geminiopencloud

以 GPU 分割共享技術，
優化 Kubernetes 與 GPU 資源利用率

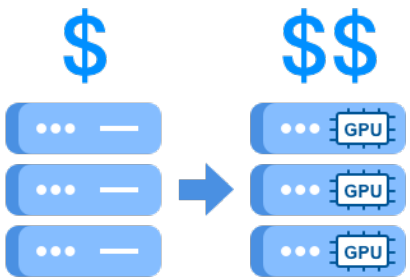
Edison Peng

geminiopencloud

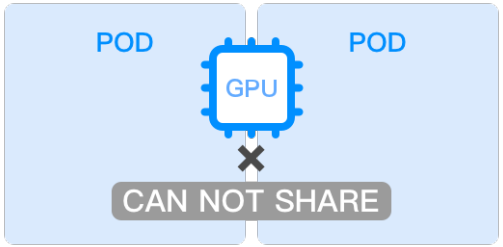
Agenda

- 在 Kubernetes 中使用 GPU 的痛點與挑戰
- GPU 分割技術 設計

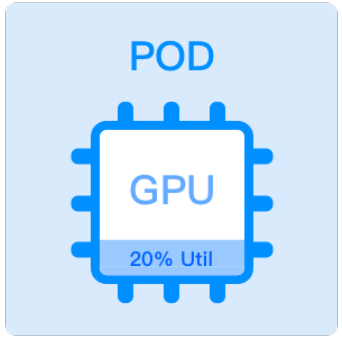
在 Kubernetes 中使用 GPU 的痛點與挑戰



GPU 的成本就佔據軟硬體設備的一半以上



僅支援排他獨占的 GPU 分配

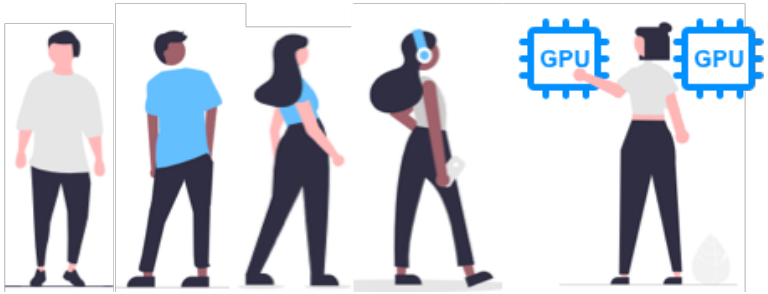


GPU 資源無法被充分利用

從前 資料科學家如何利用珍貴的GPU？

預約

排隊



預約：調度不夠彈性

預約，就必須事先評估並搶佔使用權限.....

我得先去行事曆預約，哪一天我要使用 GPU 伺服器。讓別人不要用。

萬一臨時有事，我又要重新預約，擔誤我的時間。臨時空下來的設備，也很可能來不及讓給別人用。

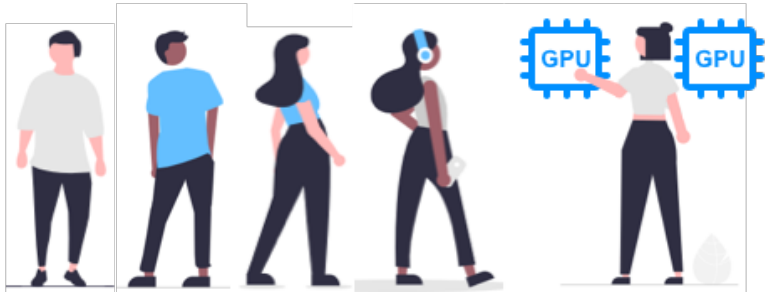
萬一我的工作太多，那一天之內處理不完，不論是強制中斷我的工作，或者拖延到下一天別人的工作，大家都很不便。

排隊：不知何時輪到我，等待時間長

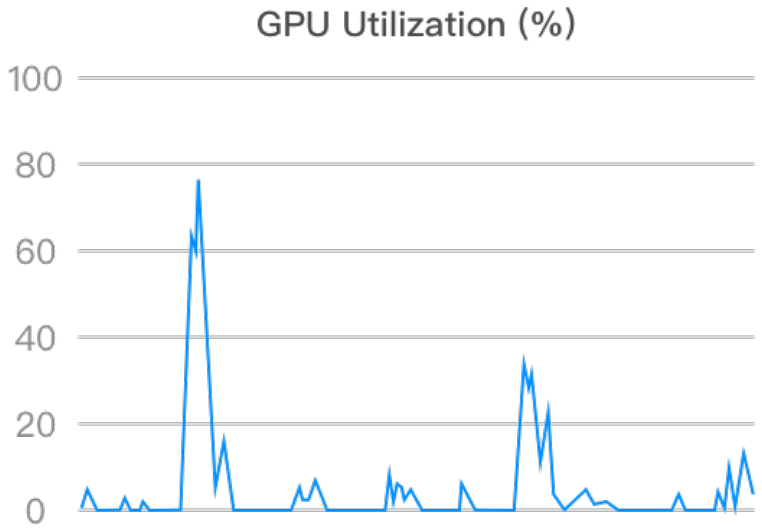
排隊，就必須等待前面的專案或容器結束。

我不知道前面的人還要多久，只能一直等待，不知道什麼時候輪到我，也不知道什麼時候我的工作能處理完。

如果前面的人在做長達數十小時的訓練，我也只能一直等待下去……



問題：GPU整體平均使用率很低！



雖然有一些高峰，但大多時候都是低使用率和未充分利用的資源.....

