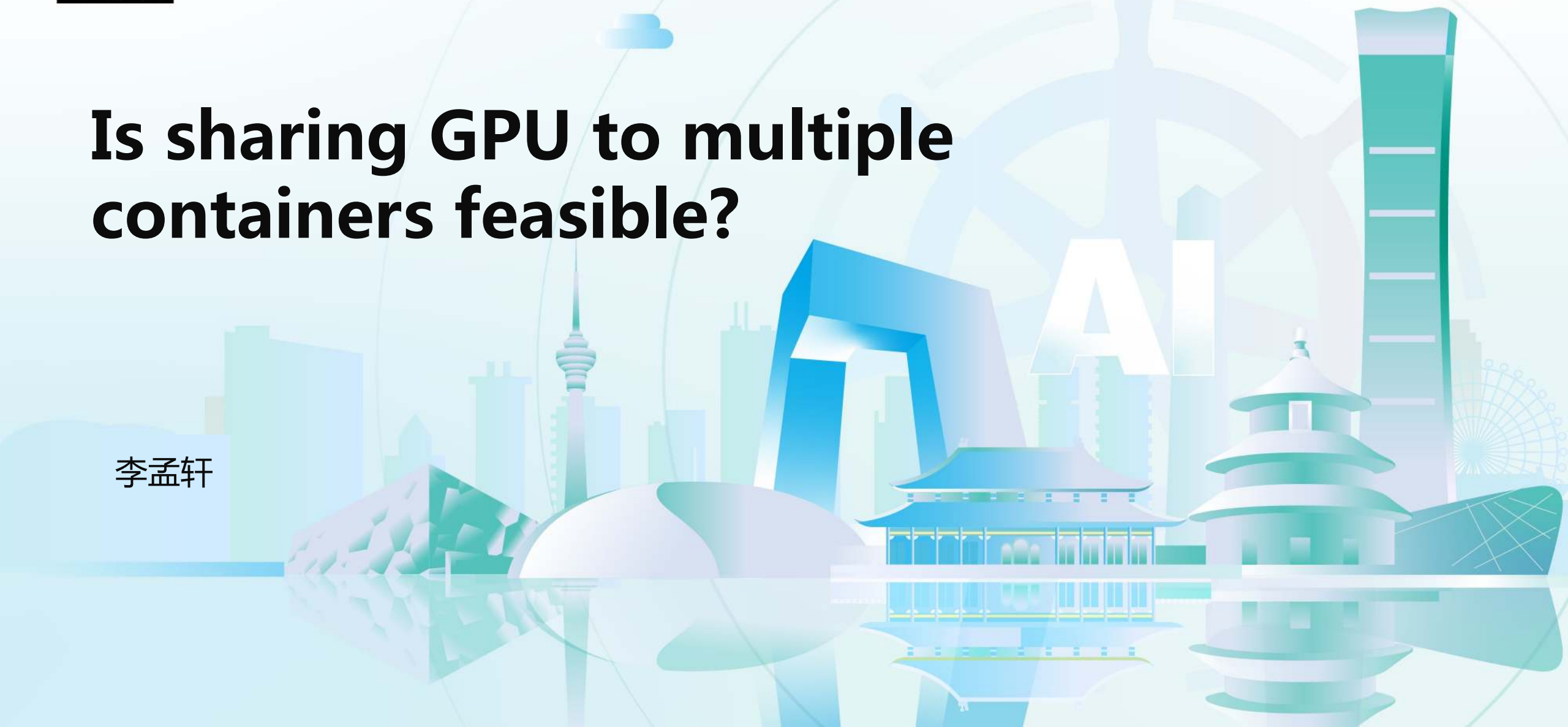
KUBERNETES COMMUNITY DAYS BEIJING 2025

# Is sharing GPU to multiple containers feasible?

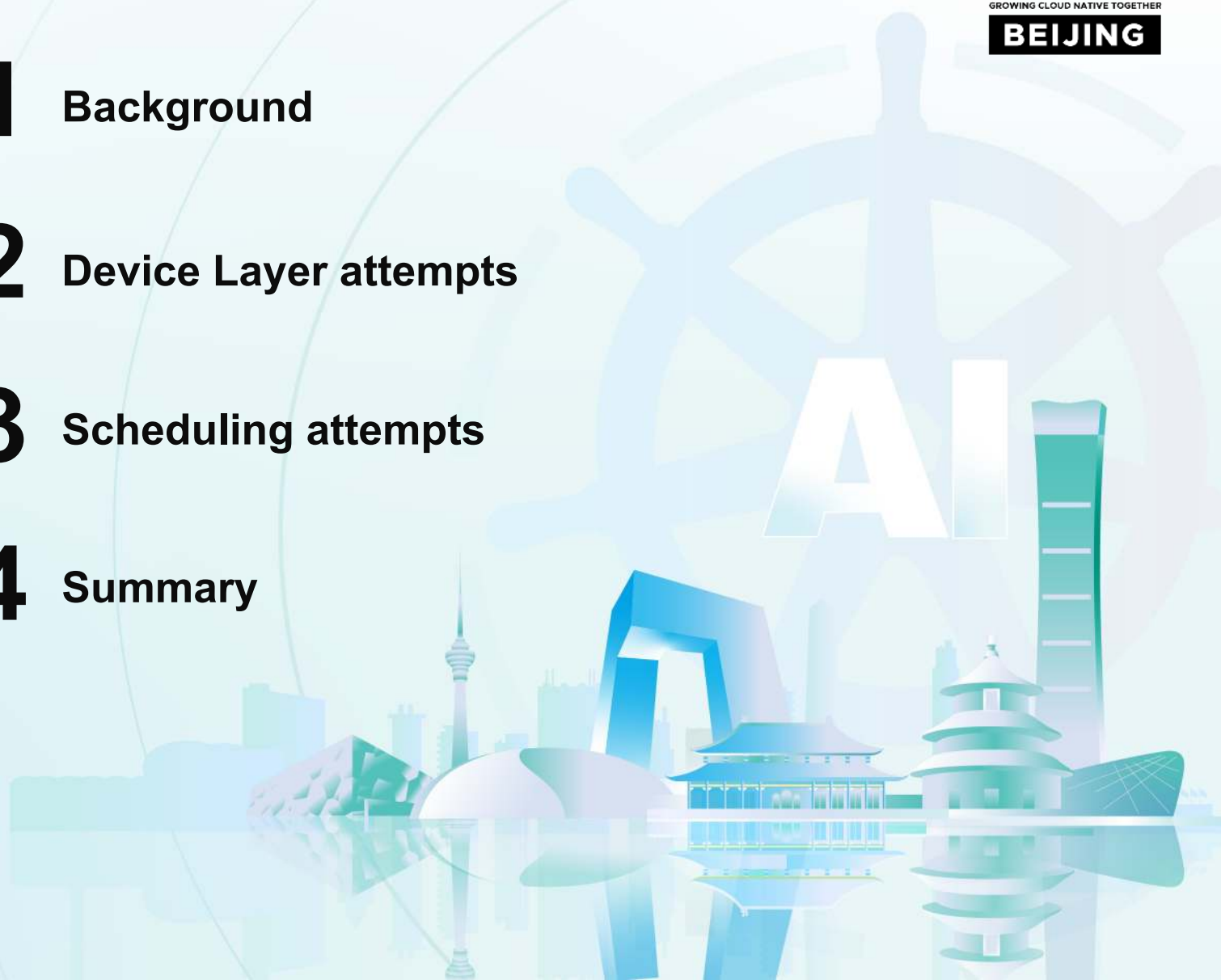


李孟轩

# Content

# 目录

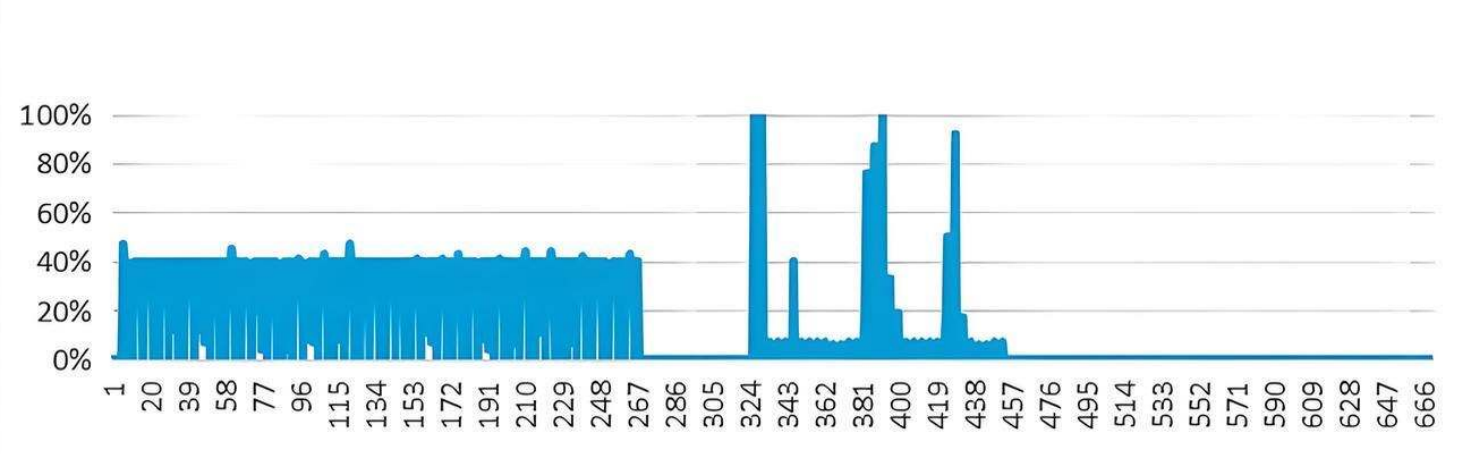




# Part 01
# Background:

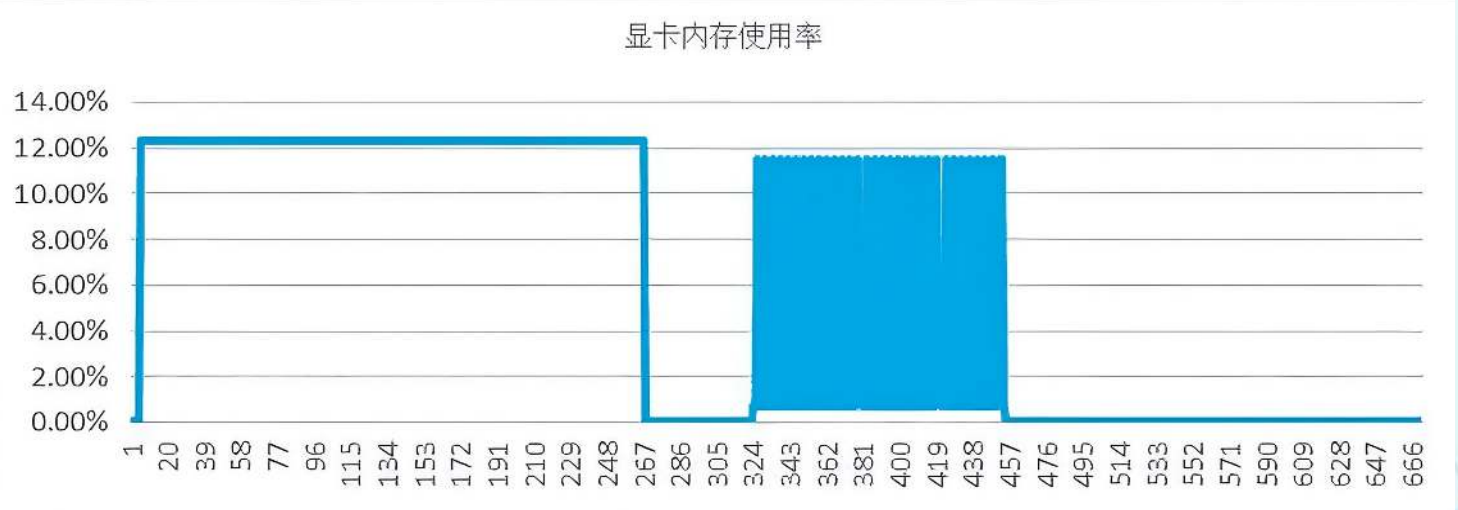
Device can’t be fully utilized

# A typical GPU utilization in production environment




*A typical GPU utilization in GPU task in kubernetes：*

