
dapr
HELM
TiKV
KEDA
KubeCon
CloudNativeCon
China 2025

KubeCon CloudNativeCon China 2025

Smart GPU Management: Dynamic pooling, Sharing, and Scheduling for AI workloads in Kubernetes

MengXuan Li
Wei Chen

About us

MengXuan Li

software engineer

- Github @archlitchi
- Dynamia.ai

Wei Chen

software engineer

- Github @sailorvii
- ChinaUnicom

You Will Learn

Part One

Opening & Problem Statement

Part Two

Existing alternatives

Part Three

Our solution

Part Four

Benifits & Results

PART 01

Opening & Problem Statement

Feature background

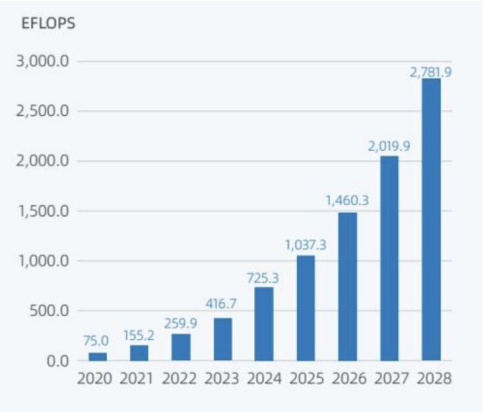


Figure 1:
Compute power requirement grows tremendously

- AI technology has entered the stage of commercialization and requires more and more computing power.
- The demand for computing power can be quite exaggerated with the emergence of LLM(i,e deepseek)
- In order to match the trend of computing power growth, GPU manufacturers have released new GPUs rapidly, with **more powerful computing power**, and **higher price**.

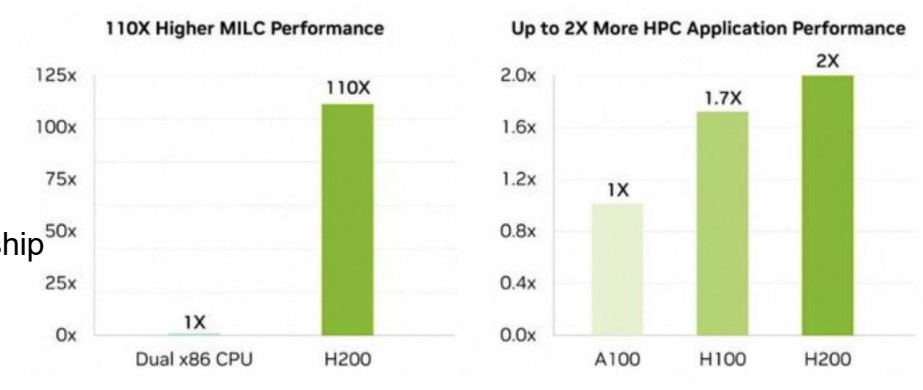
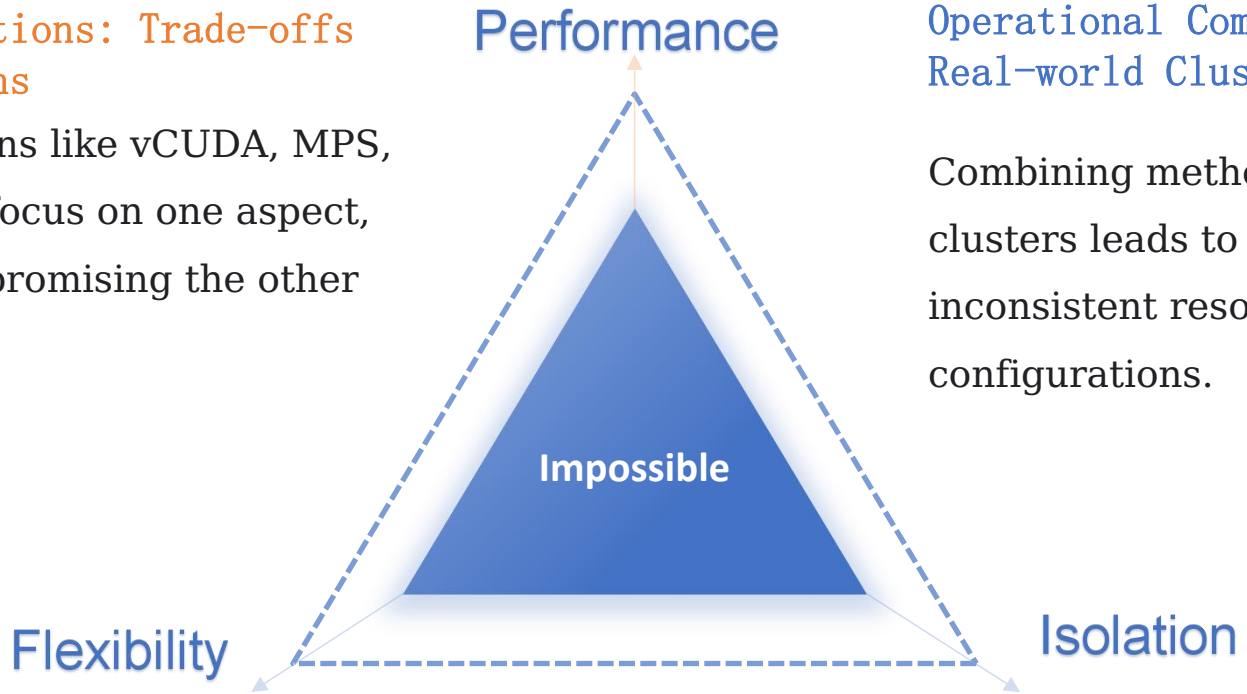