- **預約**、**排隊** 的資源管理與調度太低效
- GPU 運算能力太好，但又無法分享多餘資源，使資源被浪費

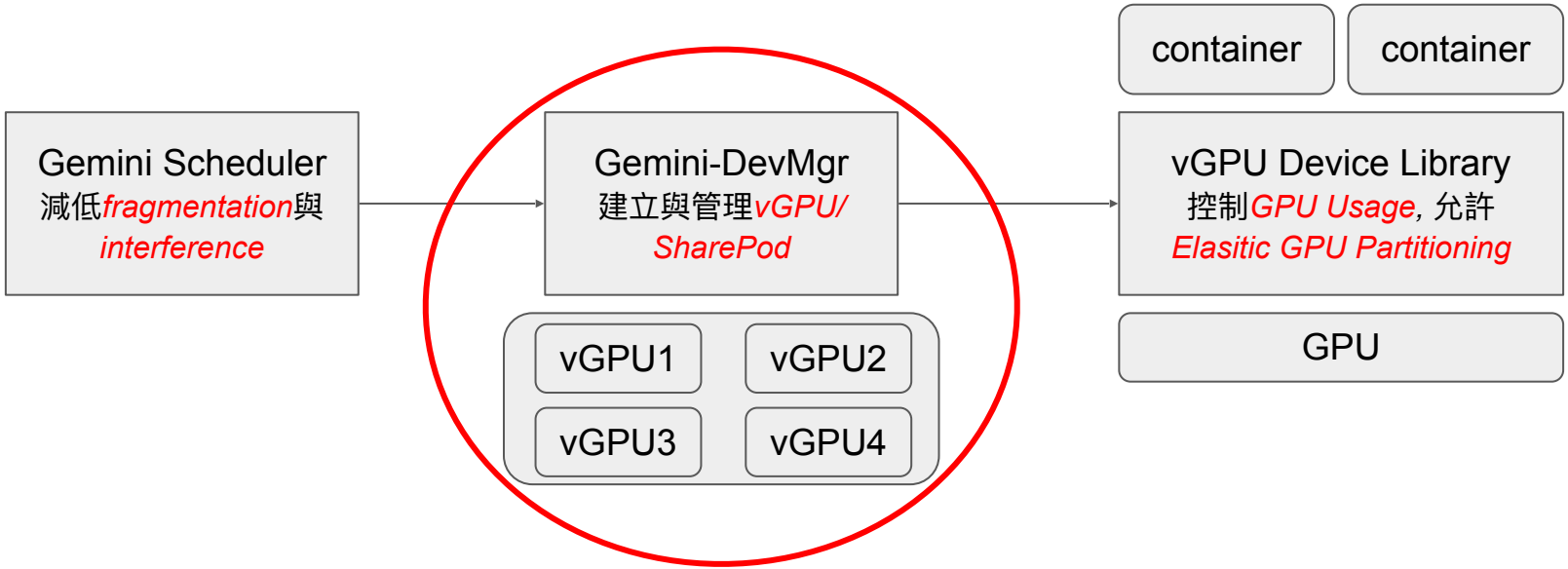
geminiopencloud

Agenda

- 在 Kubernetes 中使用 GPU 的痛點與挑戰
- GPU 分割技術架構設計

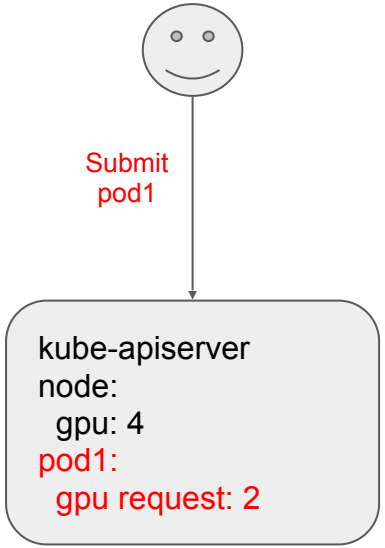
Gemini GPU Partitioning

目標：在Kubernetes環境下提供GPU分割技術，解決utilization, fragmentation與interference的問題



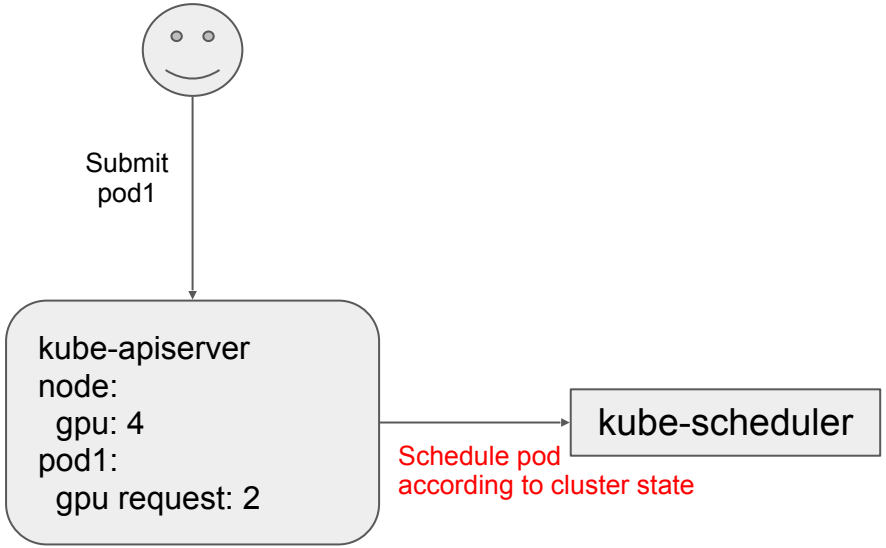
Kubernetes Device Plugin Framework

