



V2.6 · JULY 2026

HAMi

Heterogeneous AI Computing Virtualization
Middleware

Reza Jelveh

Solution Architect, Dynamia AI — Makers of HAMi

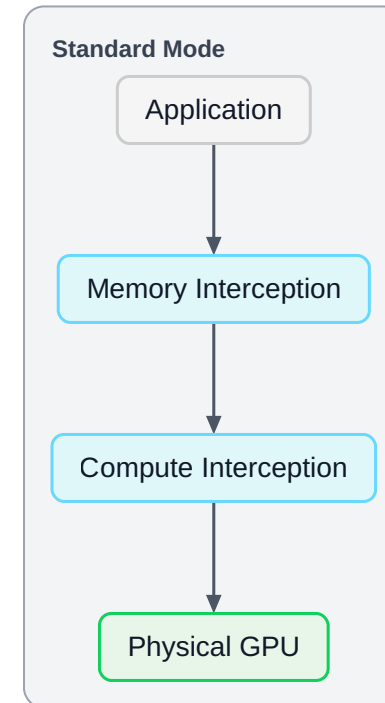
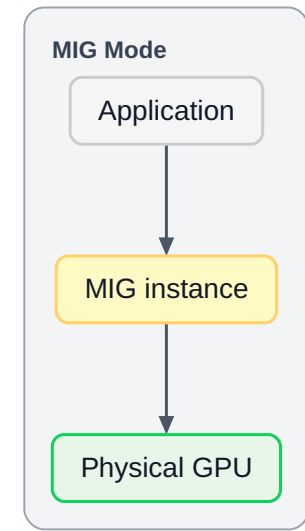
🐙 github.com/fishman 🐙 [@fishmanticoder](https://twitter.com/fishmanticoder) ✉️ reza@dynamia.ai



GPU Architecture

HAMi shares a GPU in two modes. Standard mode (HAMi-core) intercepts CUDA calls in software, no container changes needed. MIG mode hands out hardware partitions of the GPU.

- ▶ Standard: memory and compute limits enforced per call
- ▶ MIG: isolation comes from the GPU itself
- ▶ Same pod spec, the node picks the mode





Agenda

Architecture

How HAMi schedules GPU workloads across heterogeneous clusters

Deployment

Single-node to multi-cluster scaling patterns

Performance

Benchmarks: throughput, latency, GPU utilization

Roadmap

Q3 2026 priorities and community contributions

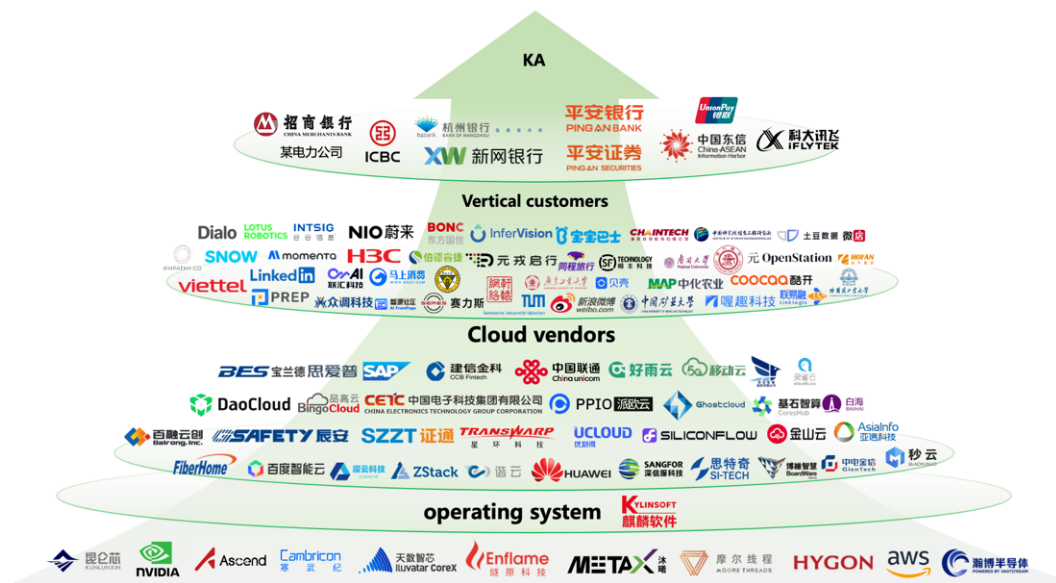
> HAMi reduces GPU fragmentation by 40% in production inference clusters