- Core utilization can be 0 for a long period of time
- 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**.




*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**.

# Issue #52757

## Is sharing GPU to multiple containers feasible? #52757

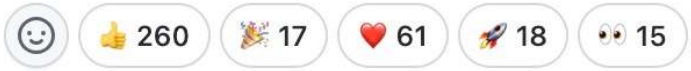
Open

tianshapjq opened on Sep 20, 2017

**Is this a BUG REPORT or FEATURE REQUEST?**: feature request
/kind feature

**What happened**:
As far, we do not support sharing GPU to multiple containers, one GPU can only be assigned to one container at a time. But we do have some requirements on achieving this, is it feasible that we manage GPU just like CPU or memory?

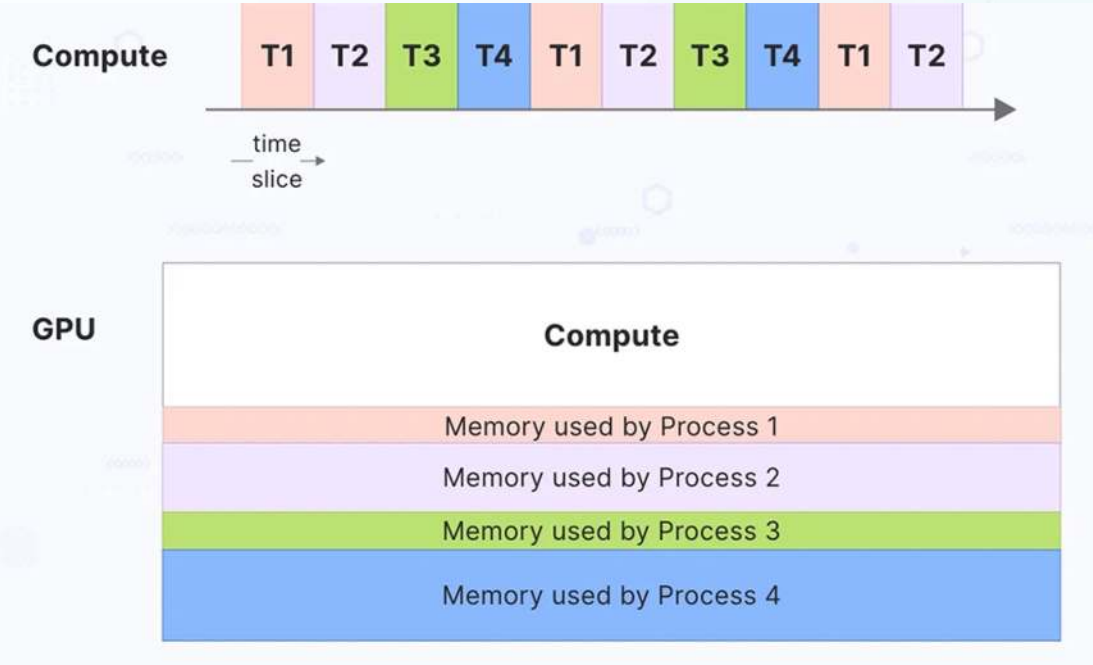
**What you expected to happen**:
sharing GPU to multiple containers just like CPU and memory.



# Part 02
# Device layer Attempts:

How to construct a GPU-resource sandox?