支援custom computing resource, 例如GPU, FPGA



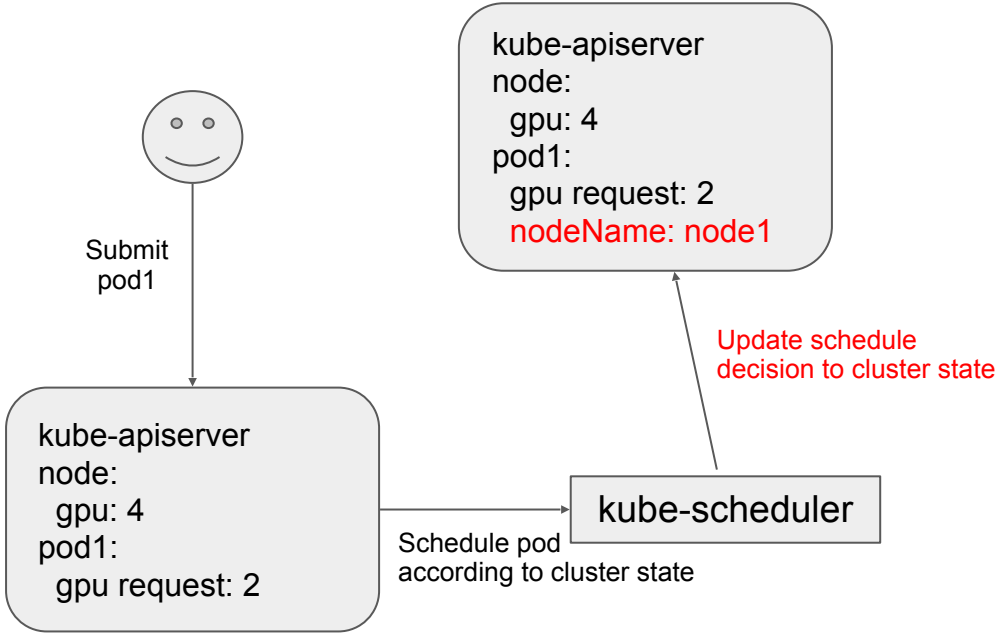
Kubernetes Device Plugin Framework

支援custom computing resource, 例如GPU, FPGA



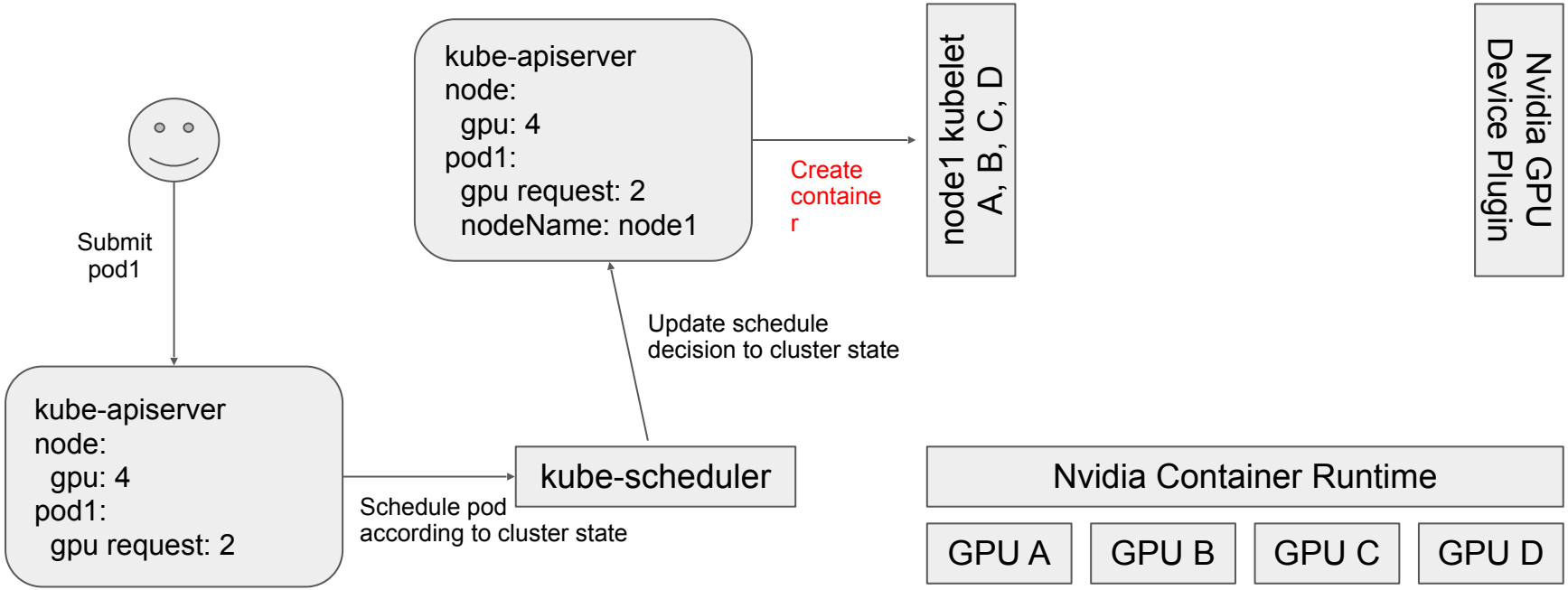
Kubernetes Device Plugin Framework

支援custom computing resource, 例如GPU, FPGA



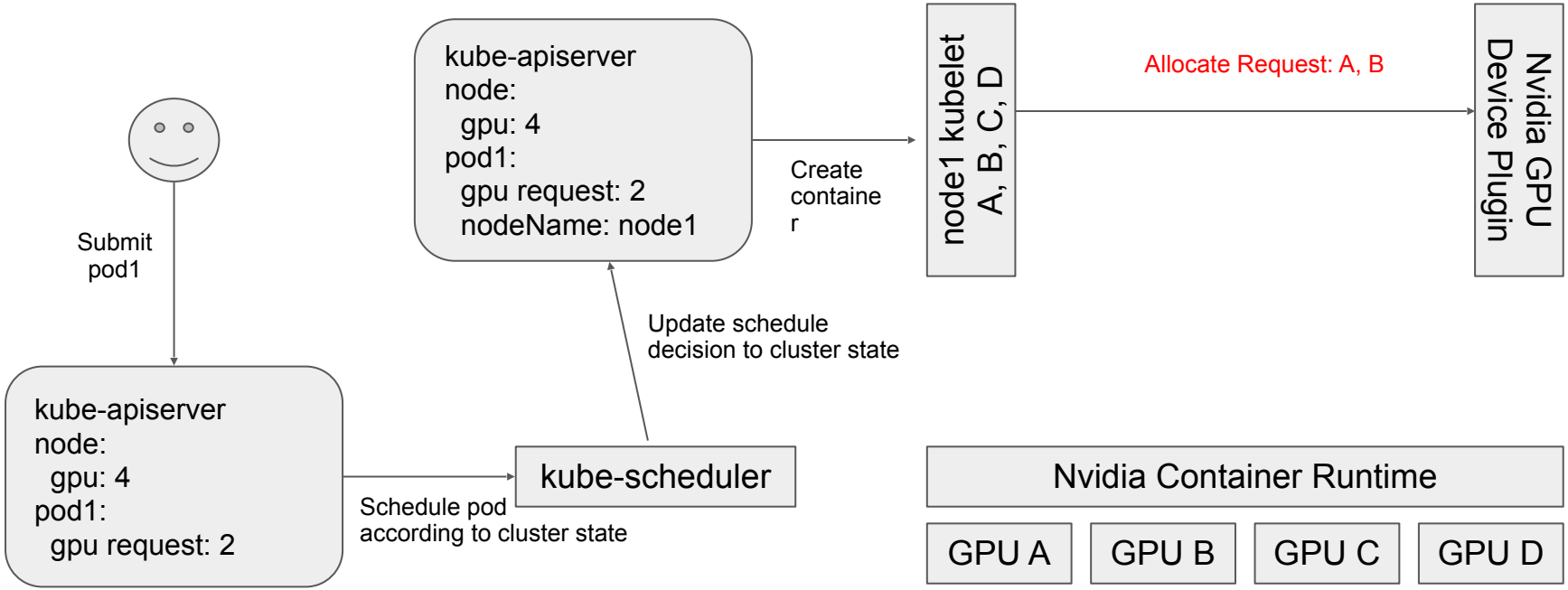
Kubernetes Device Plugin Framework

支援custom computing resource, 例如GPU, FPGA