Figure 2:
NVIDIA Flagship GPU for ML

The GPU Sharing Dilemma: The impossible Trinity

Existing Solutions: Trade-offs and Limitations

Current solutions like vCUDA, MPS, and MIG each focus on one aspect, inevitably compromising the other two.



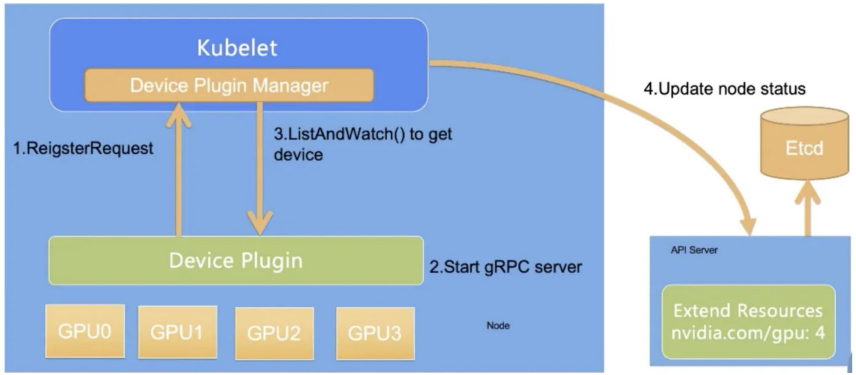
Operational Complexities in Real-world Clusters

Combining methods in real-world clusters leads to complexity due to inconsistent resource names and configurations.

Chanllange 1: Insufficient GPU usage

Two factors lead to low utilization of GPU devices in k8s clusters:

- GPU resources can only be applied by container in an **exclusive** manner
- In order to match the trend of computing power growth, GPU manufacturers have released new GPUs rapidly, with **more powerful computing power**, and **higher price.**



What you expect

```
|   5  Tesla V100-SXM2...   On   | 00000000:63:00.0 Off |                    0 |
| N/A   58C    P0   312W / 300W |  15429MiB / 16160MiB |    100%      Default |
|   6  Tesla V100-SXM2...   On   | 00000000:64:00.0 Off |                    0 |
| N/A   71C    P0   300W / 300W |  15429MiB / 16160MiB |    100%      Default |
```

What you actually get

```
|   1  Tesla V100-SXM2...   On   | 00000000:40:00.0 Off |                    0 |
| N/A   35C    P0    66W / 300W |   8707MiB / 16160MiB |      7%      Default |
|   2  Tesla V100-SXM2...   On   | 00000000:41:00.0 Off |                    0 |
| N/A   36C    P0    55W / 300W |   4663MiB / 16160MiB |      0%      Default |
```

A typical GPU utilization in GPU task in kubernetes:

- Most time core utilization can be < 10% .

