KubeCon
CloudNativeCon
North America 2024

KubeCon | CloudNativeCon

North America 2024

A tale of 2 Drivers: GPU Configuration on the Fly

Alay Patel, Nvidia
Varun Sekar, Nvidia

Topics at a Glance

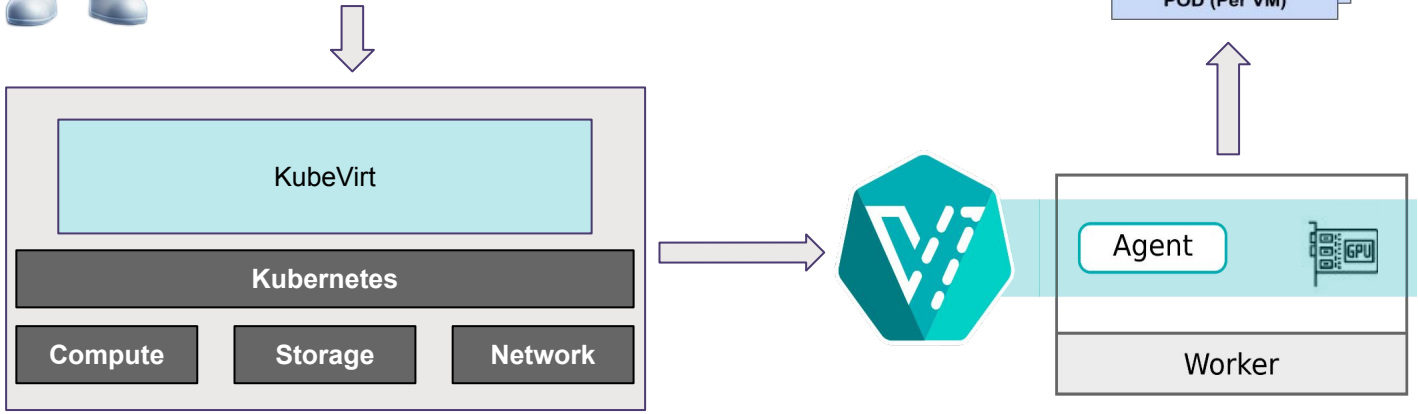
- NVIDIA GeForceNow Cloud
- Use Cases
- VMs with DevicePlugins
- Issues with current design
- DRA to the rescue
- Demo

NVIDIA's GeForceNow Cloud

- Cloud for running GPU workloads
- Kubernetes extensions running in the cluster
  - KubeVirt
  - OVN-Kubernetes
  - Fluentbit
  - Prometheus
  - OPA, Gatekeeper
- Scale:
  - 40+ DCs
  - 30000+ managed nodes
  - 60000+ managed GPUs
  - 1M+ daily workload churn

VirtualMachine is a CustomResource!

```yaml
apiVersion: kubevirt.io/v1
kind: VirtualMachineInstance
...
spec:
  domain:
    cpu:
      cores: 2
    devices:
      disks:
      - windows11
      gpus:
      - deviceName: nvidia.com/gpu
        name: pgpu
```



Use Cases: full-GPU

User 1: I want to play AAA games at the highest FPS and maximum graphics that my PC can't support!

GeForceNow: Okay, I'll run a large VM with a full-GPU (VFIO passthrough) whenever you want to play!

Use Cases: n:1 vGPU

User 2: I'm getting a membership to play games at decent FPS with at least 1080p resolution.

GeForceNow: Okay, I'll run a medium VM with a half-GPU (2:1 vGPU) whenever you want to play!

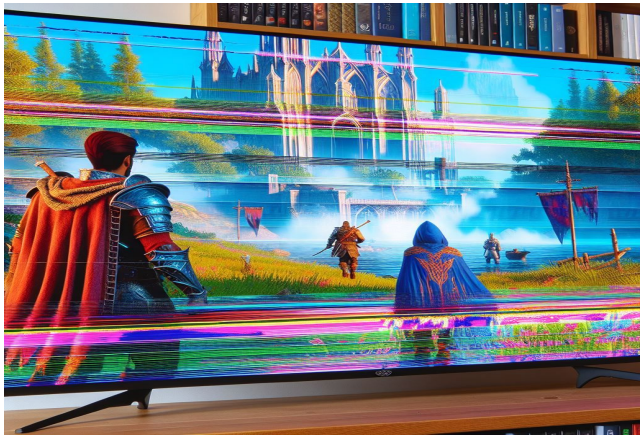
User 3: I'm getting a membership just to explore the offering. So my graphics quality doesn't matter to me.

