
- **Move to DRA** - keep the same isolation, swap the custom API for the standard Kubernetes one

DYNAMIA.AI
Creators of HAMi

Part 4: Where We Are Today

Adoption, Case Studies, Community

Where We Are Today

HAMi in Production: 7 case studies, more coming

3x	10x	80%	2x
GPU density	Workloads per device	Less ops overhead	GPU utilization

China Merchants Bank · SNOW Corp. · NIO · KE Holdings · DaoCloud · SF Technology · Prep Education · cncf.io/case-studies · Reach out for help submitting yours.

Community & Adopters

Devices, integrations, and who uses HAMi

Open Source, CNCF Backed, Production Ready

4.1k
Github Stars

325k
Docker Pulls

500+
Contributors

27
Contributor Countries

kubernetes VOLCANO Kueue koordinator KAI Scheduler COZYSTACK

Ecosystem & Device Support

NVIDIA Ascend Cambricon 寒武纪 HYGON 中科海光 天数智芯 Iluvatar CoreX METAX 沐曦 摩尔线程 MOORE THREADS 昆仑芯 KUNLUNXIN Enflame 燧原科技 aws 瀚博半导体

Adopters

4Paradigm 百度智能云 贝壳 招商银行 中国移动 China Mobile 中国联通 China unicom DaoCloud DYNAMIA H3C HUAWEI LinkedIn 马上消费 NIO PPIO PREP SAP SF TECHNOLOGY 思特奇 SI-TECH SNOW viettel

DYNAMIA.AI
Creators of HAMi

THANK YOU

Questions?

Reza Jelveh
Solution Architect, Dynamia AI - Makers of HAMi
github.com/rezajelveh linkedin.com/in/rezajelveh

The Anh Nguyen
Software Engineer, Viettel Networks - CNCF Kubestronaut
github.com/ntheanh201 linkedin.com/in/ntheanh201