# Nvidia TimeSlice

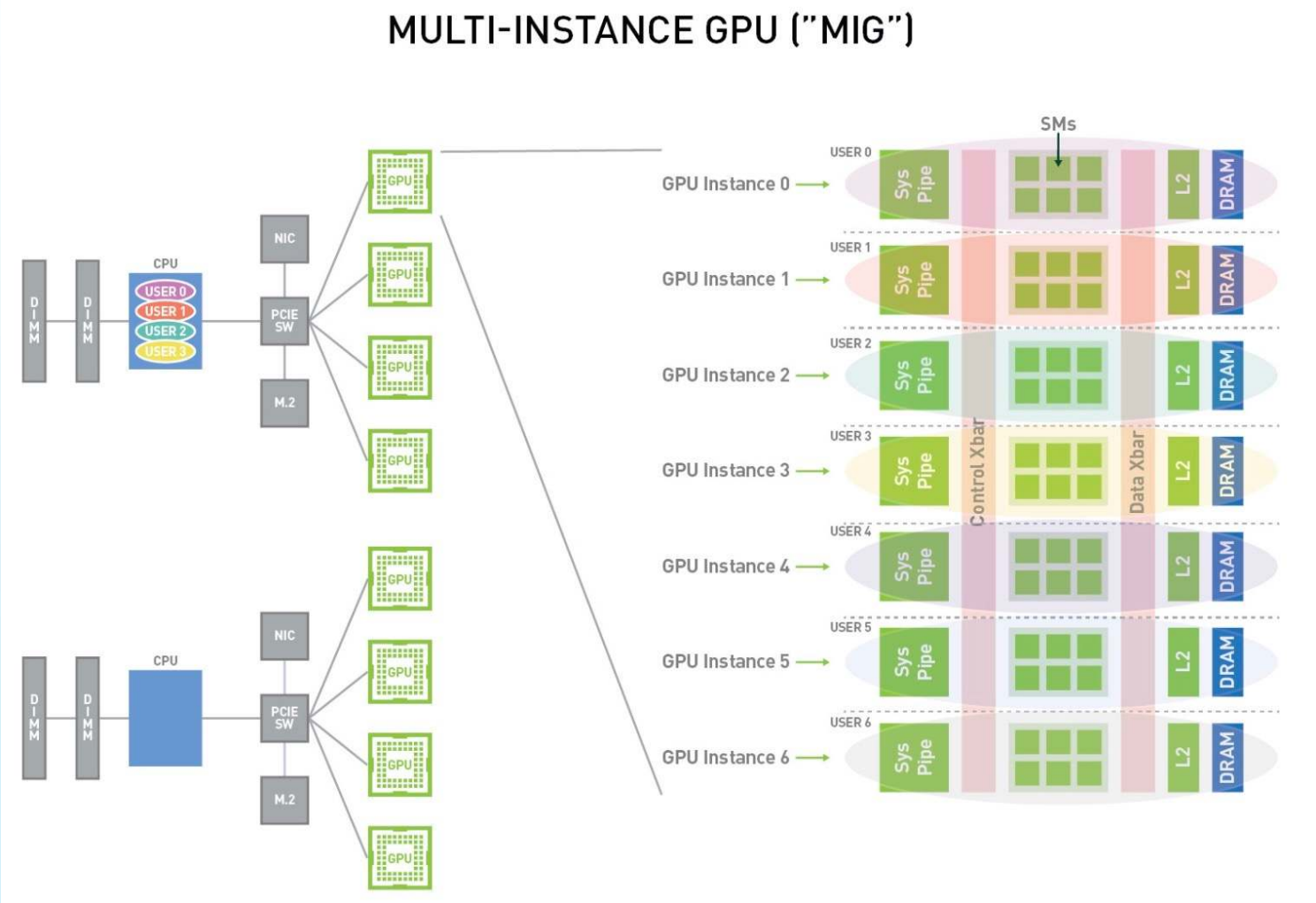



```
apiVersion: v1
kind: ConfigMap
metadata:
  name: time-slicing-config-all
data:
  any: |-
    version: v1
    flags:
      migStrategy: none
    sharing:
      timeSlicing:
        renameByDefault: false
        failRequestsGreaterThanOne: false
        resources:
          - name: nvidia.com/gpu
            replicas: 4
```