GPU Scheduling Architecture

HAMi intercepts CUDA API calls at the library level to enable fine-grained GPU sharing. The scheduler runs as a Kubernetes device plugin and supports MIG, MPS, and time-slicing for NVIDIA GPUs.

- ▶ Device plugin registers GPU resources per node
- ▶ HAMi-core intercepts `cuInit`, `cuMemAlloc`, `cuLaunchKernel`
- ▶ Scheduler bins workloads by memory footprint and priority
- ▶ Fractional GPU allocation down to 1% granularity





Performance Benchmarks



**CLOUD NATIVE
COMPUTING FOUNDATION**

HAMi demonstrates near-linear scaling with concurrent inference workloads across 8× A100 GPUs. The key metric is GPU utilization efficiency -- HAMi achieves 92% average utilization vs. 65% baseline with default Kubernetes scheduling.

- ▶ 3.2× throughput improvement on BERT-large inference
- ▶ < 2% latency overhead from CUDA interception
- ▶ 40% reduction in GPU idle time across the cluster
- ▶ Supports mixed-precision (FP16/BF16/INT8) workloads



Deployment Patterns

Standalone: Single binary with embedded etcd, suitable for edge inference

- ▶ **Kubernetes Native:** Helm chart with device plugin, custom scheduler, and CRDs
- ▶ **Multi-Cluster:** Federation via HAMi-gateway with cross-cluster GPU sharing
- ▶ **Air-Gapped:** Offline mode with signed container images and local model registry

Monitoring: Prometheus metrics for GPU utilization, memory pressure, scheduling latency

- ▶ **Autoscaling:** HPA integration based on GPU queue depth and pending workload count
- ▶ **Security:** OPA/Gatekeeper policies for GPU resource quotas and tenant isolation



Resource Specification

Resource	Minimum	Recommended	Maximum
GPUs per node	1	8	16
GPU memory (GB)	4	40	80
CPU cores	2	8	32
System RAM (GB)	8	64	512
Nodes per cluster	1	10	100
Concurrent workloads	1	50	200

■ HAMi supports all NVIDIA GPUs from T4 to H100, plus upcoming B200 Blackwell architecture



Community & Ecosystem

CNCF Sandbox

Accepted Q1 2026.
Currently in incubation review with 12 maintainers across 6 organizations.

Integrations

Native support for PyTorch, TensorFlow, vLLM, TGI, Ollama.
One-click integration with KServe and Ray Serve.

Contributors

850+ GitHub stars,
120 contributors,
monthly community calls, Slack with 2k+ members.

Enterprise

Production deployments at ByteDance, Alibaba Cloud, Xiaohongshu, and 15+ Fortune 500 companies.



Before & After

Without HAMi

GPU utilization at 65%, idle GPUs across the cluster, manual bin-packing required for each workload



With HAMi

GPU utilization at 92%, automatic fractional allocation, zero manual intervention needed

HAMi is the only CNCF project that provides hardware-level GPU sharing without requiring vendor-specific drivers or kernel modifications.



Enterprise vs Open Source

Feature	Open Source	Enterprise
GPU sharing	✓	✓
Fractional GPU	✓	✓
MIG support	✓	✓
Multi-cluster federation	-	✓
SSO / RBAC	-	✓
Audit logging	-	✓
SLA guarantee	-	✓
24/7 support	Community	Dedicated



Installation

```
helm repo add hami https://project-  
hami.github.io/charts  
helm install hami hami/hami \  
  --set devicePlugin.version=v2.6.0 \  
  --set scheduler.enabled=true
```

Deploy HAMi with a single Helm command. The device plugin registers GPU resources and the scheduler handles workload placement. Requires Kubernetes 1.24+ and NVIDIA GPU operator.