Chanllange 2: Opensource solution are not compatible with Volcano

KubeCon CloudNativeCon China 2025

NVIDIA-k8s-device-plugin

https://github.com/NVIDIA/k8s-device-plugin

The NVIDIA device plugin allows oversubscription of GPUs through a set of extended options in its configuration file. There are two flavors of sharing available: Time-Slicing and MPS.

- Time-Slicing

Each workload has access to the GPU memory and runs in the same fault-domain as of all the others (**meaning if one workload crashes, they all do**)

- With CUDA MPS

All the clients running on the subset of GPUs in which the fatal fault is contained **will be affected** —— <NVIDIA MULTI-PROCESS SERVICE vR550>

Prject-HAMi

https://github.com/Project-HAMi/HAMi

Project HAMi can provide device-sharing support for a variety of heterogeneous computing devices(including nvidia GPUs) in k8s clusters. It ensures the isolation of resources inside containers.

It uses scheduler extender API provided by kube-scheduler, which is not compatible with Volcano

Chanllange 3: NVIDIA-MIG is inflexible to use

Config	GPC Slice #0	GPC Slice #1	GPC Slice #2	GPC Slice #3	GPC Slice #4	GPC Slice #5	GPC Slice #6	OFA	NVDEC	NVJPG	P2P	GPU Direct RDMA
1	7							1	5	1	No	
2	4				3			0	2+2	0	No	
3	4				2		1	0	2+1+0	0	No	
4	4				1	1	1	0	2+0+0+0	0	No	
5	3			3				0	2+2	0	No	
6	3			2		1		0	2+1+0	0	No	
7	3		1	1		1		0	2+0+0+0	0	No	
8	2		2		3			0	1+1+2	0	No	
9	2		1	1	3			0	1+0+0+2	0	No	
10	1	1	2		3			0	0+0+1+2	0	No	
11	1	1	1	1	3			0	0+0+0+0+2	0	No	
12	2		2		2		1	0	1+1+1+0	0	No	
13	2		1	1	2		1	0	1+0+0+1+0	0	No	
14	1	1	2		2		1	0	0+0+1+1+0	0	No	
15	2		1	1	1	1	1	0	1+0+0+0+0	0	No	
16	1	1	2		1	1	1	0	0+0+1+0+0+0	0	No	
17	1	1	1	1	2		1	0	0+0+0+0+1+0	0	No	
18	1	1	1	1	1	2		0	0+0+0+0+0+1	0	No	
19	1	1	1	1	1	1	1	0	0+0+0+0+0+0+0	0	No	

Supported MemBW proportional to size of the instance

```
$ nvidia-smi mig -lgip
+-----------------------------------------------------------------------------+
| GPU instance profiles:                                                      |
| GPU   Name             ID    Instances   Memory     P2P    SM    DEC   ENC  |
|                              Free/Total   GiB              CE    JPEG  OFA  |
|=============================================================================|
|   0  MIG 1g.5gb        19     7/7         4.75       No     14     0     0  |
|                                                              1     0     0  |
+-----------------------------------------------------------------------------+
|   0  MIG 1g.5gb+me     20     1/1         4.75       No     14     1     0  |
|                                                              1     1     1  |
+-----------------------------------------------------------------------------+
|   0  MIG 1g.10gb       15     4/4         9.62       No     14     1     0  |
|                                                              1     0     0  |
+-----------------------------------------------------------------------------+
|   0  MIG 2g.10gb       14     3/3         9.62       No     28     1     0  |
|                                                              2     0     0  |
+-----------------------------------------------------------------------------+
|   0  MIG 3g.20gb        9     2/2         19.50      No     42     2     0  |
|                                                              3     0     0  |
+-----------------------------------------------------------------------------+
|   0  MIG 4g.20gb        5     1/1         19.50      No     56     2     0  |
|                                                              4     0     0  |
+-----------------------------------------------------------------------------+
|   0  MIG 7g.40gb        0     1/1         39.25      No     98     5     0  |
|                                                              7     1     1  |
+-----------------------------------------------------------------------------+
```

NVIDIA MIG:

The new Multi-Instance GPU (MIG) feature allows GPUs to be securely partitioned into **up to seven separate GPU Instances** for CUDA applications. providing multiple users with separate GPU resources for optimal GPU utilization. This feature is particularly beneficial for workloads that do not fully saturate the GPU's compute capacity and therefore users may want to run different workloads in parallel to maximize utilization.