GeForceNow: Okay, I'll run a small VM with a quarter-GPU (4:1 vGPU) whenever you want to play!

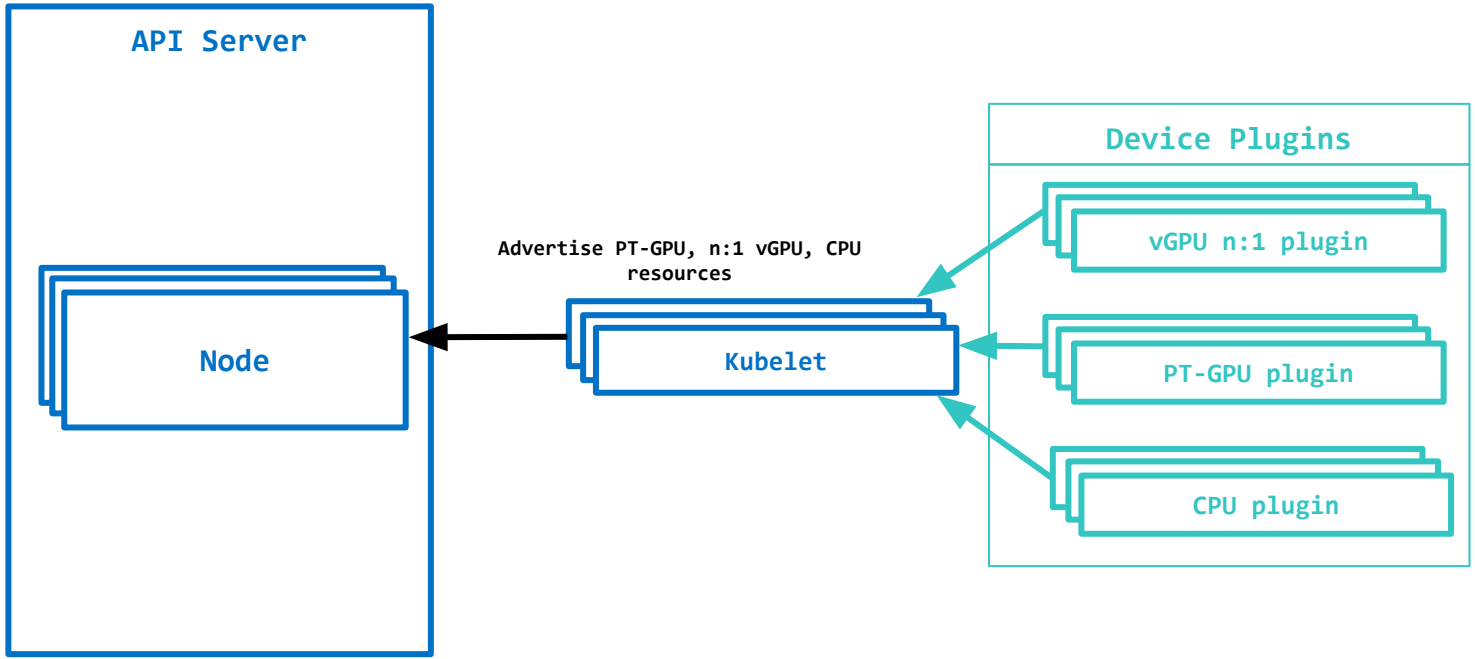
Use Cases: CPU-GPU Numa Alignment

User: Hey! I don't want my games to stutter!