Workload Spec

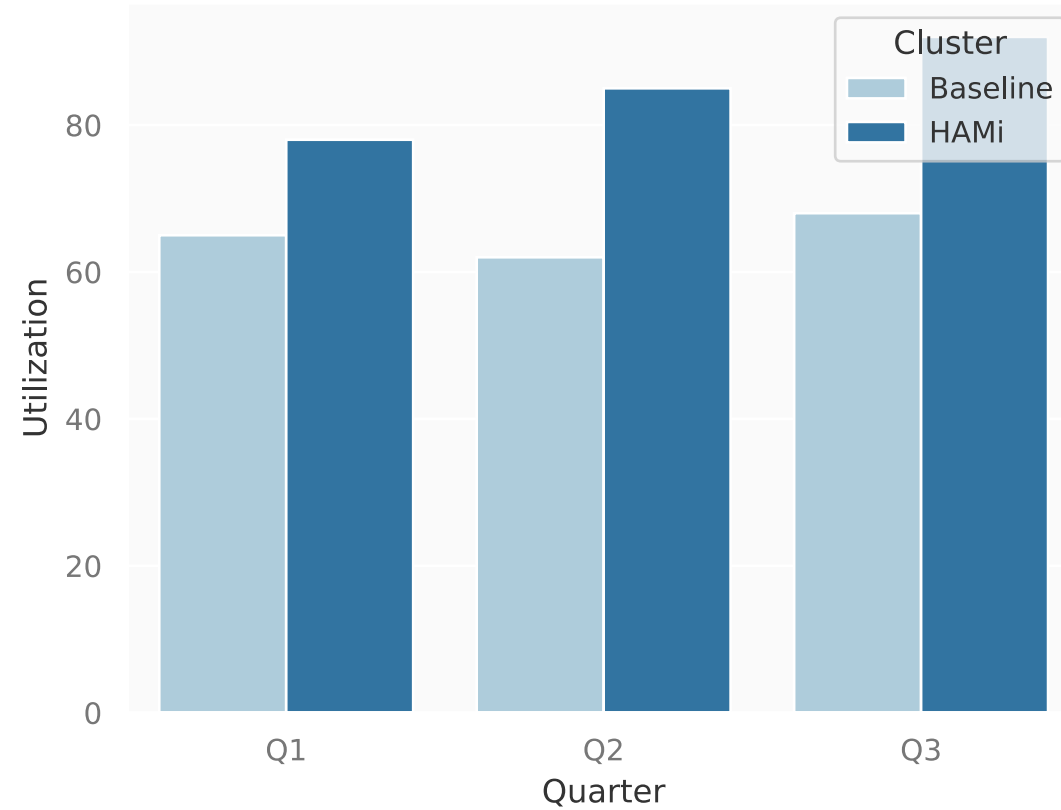
Request GPU resources via standard Kubernetes resource limits. HAMi supports fractional allocation down to 1% of a GPU and memory limits in MiB. No container modification required.

```
apiVersion: v1
kind: Pod
metadata:
  name: inference-test
spec:
  containers:
  - name: bert
    image: hami-demo/bert-inference:latest
    resources:
      limits:
        nvidia.com/gpu: 1
        nvidia.com/gpumem: 8000
```