*Nvidia Time-slice is like put multiple containers directly into that GPU:*

- No resource control
- No Overhead

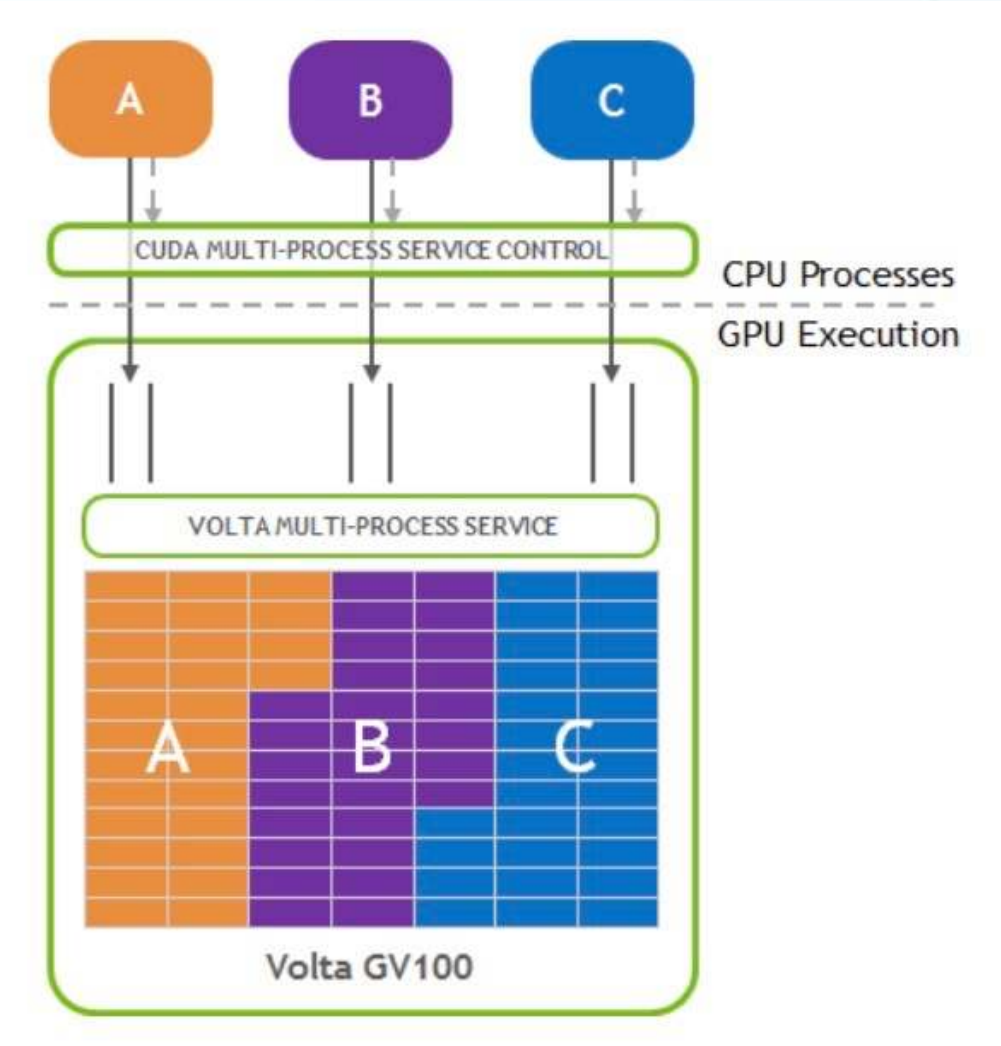
# Nvidia MIG




*Nvidia MIG splits a GPU into serveral MIG-instances:*

- Resource Isolation gurantee
- Low Overhead
- Only apply for ampere or later GPUs
- Device memory and compute-core are cut simultaneously
- Have to follow certain template
- Hard to configure dynamically in kubernetes

# Nvidia MPS

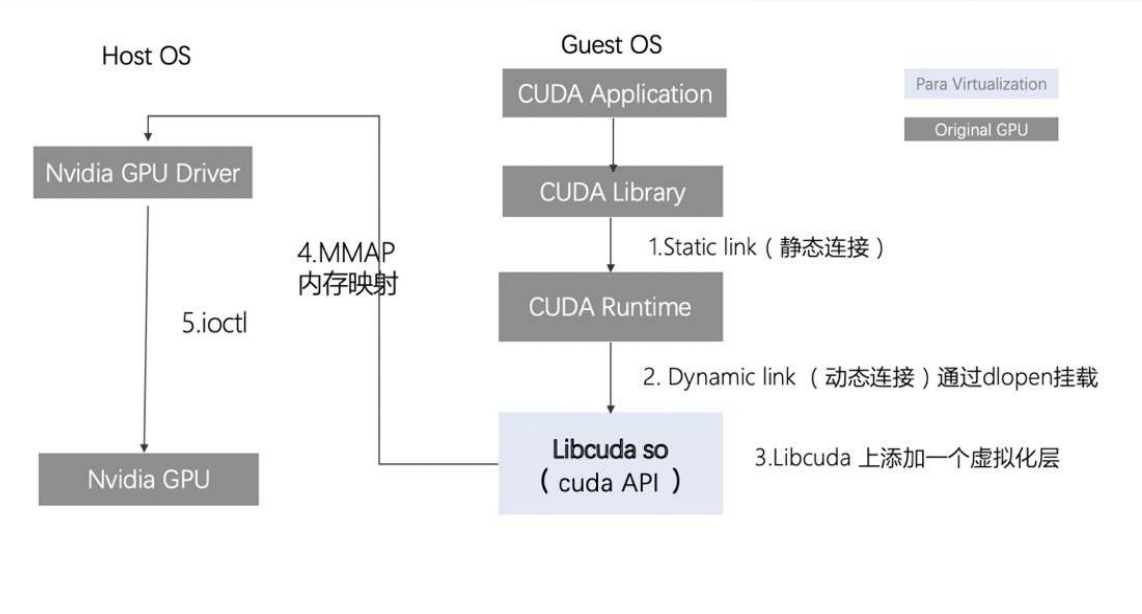



*Nvidia MPS smashes tasks from multiple containers into a single context, which brings high-performace, but high-risks:*

- Resource Isolation gurantee
- High performance
- High risk for task failure
- Hard to configure inside kubernetes

```
version: v1
sharing:
  mps:
    renameByDefault: true
    resources:
    - name: nvidia.com/gpu
      replicas: 10
    ...
```