Features:

- Officially Supported
- Isolation guarantee

Limitations:

- Inflexible, need to preconfigure
- Different templates for different GPU types
- Needs to be specify mig-instance name when using
- *Only support Ampere or later Architecture*

PART 02

Existing alternatives

Solution 1: Volcano+nvidia-dra-driver

```yaml
apiVersion: resource.k8s.io/v1beta1
kind: DeviceClass
name: resource.example.com
spec:
  selectors:
  - cel:
      expression: device.driver == "resource-driver.example.com"
---
apiVersion: resource.k8s.io/v1beta1
kind: ResourceClaimTemplate
metadata:
  name: large-black-cat-claim-template
spec:
  spec:
    devices:
      requests:
      - name: req-0
        deviceClassName: resource.example.com
        selectors:
        - cel:
            expression: |-
              device.attributes["resource-driver.example.com"].color == "black"
&&
              device.attributes["resource-driver.example.com"].size == "large"
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-cats
spec:
  containers:
  - name: container0
    image: ubuntu:20.04
    command: ["sleep", "9999"]
    resources:
      claims:
      - name: cat-0
  - name: container1
    image: ubuntu:20.04
    command: ["sleep", "9999"]
    resources:
      claims:
      - name: cat-1
  resourceClaims:
  - name: cat-0
    resourceClaimTemplateName: large-black-cat-claim-
template
  - name: cat-1
    resourceClaimTemplateName: large-black-cat-claim-
template
```

Advantages:

- Official Device Share Capability
- No more conflicts between device vendors

Limitations:

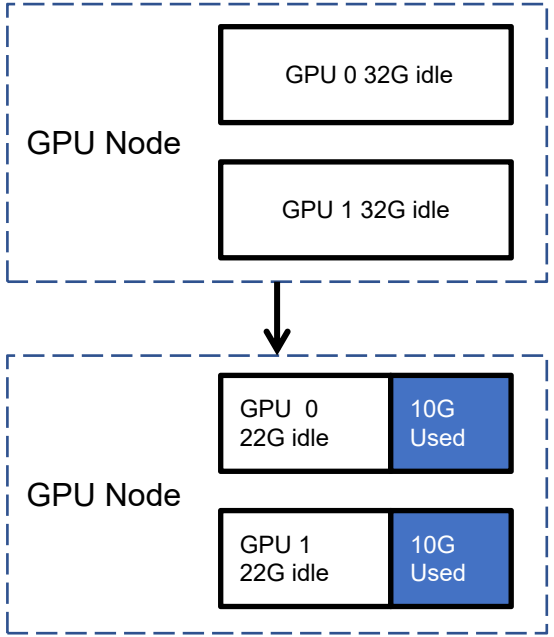
- Require high kubernetes version >= 1.32
- Needs to create resourceclaim and deviceclass
- Needs to be enabled explicitly
- Few device vendors implement dra-driver

Solution 2: Volcano-vgpu-device-plugin

A typical vGPU-task in kubernetes is shown below:

- Specify number of GPU mounted in container using 'volcano.sh/vgpu-number' resource name
- Specify available device memory for each GPU mounted using 'volcano.sh/vgpu-memory' resource name
- Set '.spec.schedulerName' to volcano

```
$ cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod12
spec:
  schedulerName: volcano
  containers:
    - name: ubuntu-container
      image: ubuntu:18.04
      command: ["bash", "-c", "sleep 86400"]
      resources:
        limits:
          volcano.sh/vgpu-number: 2 # requesting 2
vGPUs
          volcano.sh/vgpu-memory: 10240
```