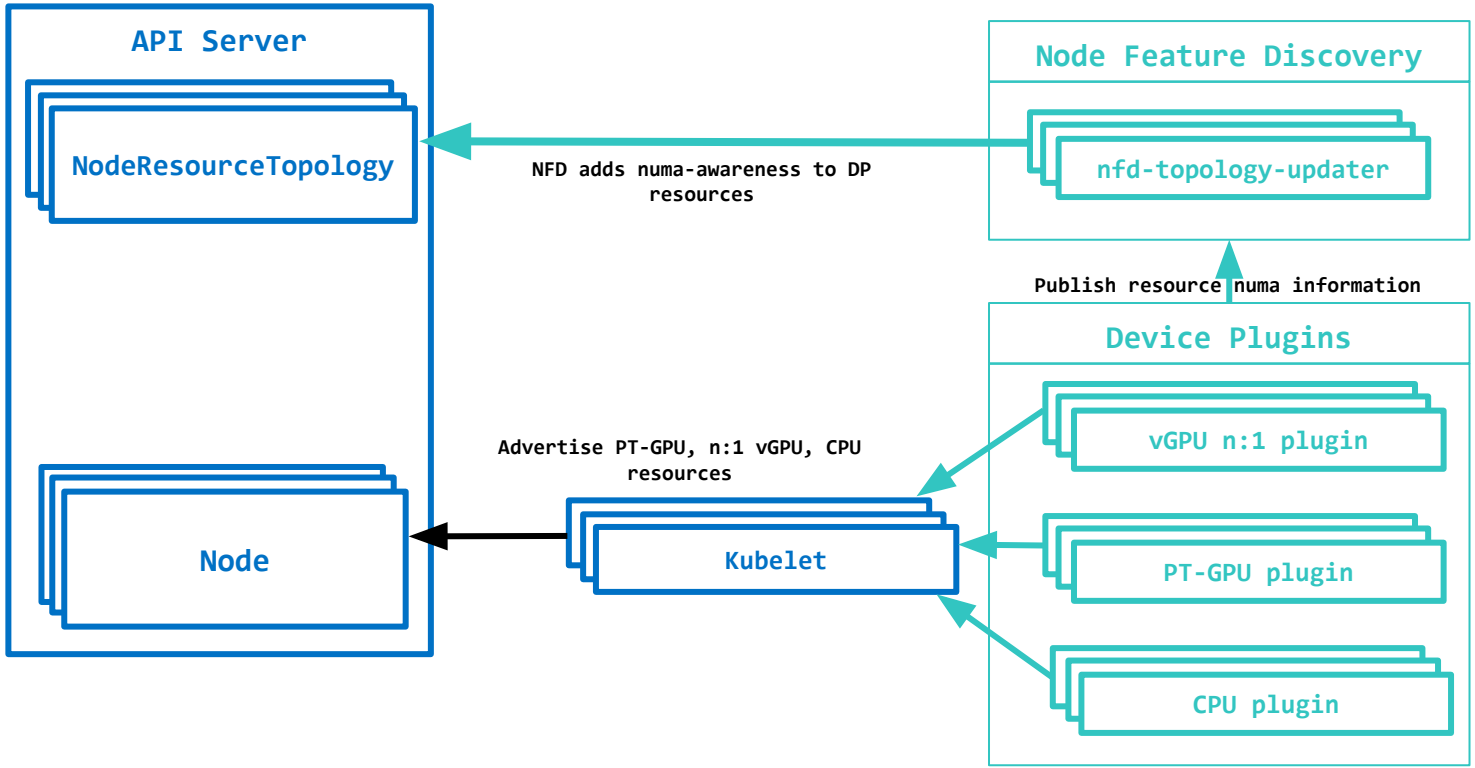
GeForceNow: Okay, I'll make sure all the VMs I run are tuned for optimal performance with NUMA-aligned resources!