# HAMi-Core



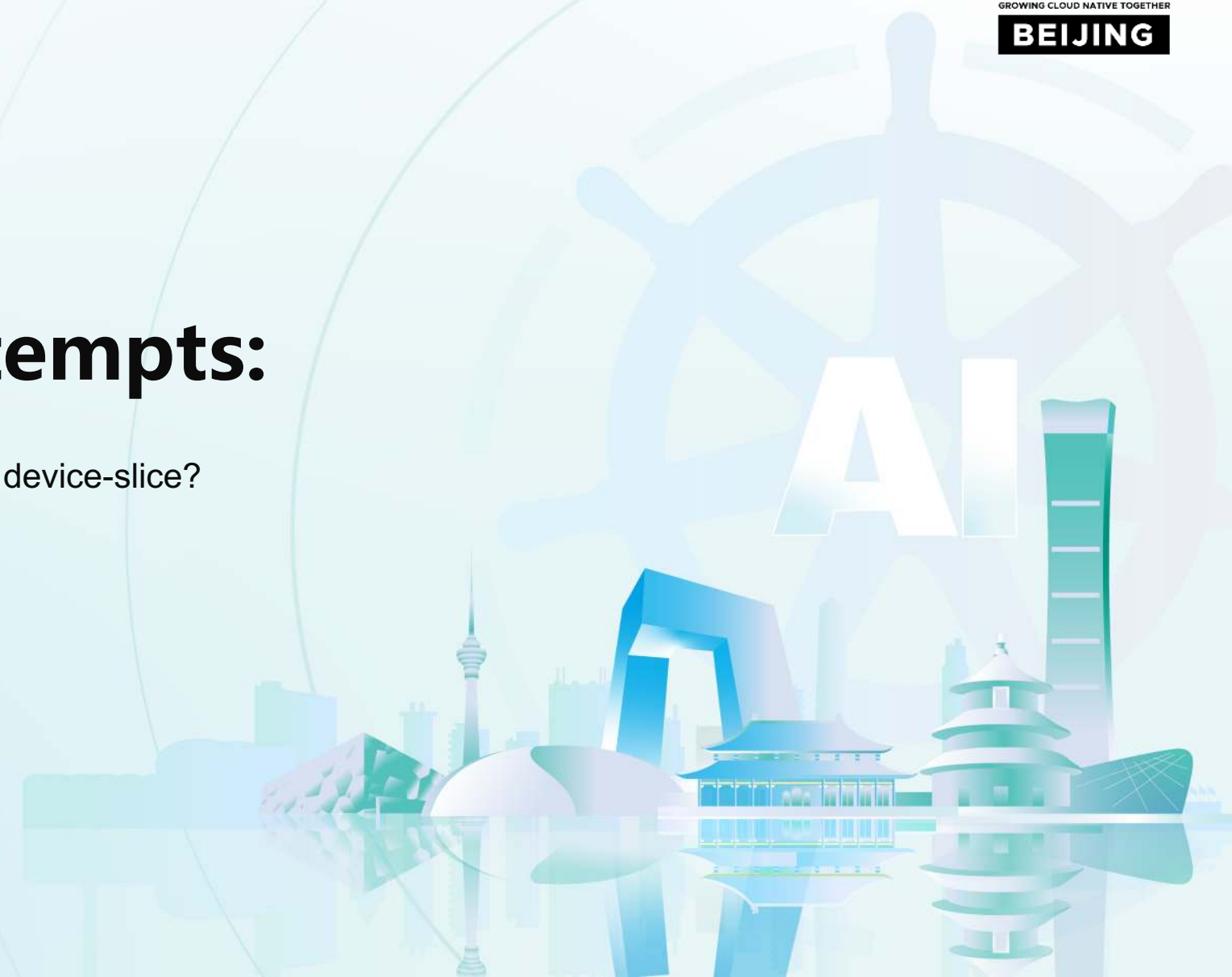

```
> docker run ${CUDA_ARGS}   -e CUDA_DEVICE_MEMORY_LIMIT=518m  cuda_override:tf1.8-cu90  nvidia-smi
Mon May  6 04:41:40 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 410.73       Driver Version: 410.73       CUDA Version: 10.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 108...  Off  | 00000000:05:00.0 Off |                  N/A |
| 21%   40C    P0    54W / 250W |      0MiB /   518MiB |      1%      Default |
+-------------------------------+----------------------+----------------------+

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

*HAMi-core is a third-party resource controller for NVIDIA GPUs inside container*

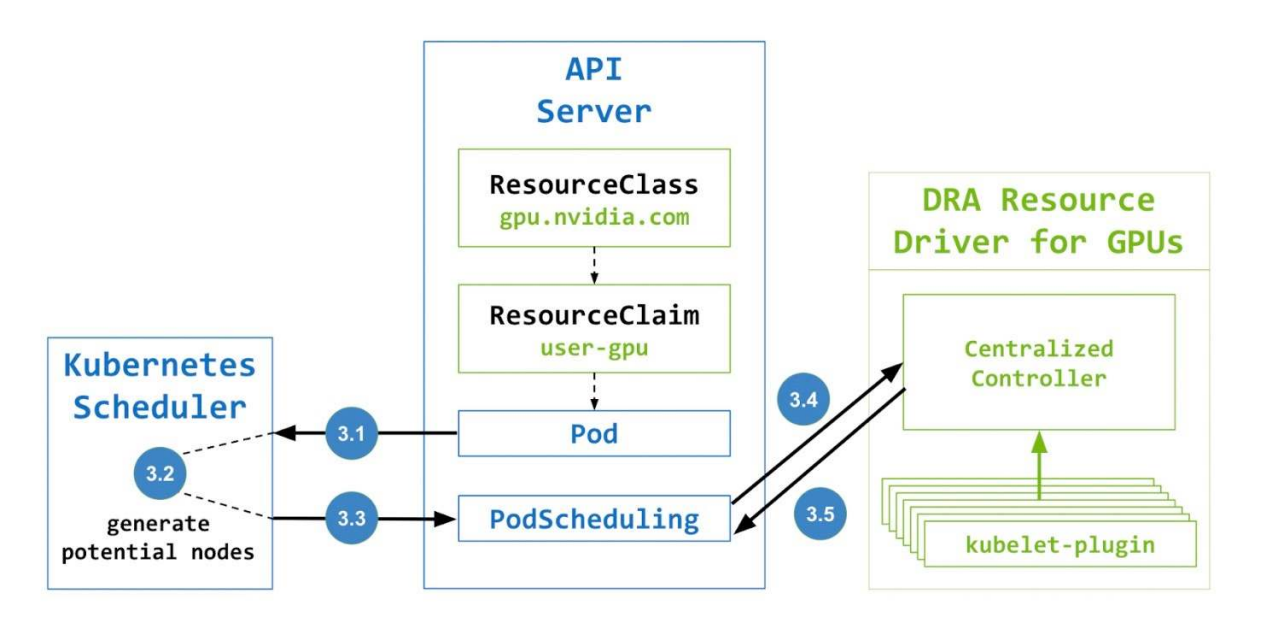
- Resource Isolation gurantee
- Low overhead
- Failure Isolation
- Easy to integrate

# Part 03
# Scheduling Attempts:

How does scheduler cooperate with device-slice?



# Nvidia DRA: https://github.com/NVIDIA/k8s-device-plugin




*Dynamic Resource Allocation (DRA) is an upcoming Kubernetes feature that puts resource scheduling in the hands of 3rd-party developers. providing an API more akin to that of persistent volumes. Under the hood it uses CDI to do its device injection.*

- Official
- Only works on kubernetes v1.26+
- Not easy to configure

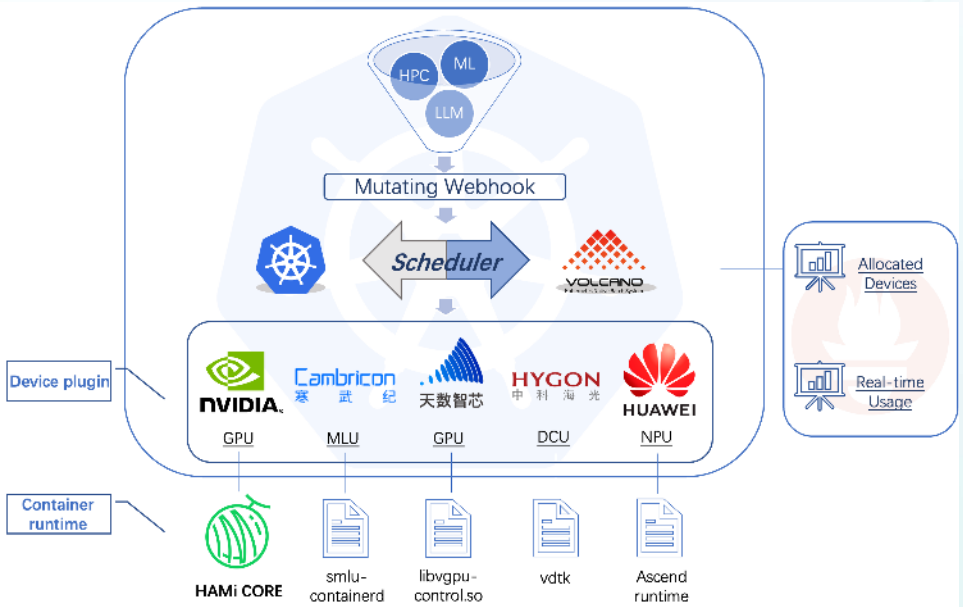
**GPU Sharing within a Pod**