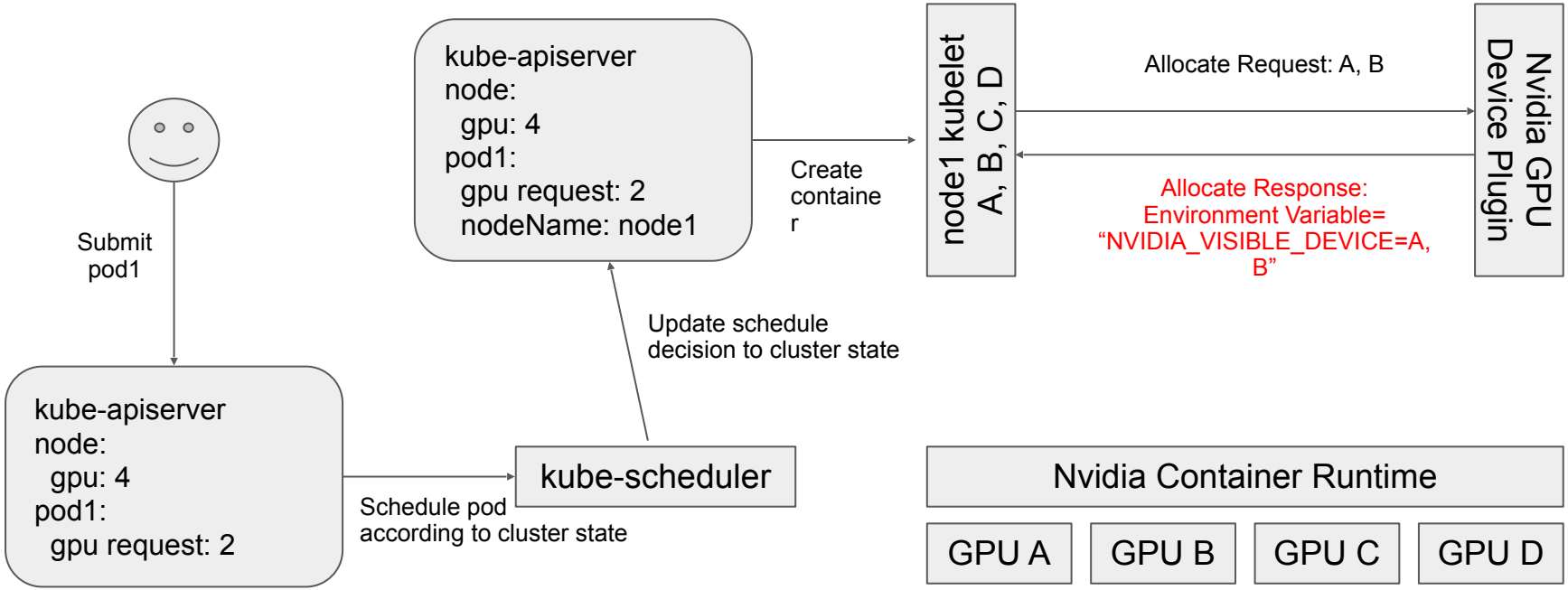
Kubernetes Device Plugin Framework

支援custom computing resource, 例如GPU, FPGA



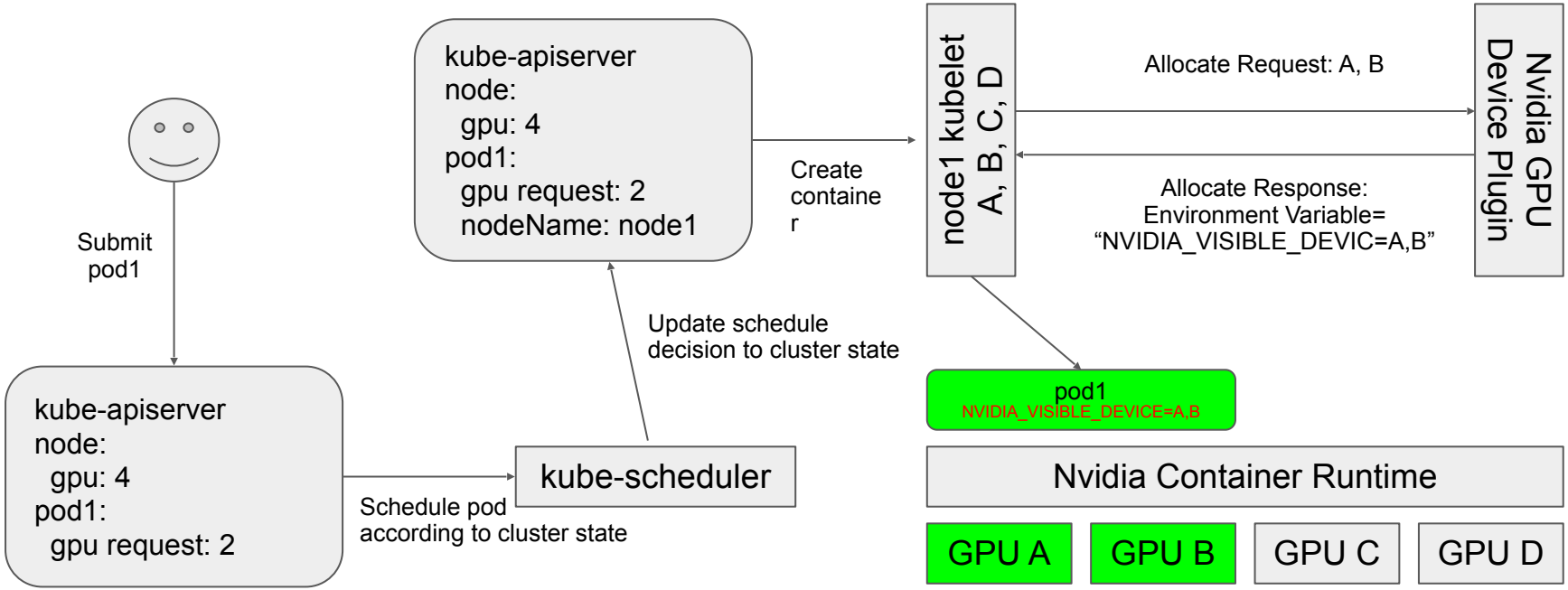
Kubernetes Device Plugin Framework

支援custom computing resource, 例如GPU, FPGA

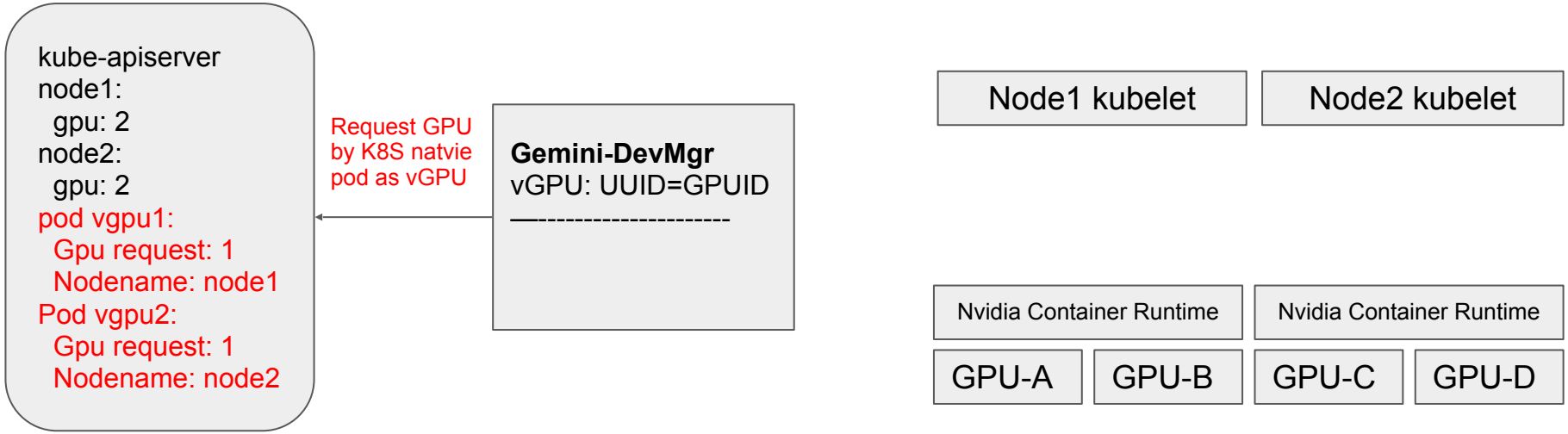


Kubernetes Device Plugin Framework

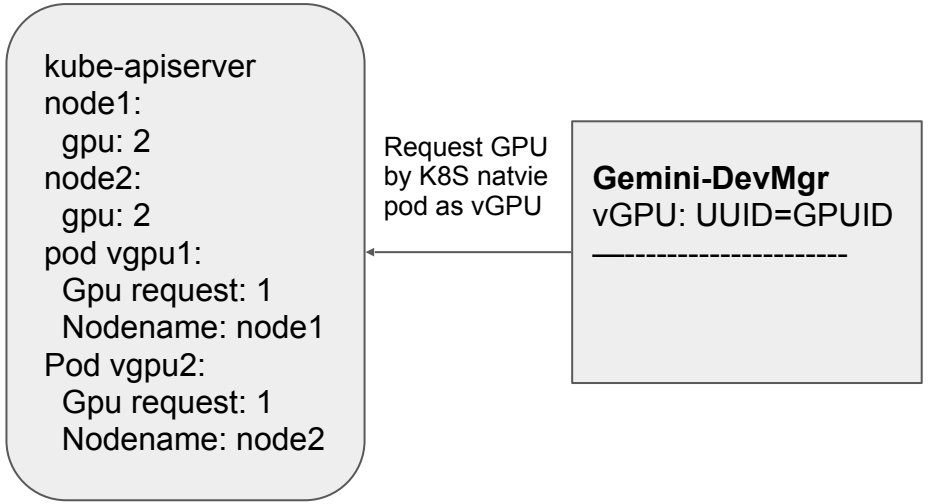
支援custom computing resource, 例如GPU, FPGA