Node status after submitting the vGPU-task

Solution 2: Volcano-vgpu-device-plugin

HAMi-Core

https://github.com/Project-HAMi/HAMi-core

```
root@gpu-pod:/# nvidia-smi
[4pdvGPU Msg(18:139914332944192:libvgpu.c:869)]: Initializing.....
Fri Jan 12 10:20:10 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 515.65.01    Driver Version: 515.65.01    CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-PCIE...  Off  | 00000000:3E:00.0 Off |                    0 |
| N/A   30C    P0    24W / 250W |      0MiB /  3000MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

HAMi-Core uses symbolic hijacking to intercept CUDA calls, enabling core and memory limitation as well as isolation within containers.

Prerequisites:

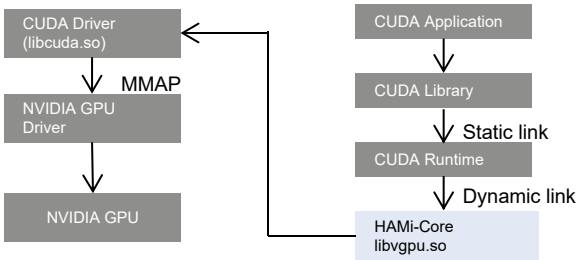
- Nvidia driver version >= 440
- CUDA version >= 10.2

Features:

- ✓ Device Memory isolation
- ✓ Core utilization limitation
- ✓ Fault isolation
- ✓ Transparent to GPU tasks

Limitations:

- Not NVIDIA official



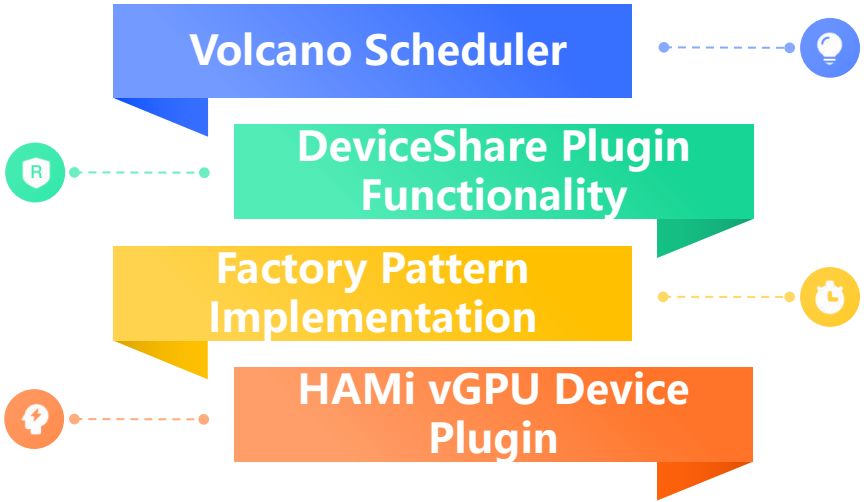
PART 03

Our Solution

Core Components of Volcano GPU Sharing