```
---
apiVersion: resource.k8s.io/v1alpha1
kind: ResourceClaimTemplate
metadata:
  name: gpu-template
spec:
  spec:
    resourceClassName: gpu.nvidia.com

---
apiVersion: v1
kind: Pod
metadata:
  name: pod
spec:
  containers:
  - name: ctr0
    image: nvidia/cuda
    command: ["nvidia-smi" "-L"]
    resources:
      claims:
      - name: gpu
  - name: ctr1
    image: nvidia/cuda
    command: ["nvidia-smi" "-L"]
    resources:
      claims:
      - name: gpu
  resourceClaims:
  - name: gpu
    source:
      resourceClaimTemplate: gpu-template
```

**GPU Sharing across Pods**

```
---
apiVersion: resource.k8s.io/v1alpha1
kind: ResourceClaim
metadata:
  name: shared-gpu
spec:
  resourceClassName: gpu.nvidia.com

---
apiVersion: v1
kind: Pod
metadata:
  name: pod0
spec:
  containers:
  - name: ctr
    image: nvidia/cuda
    command: ["nvidia-smi" "-L"]
    resources:
      claims:
      - name: gpu
  resourceClaims:
  - name: gpu
    source:
      resourceClaimName: shared-gpu

---
apiVersion: v1
kind: Pod
metadata:
  name: pod1
spec:
  containers:
  - name: ctr
    image: nvidia/cuda
    command: ["nvidia-smi" "-L"]
    resources:
      claims:
      - name: gpu
  resourceClaims:
  - name: gpu
    source:
      resourceClaimName: shared-gpu
```

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



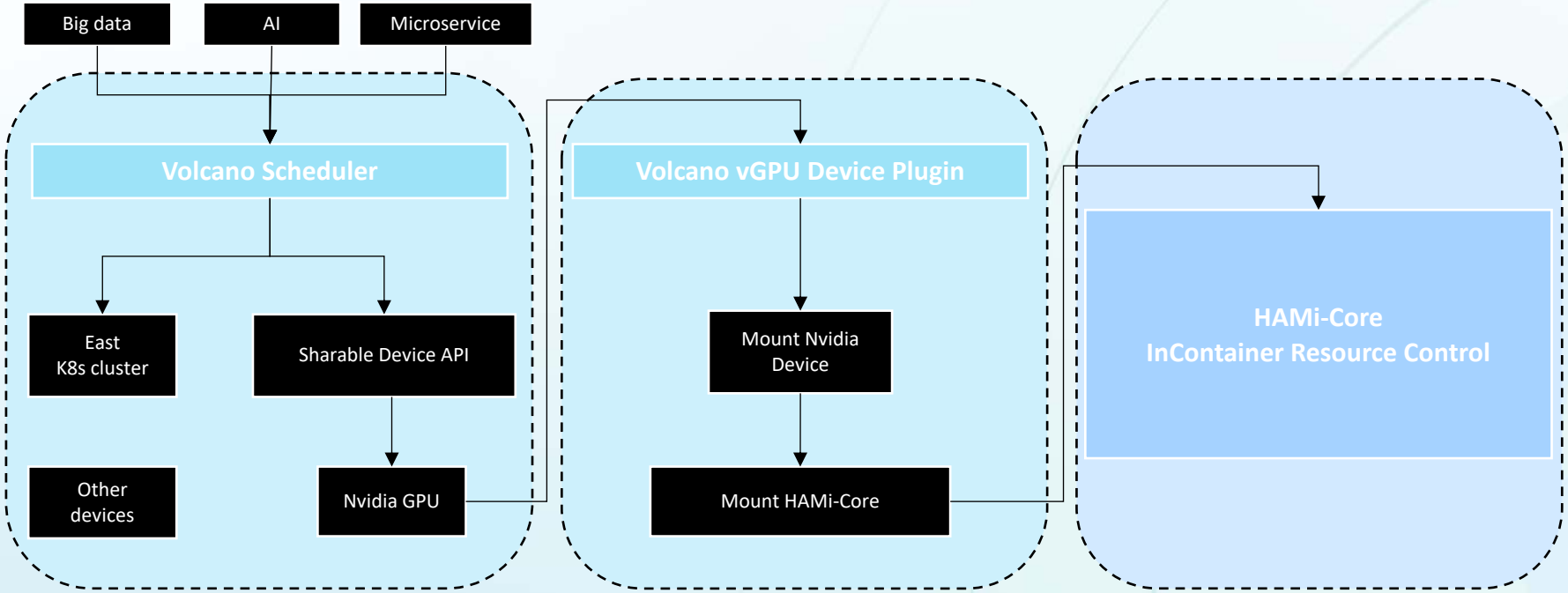