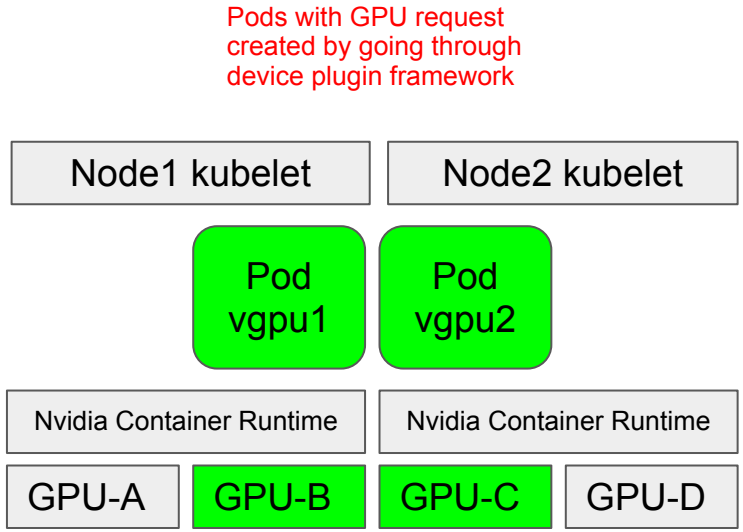


vGPU Creation

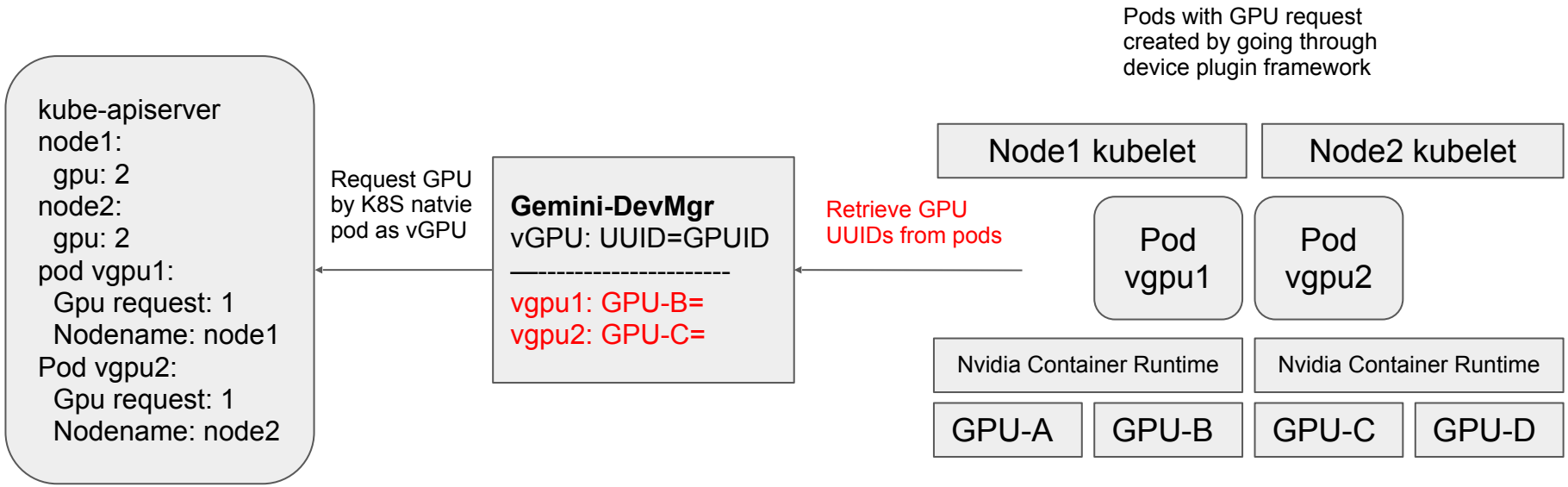


vGPU Creation

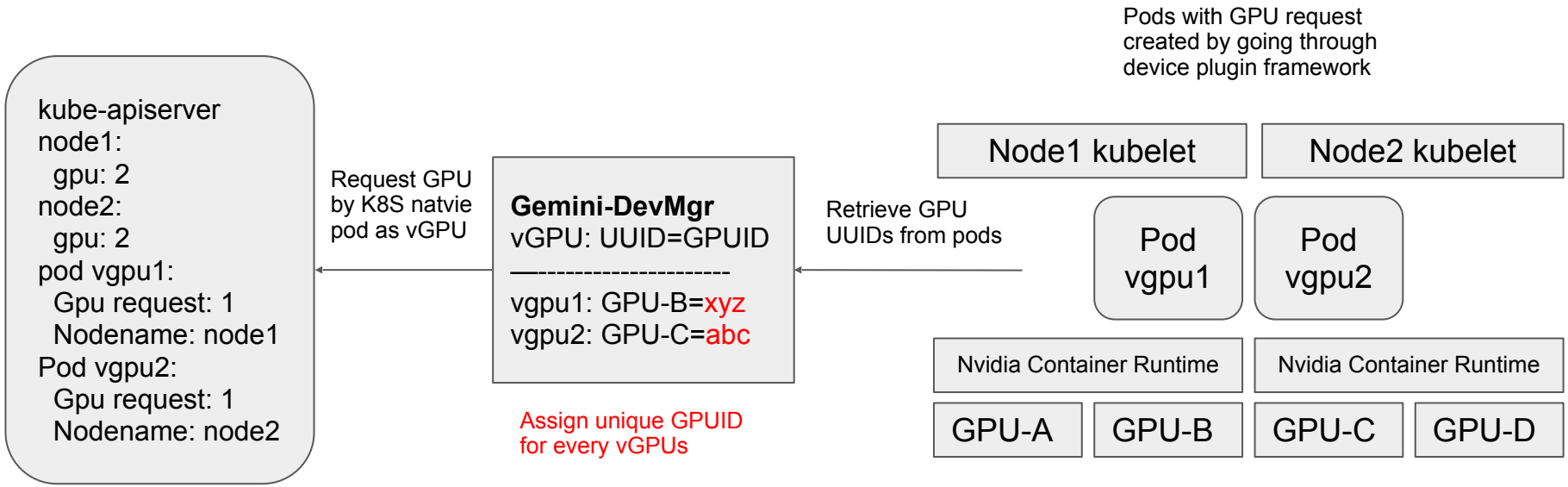




vGPU Creation



vGPU Creation



SharePod Creation

- SharePod 為我們自行定義的CRD（Custom Resource Definition）
- 一個ShardPod即為一個有attach vGPU的pod

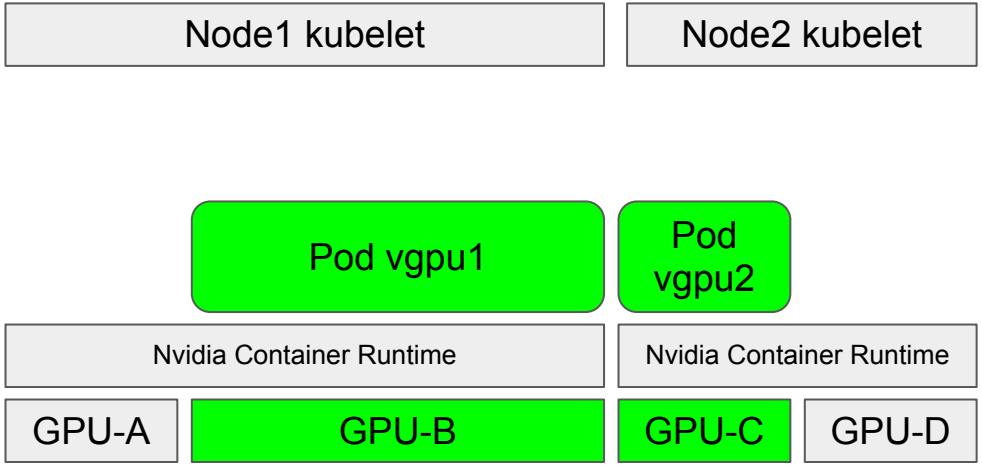
```
kind: SharePod
metadata:
  name: sharepod1
  annotations:
    "gemini/gpu_request": "0.5"
spec:
  containers:
  - name: cuda
    image: nvidia/cuda:9.0-base
    command: ["nvidia-smi", "-L"]
    resources:
      limits:
        cpu: "1"
          memory: "500Mi"
```

SharePod Creation

Sharepod1:
Gpu_req: 0.4
Node: node1
Gpuid: xyz
Sharepod2:
Gpu_req: 0.6
Node: node1
Gpuid: xyz

Submit sharepod with GPUID "xyz"

Gemini-DevMgr
vGPU: UUID=GPUID
—----------
vgpu1: GPU-B=xyz
vgpu2: GPU-C=abc



SharePod Creation

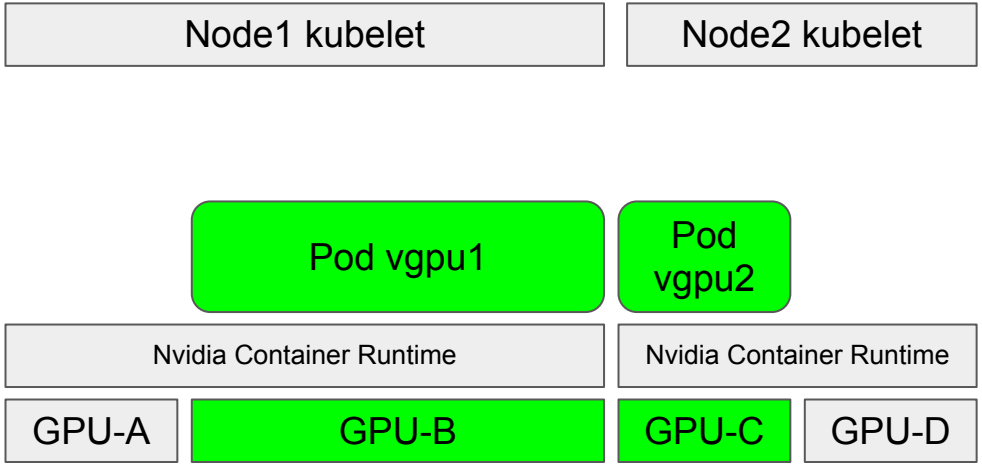
Sharepod1:
Gpu_req: 0.4
Node: node1
Gpuid: xyz
Sharepod2:
Gpu_req: 0.6
Node: node1
Gpuid: xyz

Submit sharepod with GPUID "xyz"