The DeviceShare Plugin extends Volcano Scheduler's capabilities, facilitating the sharing of GPU devices across multiple users and applications.

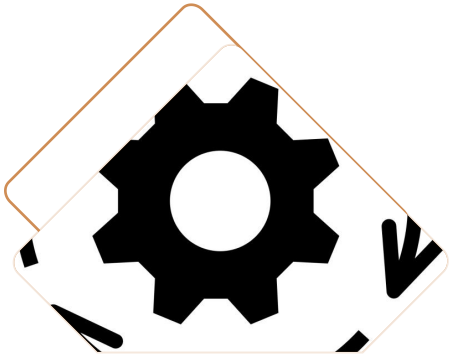
HAMi vGPU device plugin enables virtualized GPU access, allowing multiple processes to share physical GPU resources concurrently.

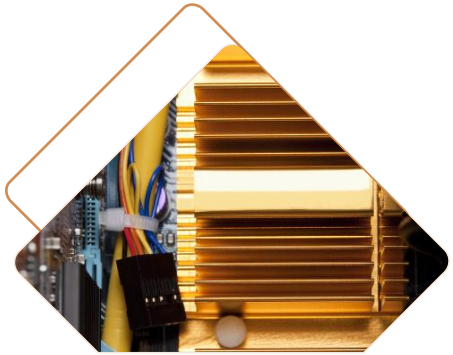


Volcano Scheduler orchestrates GPU resources, enabling efficient allocation and management through its DeviceShare Plugin.

Factory Pattern implementation provides a scalable approach to creating instances for various GPU sharing technologies, ensuring flexibility and maintainability.

Supported Sharing Modes





HAMi-core: Dynamic Sharing Mechanism

HAMi-core employs a dynamic sharing mechanism using a CUDA-hacking library, optimizing GPU utilization by allowing concurrent execution of diverse workloads.

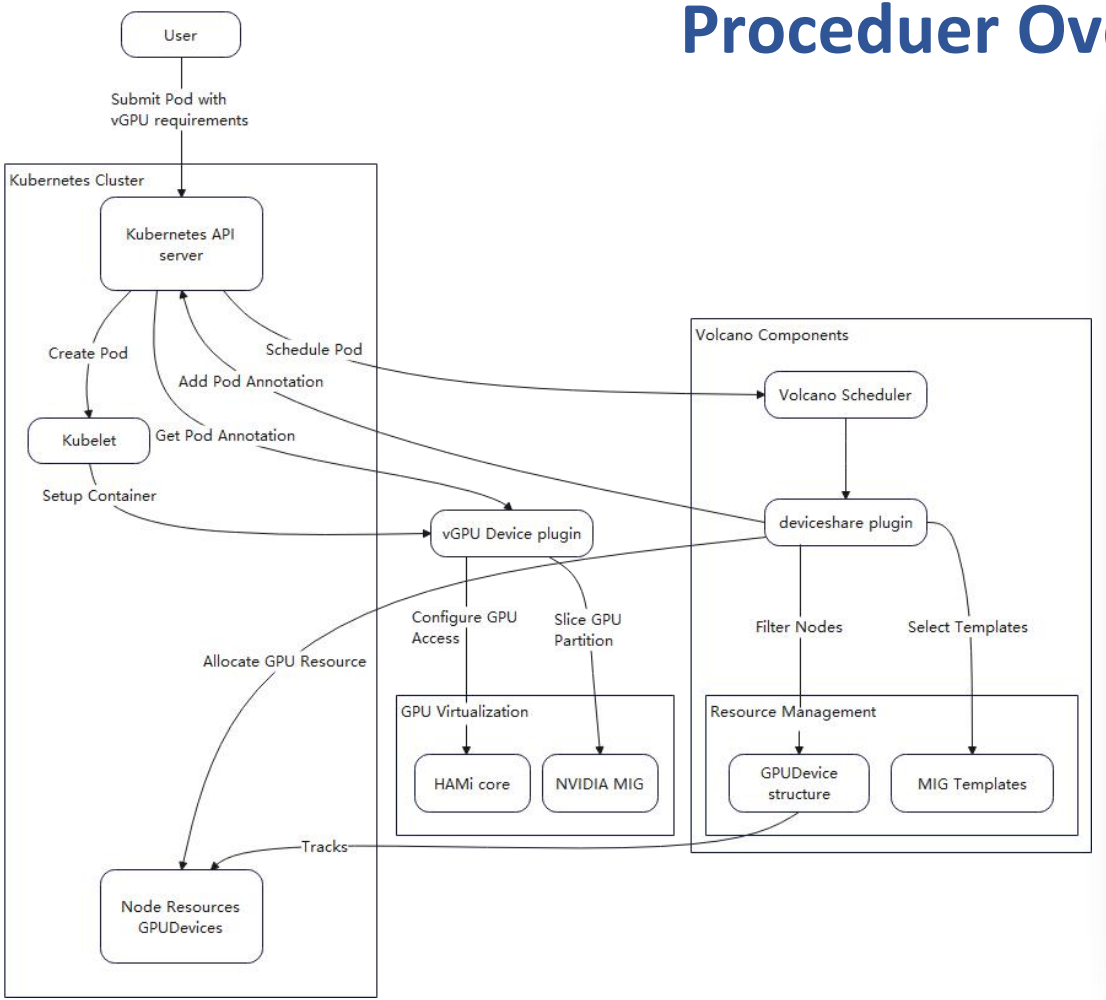
MIG: Hardware Partitioning Details

With MIG, the GPU is hardware-partitioned into smaller, isolated instances, each capable of running separate tasks simultaneously, enhancing security and efficiency.