```
root@gpu-demo-6b6b88b75b-x9tjb:~# nvidia-smi
[HAMI-core Msg(35:140111864956736:libvgpu.c:836)]: Initializing.....
Thu Apr 18 06:22:54 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.12             Driver Version: 535.104.12   CUDA Version: 12.2     |
|-------------------------------+----------------------+----------------------+
| 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  NVIDIA A800 80GB PCIe          On  | 00000000:13:00.0 Off |                    0 |
| N/A   36C    P0              81W / 300W |      0MiB / 10240MiB |      0%      Default |
|                                         |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA A800 80GB PCIe          On  | 00000000:1C:00.0 Off |                    0 |
| N/A   39C    P0              82W / 300W |      0MiB / 10240MiB |      0%      Default |
|                                         |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
[HAMI-core Msg(35:140111864956736:multiprocess_memory_limit.c:434)]: Calling exit handler 35
root@gpu-demo-6b6b88b75b-x9tjb:~#
```

```
resources:
  limits:
    nvidia.com/gpu: 2 # requesting 1 vGPUs
    nvidia.com/gpumem: 10240
    nvidia.com/gpucores: 30
```

*HAMi is an 'all-in-one' chart designed to manage Heterogeneous AI Computing Devices in a k8s cluster. It can provide the ability to share Heterogeneous AI devices and provide resource isolation among tasks.*

- GPU Sharing
- Task Schedule Optimization
- Support Multiple heterogeneous devices

# Volcano-vgpu: https://github.com/Project-HAMi/volcano-vgpu-device-plugin




```
resources:
  limits:
    volcano.sh/vgpu-number: 2 # requesting 2 vGPUs
    volcano.sh/vgpu-memory: 10240
    volcano.sh/vgpu-cores: 30
```

*Project-HAMi donates its GPU resource isolation components(HAMI-Core) to volcano, the whole architect is shown as the figure：*

- At the scheduling layer, the Volcano scheduler is responsible for **assigning tasks to appropriate nodes**
- At the device layer, the volcano device plugin is responsible for **mounting the corresponding GPU device and HAMi-Core**
- At the container layer it uses hami-core to control device resources

# Part 04
# Summary

Which is the best practice?

# Summary

| Key features | HAMi vgpu/ volcano-vgpu | CUDA Streams | MPS | Time-slicing | MIG | Nvidia vGPU |
|---|---|---|---|---|---|---|
| **Target Use Cases** | The same cluster contains multiple heterogeneous AI devices+ Gpu sharing + flexible scheduler policies | Optimized for concurren cywithin a single application | When running multipleapplicatio ns in parallel butcan deal with limitedresiliency | When running multipleapplic ations that are notlatency-sensitive or cantolerate jitter | When running multipleapplicat ions in parallel butneed resiliencyand QoS | When needing to supportmulti-tenancy on the GPUthrough virtualization |
| **Partition Type** | Logical | Single Process | Logical | Temporal | Physical | Temporal & Physical (VM) |
| **Max Partitions** | Unlimited | Unlimited | 48 | Unlimited | 7 | Variable |
| **SM Performance Isolation** | Yes(by % not per client) | No | Yes(by % not per client) | Yes | Yes | Yes |
| **Memory Protection** | Yes | No | Yes | Yes | Yes | Yes |
| **Memory Bandwidth QoS** | No | No | No | No | Yes | Yes |
| **Error Isolation** | Yes | No | No | Yes | Yes | Yes |
| **Cross-PartitionInteroper ability** | | Always | IPC | Limited IPC | Limited IPC | No |
| **Reconfiguration** | At process Launch | Dynamic | At process Launch | Time-Slice Duration Only | When Idle | No |
| **Telemetry** | Yes | No | Limited | No | Yes(including in containers) | Yes(including live migration) |
| **Other noteworthy** | Supports all GPUs, open source | | cudaCapability >= 3.5 | cudaCapability >= 7.0 | cuda capability >= 8.0 Hopper,Ampere | license required |

# Thanks.

Project HAMi

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

*Thank for donating HAMi-Core, which is vital for in-container resource control*

Volcacno-vgpu

*https://github.com/Project-HAMi/volcano-vgpu-device-plugin*