Gemini-DevMgr
vGPU: UUID=GPUID
—----------
vgpu1: GPU-B=xyz
vgpu2: GPU-C=abc

pod1:
Gpu_req: 0.4
~~Gpu request~~
Node: node1
Gpuid: xyz
Env:
NVIDIA_VISIABLE_DEVICES=GPU-B
pod2:
Gpu_req: 0.6
~~Gpu request~~
Node: node1
Gpuid: xyz
Env:
NVIDIA_VISIABLE_DEVICES=GPU-B

Create pod with env var "NVIDIA_VISIBLE_DEVICE" to restrict GPU visibility in container



SharePod Creation

Sharepod1:
Gpu_req: 0.4
Node: node1
Gpuid: xyz
Sharepod2:
Gpu_req: 0.6
Node: node1
Gpuid: xyz

Submit sharepod with GPUID "xyz"

Gemini-DevMgr
vGPU: UUID=GPUID
—----------
vgpu1: GPU-B=xyz
vgpu2: GPU-C=abc

pod1:
Gpu_req: 0.4
~~Gpu request~~
Node: node1
Gpuid: xyz
Env:
NVIDIA_VISIABLE_DEVICES=GPU-B
pod2:
Gpu_req: 0.6
~~Gpu request~~
Node: node1
Gpuid: xyz
Env:
NVIDIA_VISIABLE_DEVICES=GPU-B

Create pod with env var "NVIDIA_VISIBLE_DEVICE" to restrict GPU visibility in container

Run containers in cluster

Node1 kubelet

Node2 kubelet

Pod1
NVIDIA_VISIABLE_DEVICES=GPU-B

Pod2
NVIDIA_VISIABLE_DEVICES=GPU-B

Pod vgpu1

Pod vgpu2

Nvidia Container Runtime

Nvidia Container Runtime

GPU-A

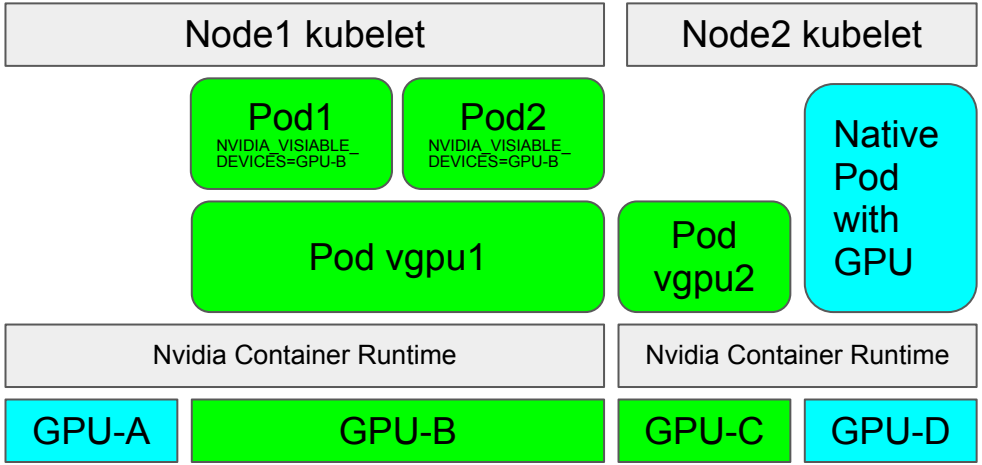
GPU-B

GPU-C

GPU-D

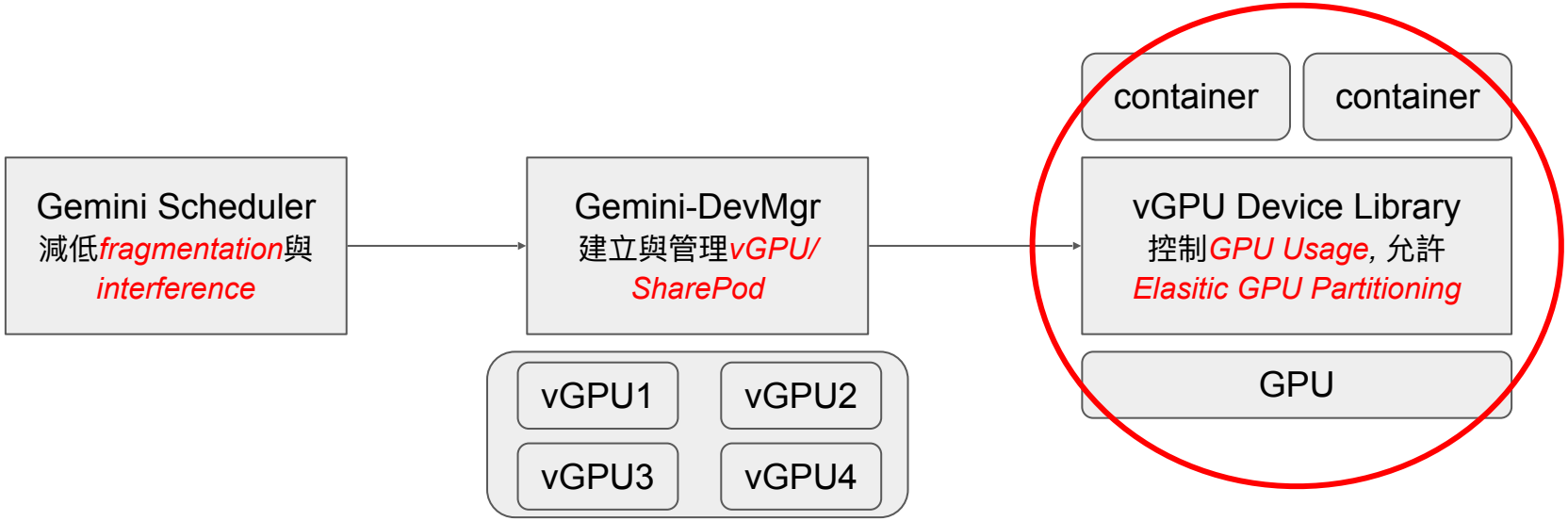
SharePod Creation

- Gemini GPU Partitioning可相容於NVIDIA GPU device plugin.



Gemini GPU Partitioning

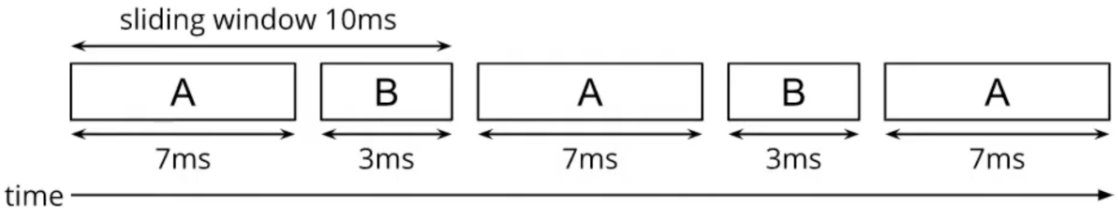
vGPU Device Library：Resource Control and Isolation



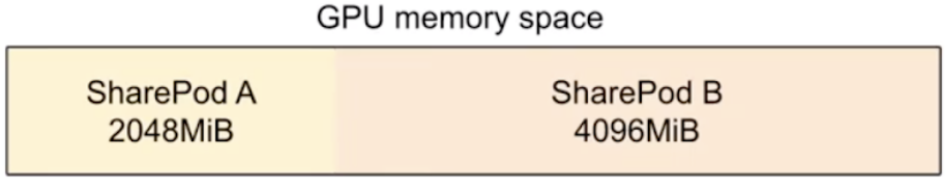
Resource Sharing Model

- Compute Resource：Time Sharing
  - Usage = (accumulated execution time in a sliding window) / (length of the sliding window)