MPS: Multi-Process Service Overview

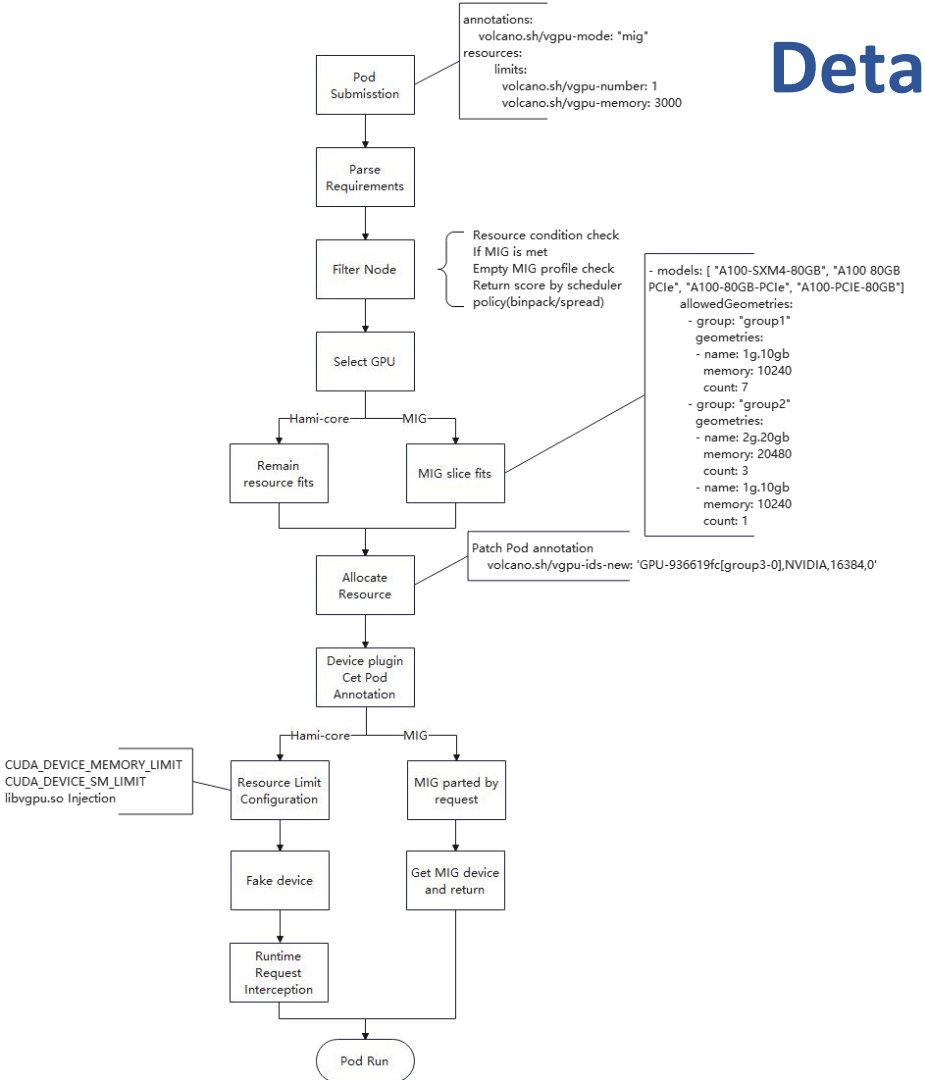
MPS provides a shared context for multiple processes to execute on a single GPU, streamlining the process and reducing context switching overhead.

Proceduer Overview



- Pod Submission: A user submits a pod requesting vGPU resources with specific requirements.
- Scheduler Processing: The volcano scheduler processes the pod request through its deviceshare plugin.
- MIG Mode Selection: The pod can explicitly request MIG mode through annotations.
- Template Selection: The deviceshare plugin iterates over MIG templates defined in the configMap to find a suitable geometry.
- MIG Instance Creation: When a suitable template is found, MIG instances are created according to the template.
- Instance Allocation: The created MIG instances are allocated to the pod.
- Container Startup: The container starts with the allocated MIG instances.

Detailed Process



- Pod Submission: User specify the annotation “volcano.sh/vgpu-mode” to choose sharing mode. If not set, scheduler will choose by Node score.