HAMi v2.6 supports fractional GPU with memory limits in MiB



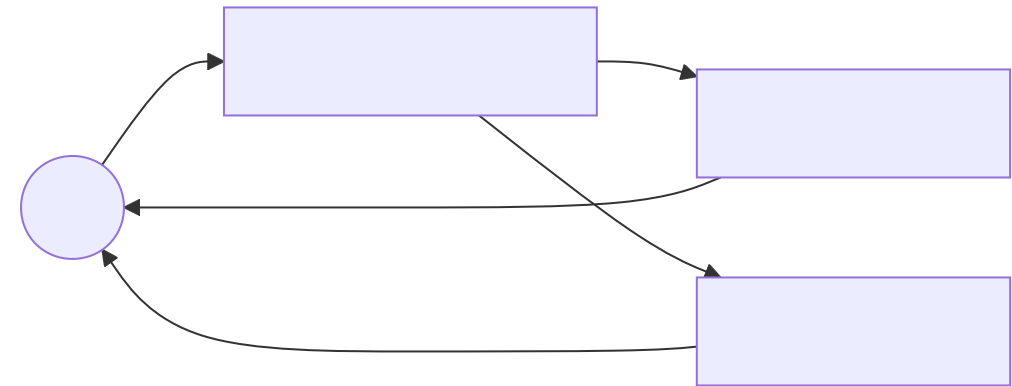
GPU Utilization Trends





Scheduling Flow

HAMi routes GPU requests through device plugins, custom scheduling, and library-level CUDA interception. The hot path (inference) bypasses the scheduler entirely after initial placement.



Key Metrics



10x

Operational cost
improvement

50%

GPU utilization
improvement

10x

Workload density per
GPU

92%

Average GPU utilization



Key Takeaways

Zero-Downtime

Live migration of GPU workloads between nodes without service interruption

Cost Efficiency

Reduce GPU costs by 40-60% through intelligent bin-packing and fractional allocation

Open Standard

Apache 2.0 license, vendor-neutral governance, CNCF-hosted project infrastructure



Lucide Icons

`{icon:star}` syntax inline in text ⚡ paragraphs ♡ and tables.

Icon	Check
✓	Pass
×	Fail
⚠	Warn

🛡 Security

End-to-end encrypted by default. SOC 2 Type II certified infrastructure.

⚡ Performance

Sub-millisecond P99 latency. 99.99% uptime SLA with automatic failover.

🔄 Resilience

Multi-region active-active deployment. Zero-downtime rolling updates.



Open Source vs Enterprise

Feature	Open Source	Enterprise
Core engine	✓	✓
Community support	✓	✓
SLA guarantees	×	✓
SSO / SAML	×	✓
Audit logging	×	✓
RBAC	×	✓
Priority fixes	×	✓
Dedicated support	×	✓