GPU compute request
SharePod A: 0.7
SharePod B: 0.3

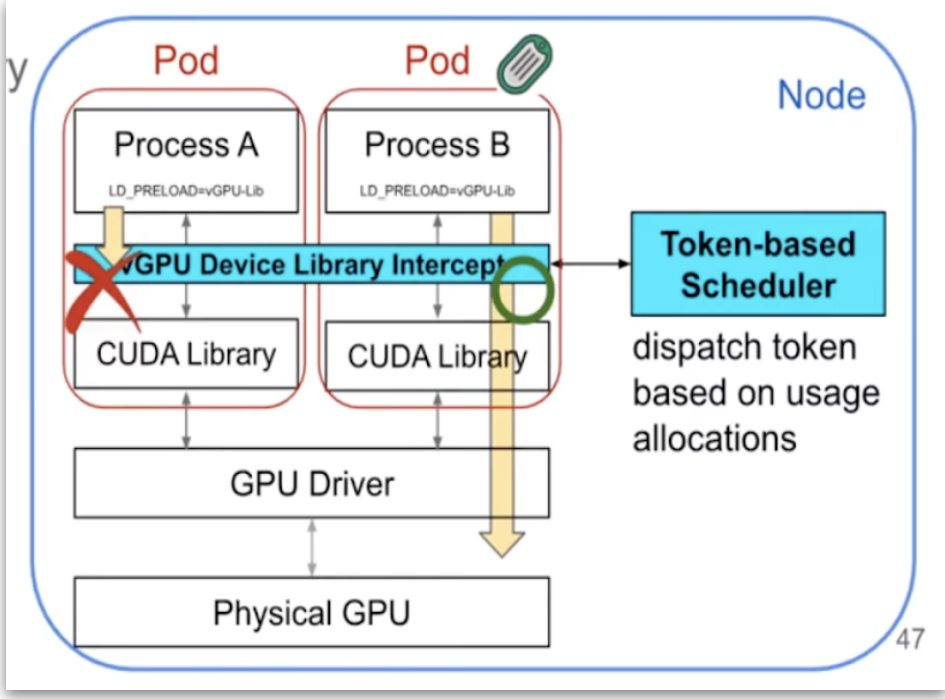


- Memory Resource：Space Sharing
  - Usage = total allocated memory size on GPU device memory



Resource Control Mechanism

- Method：intercept CUDA library calls using LD_PRELOAD
  - A pod can only launch GPU Kernels when it receives a token from scheduler
  - A pod can only allocate GPU memory when it doesn't exceed size limit



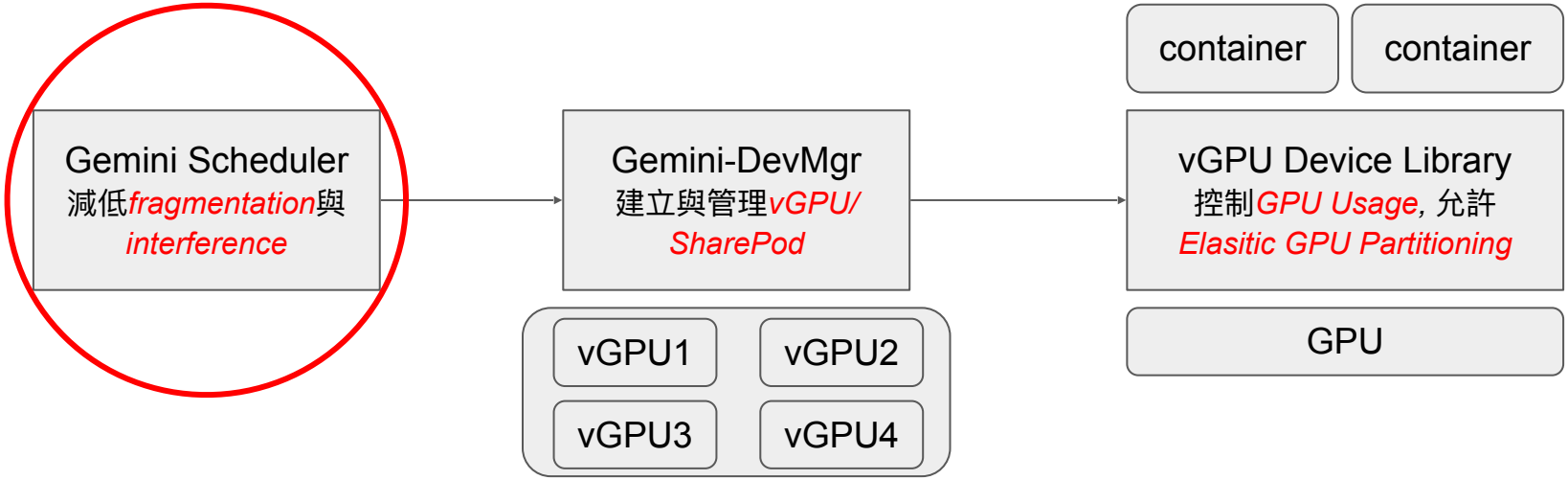
Elasitc GPU Partitioning

- More flexible resource allocation specifications for GPU time
  - Request：the minimum resource usage
  - Limit：the maximum resource usage
- Idle compute capacity can be shared without violating user requirements
  - Achieve higher gpu utilization