- MIG profile definition: the configMap is shared by Volcano and HAMi device plugin. The group is in order. First match is selected.

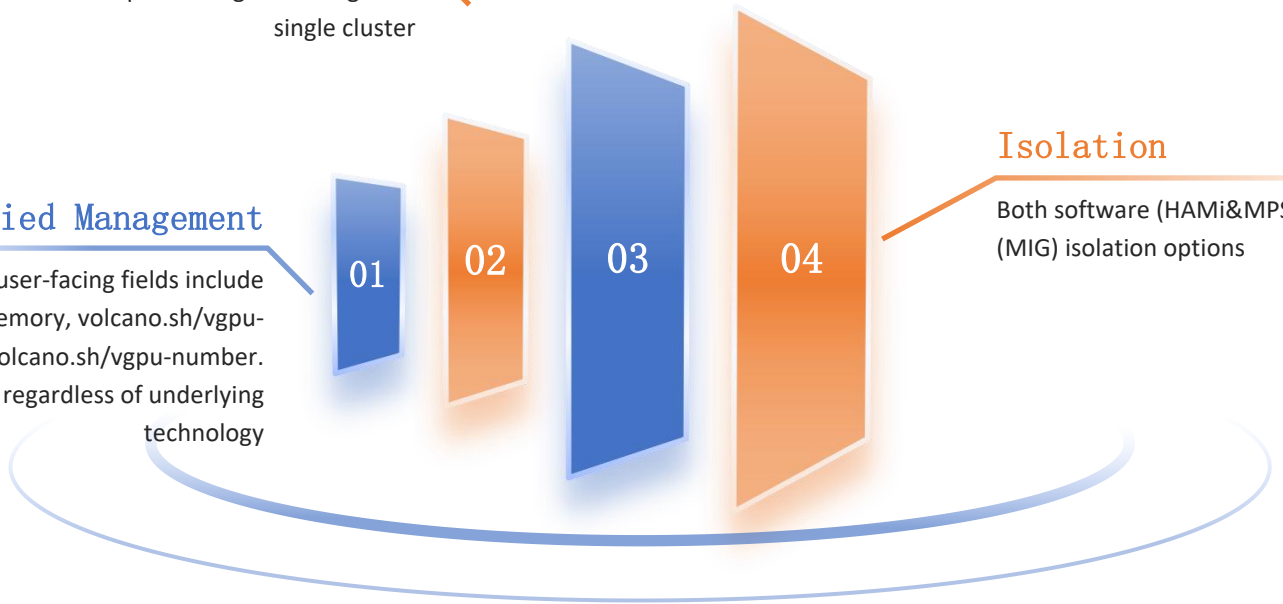
- Resource Allocation: device plugin fake/create device by Pod Annotation “volcano.sh/vgpu-ids-new” .

- HAMi core Interception: By configuring Pod runtime ENV and injecting HAMi core library, validate requests and enfore memory and compute core restrictions

PART 04

Benefits & Results

Solving the Impossible Trinity



Demo illustration

- HAMi-core pod request
- MIG pod request
- Binpack & spread scheduling
- Multi-pod without dedicated shaing mode

Future Work

2025.10

Support dynamic-MPS

01

02

2025.12

Support volcano-dashboard

2026.3

Dynamic adjustment on vGPU jobs

03

04

2026.6

Support device sharing for 3+ AI cards besides NVIDIA

Join us

Projects

Project HAMi
https://github.com/Project-HAMi/HAMi

Website
https://project-hami.io/

Volcacno
https://github.com/volcano-sh/volcano

Website
https://volcano.sh/en/