Where We Are Today

@layout metrics: big numbers with a source line

100%

Hardware pool utilization
Ke Holdings

10x

GPU utilization in CI
NIO

30%

Fewer GPU hours
China Merchants Bank

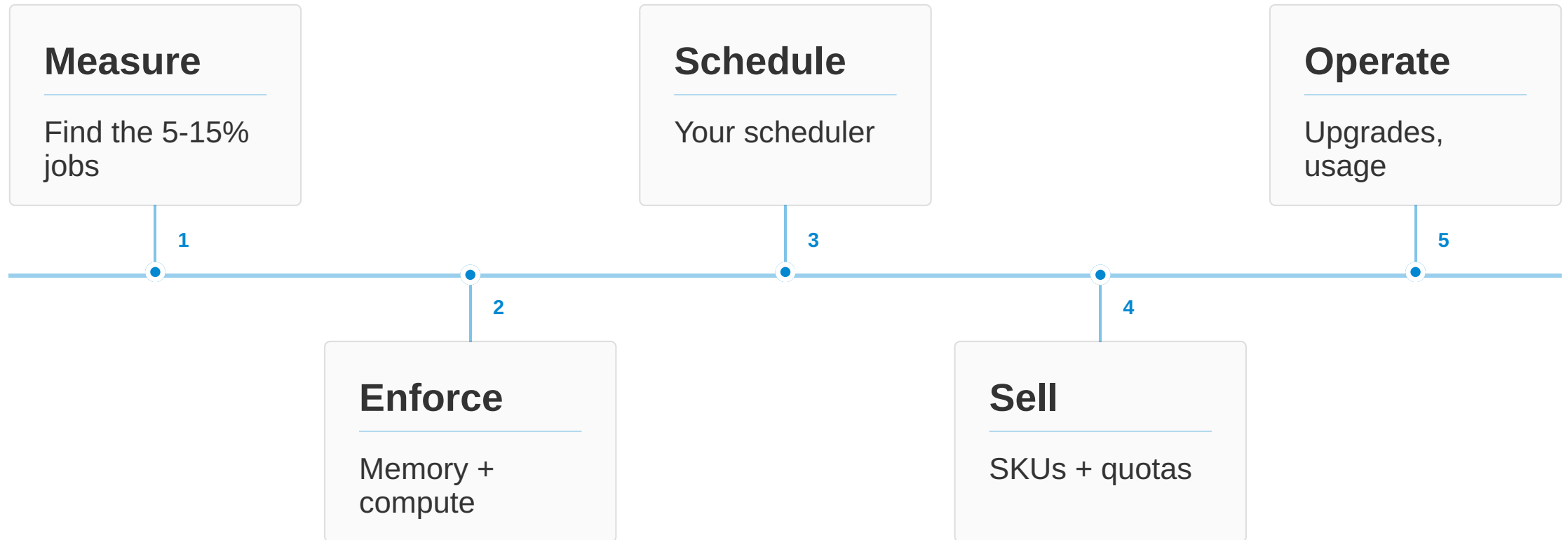
11

Device types
NVIDIA, Ascend,
Cambricon, ...



Rollout Plan

@layout timeline-horizontal with {label=...} cards; ::: notes underneath



Step zero: decide what stays whole. Big training jobs keep whole cards.



Community & Adopters

@layout ecosystem: ##### section labels over metric rows and logo cards

Open Source, CNCF Backed

4.1k

GitHub stars

325k

Docker pulls

500+

Contributors

Device Support



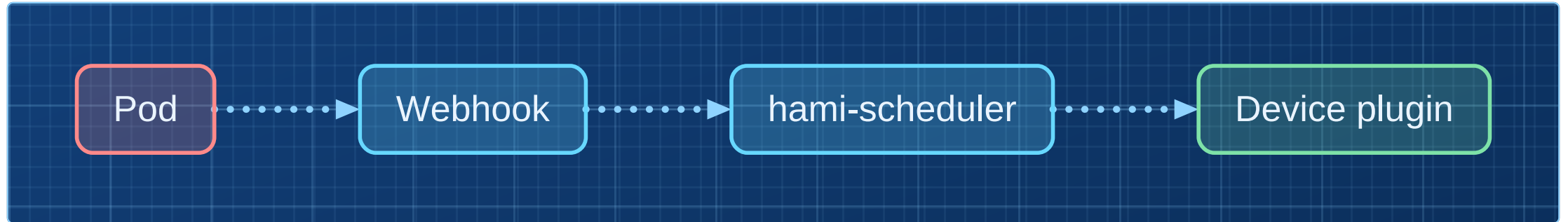
Getting Started

- ▶ Install the chart
- ▶ Label your GPU nodes
- ▶ Request `nvidia.com/gpumem` in a pod



Blueprint Diagram

::: card {blueprint} around a ```pdg diagram; this slide has @transition none





Config And Output

```
{  
  "nvidia.com/gpu": 1,  
  "nvidia.com/gpumem": 3000,  
  "nvidia.com/gpucore": 30  
}
```

```
$ nvidia-smi  
| 0 NVIDIA H200 3000MiB / 3000MiB 30% |
```



Embedding A Demo Video

@video plays in the HTML deck; PDF shows the poster image next to the clip

```
@video assets/demo/llm_test.mp4
```

- ▶ Put `llm_test.png` next to the clip as its poster and print image
- ▶ Presenter view keeps play, pause and seek in sync across windows
- ▶ ODP skips the video (add a screenshot slide if you need one there)



Talk Credits

@contributor sits beside @speaker; both take github, linkedin, twitter, email, website

Reza Jelveh

Solution Architect, Dynamia AI — Makers of HAMi

🔗 github.com/fishman 📧 @fishmanticoder in linkedin.com/in/rezajelveh

CONTRIBUTOR

Contributor Name

LFX Mentorship mentee

🔗 github.com/example in linkedin.com/in/example

QUESTIONS

 **Thank You**

Reza Jelveh
reza@dynamia.ai