name	gpu_request	gpu_limit
Job A	0.4	0.7
Job B	0.6	0.8

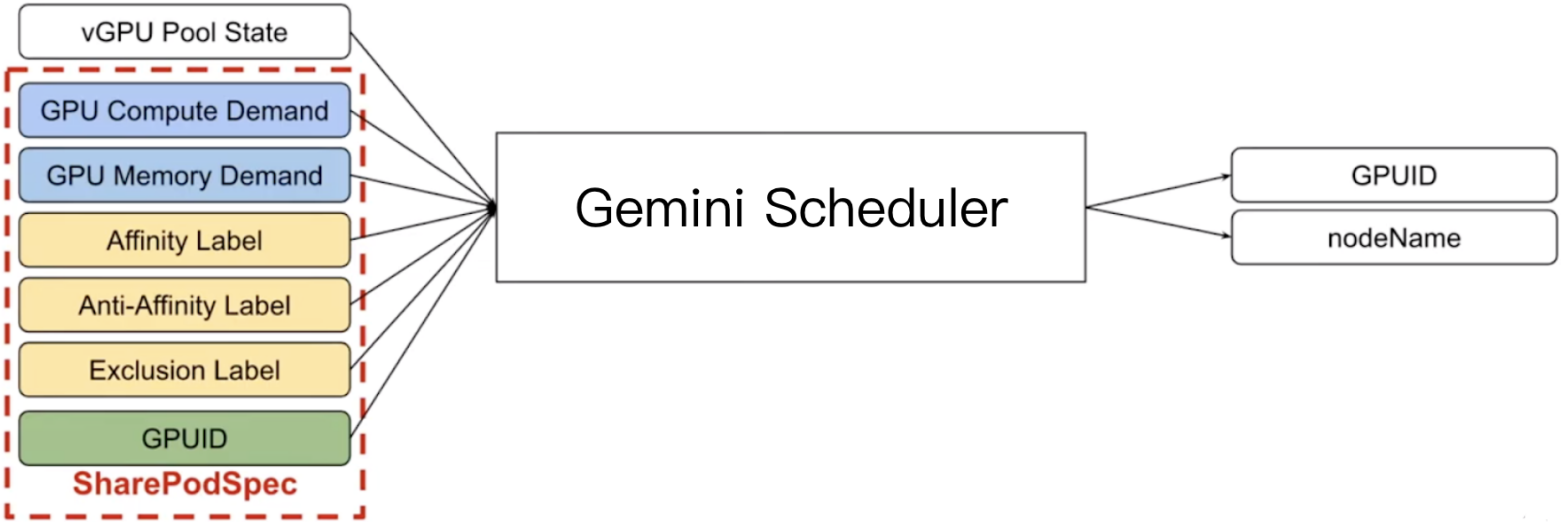
Gemini GPU Partitioning

Gemini Scheduler：Resource Requirement & Scheduling



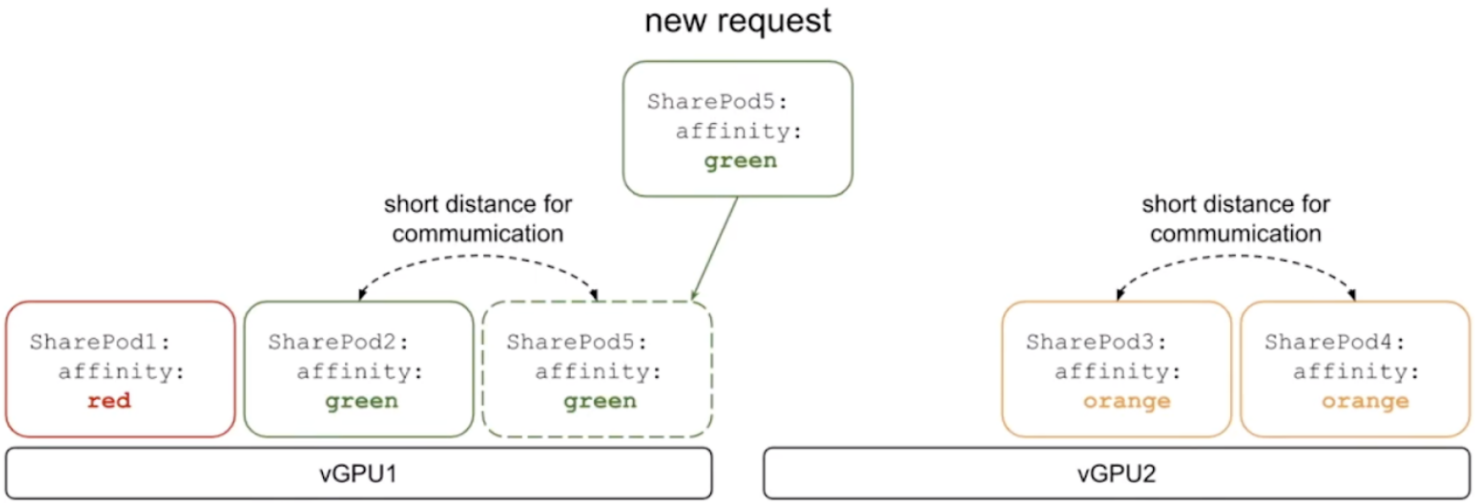
Resource Requirement Specification

- Gemini Scheduler schedules SharedPods by deciding their GPUID & nodeName
- Rich and Easy-to-use user specifications on GPU: usage, locality & identity



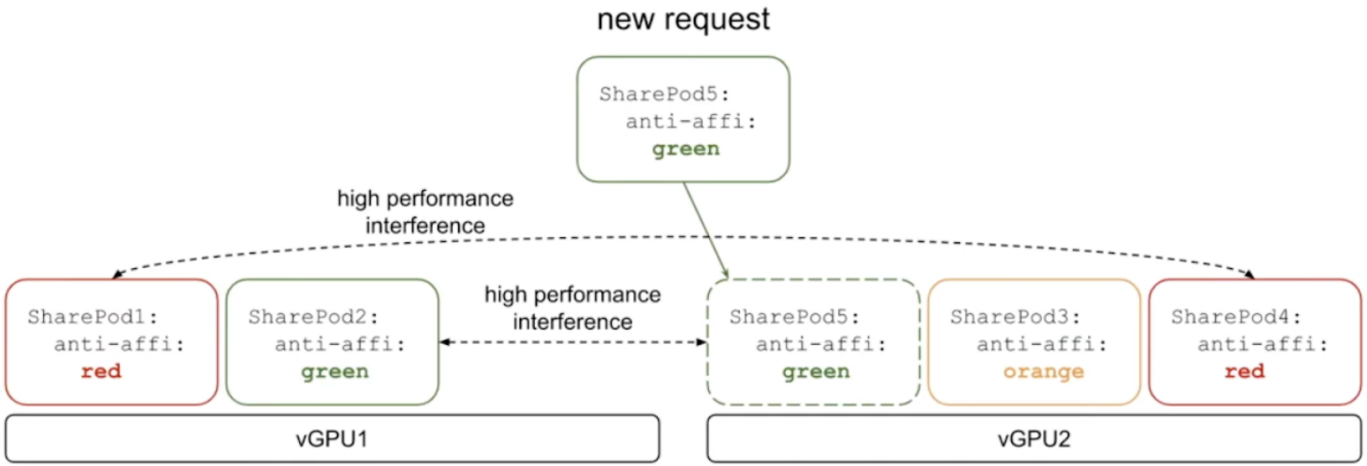
Scheduling Locality：Affinity

- Affinity forces container with the same label scheduled on the same GPU
- Reduce communication or data transfer overhead



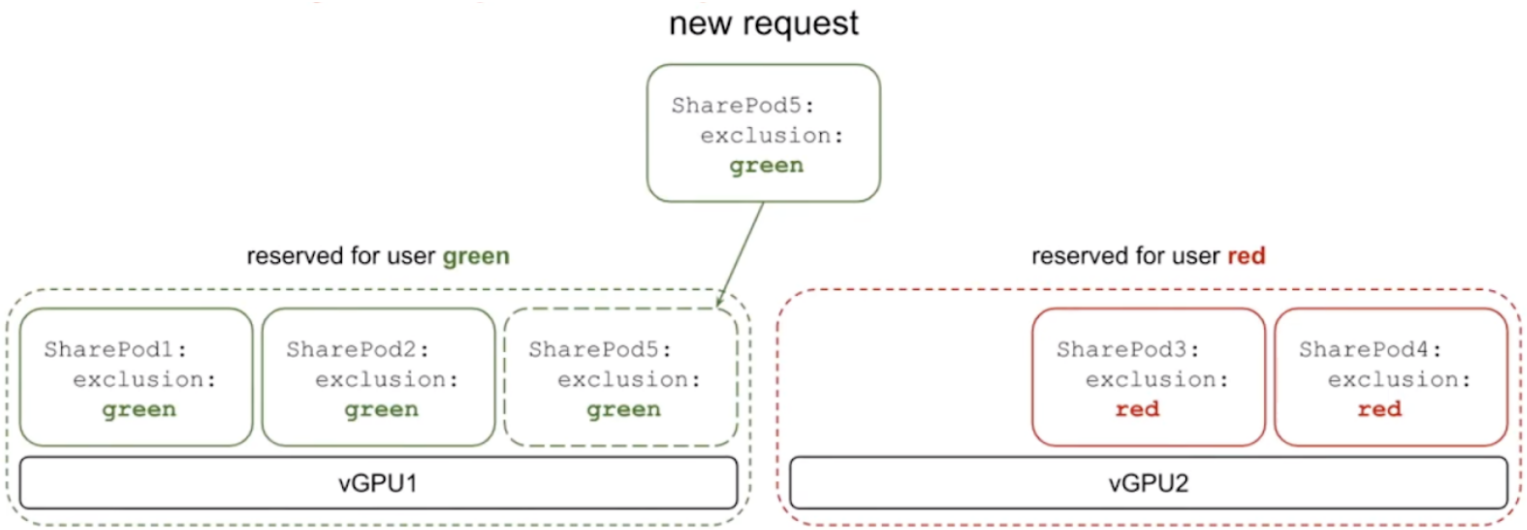
Scheduling Locality：Anti-Affinity

- Anti-Affinity forces containers with the same label scheduled on different GPUs
- Mitigate performance interference on shared GPU



Scheduling Locality：Exclusion

- Exclusion avoids GPU sharing among containers with different labels
- Dedicate GPU for specific users/applications



Agenda

- 在 Kubernetes 中使用 GPU 的痛點與挑戰
- GPU 分割技術架構設計

geminiopencloud

Summary

- Gemini GPU Partitioning 以軟體方式實作 GPU 共享的技術。
- 我們的設計架構與 Kubernetes Device Plugin與 CUDA 相容，使用者不需要額外修改程式碼即可使用 GPU Partitioning 功能。
- Gemini GPU Partitioning 提供多樣的specification，讓使用者更有彈性的使用 GPU。
- 使用 GPU Partitioning 的技術，以些微的overhead即可有效提高 GPU Utilization和Throughput。

geminiopencloud

工商時間

雙子星雲端運算 - 優化雲原生下的雲服務營運

FB粉絲團

電子報