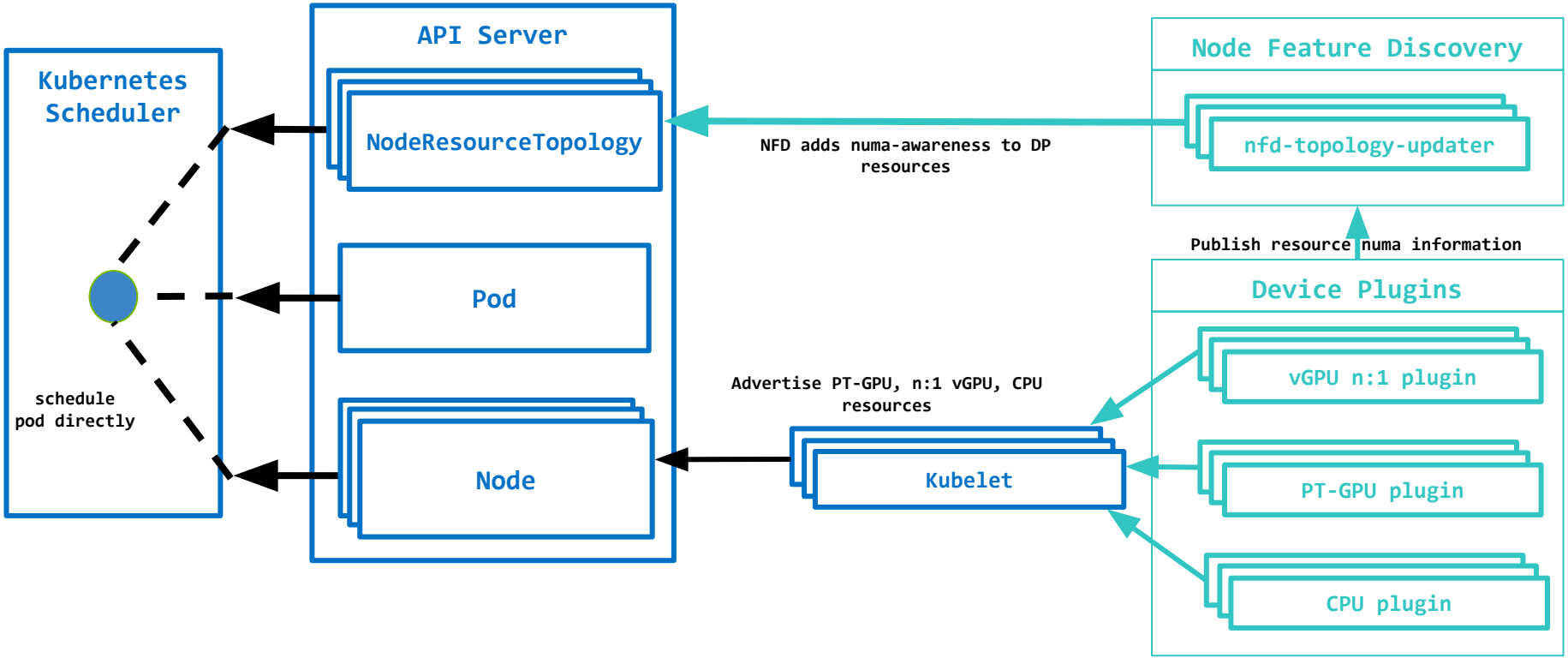
How do users get the NUMA-aligned GPU+CPU?



How do users get the NUMA-aligned GPU+CPU?



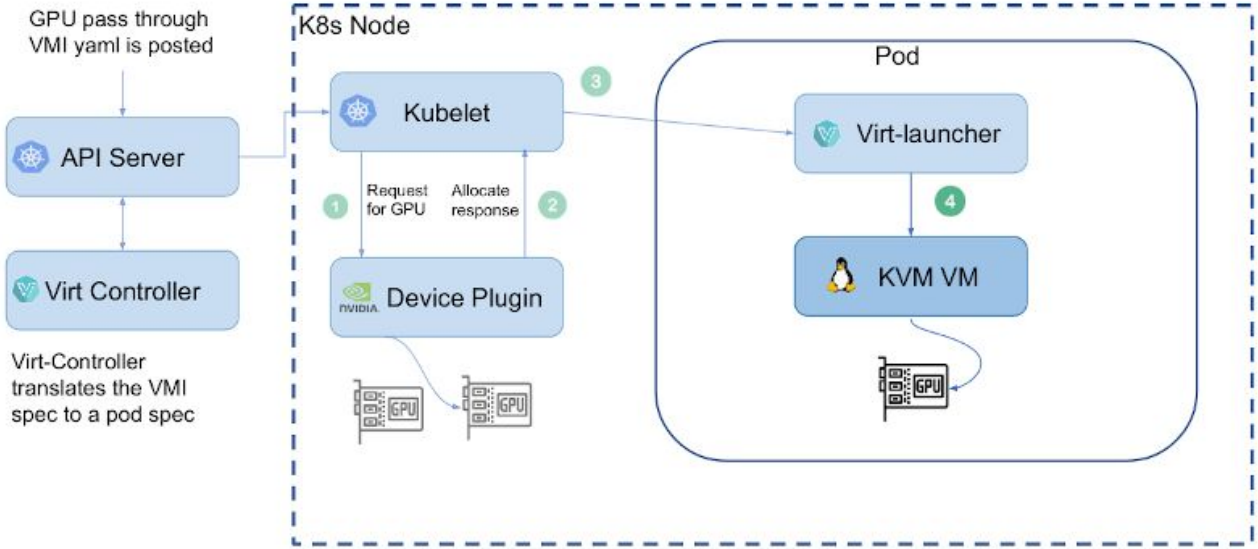
How do users get the NUMA-aligned GPU+CPU?



How do we start a VM?


```yaml
apiVersion: kubevirt.io/v1
kind: VirtualMachineInstance
...
spec:
  domain:
    cpu:
      cores: 2
    devices:
      gpus:
      - deviceName: nvidia.com/gpu
        name: gpu1
    resources:
      requests:
        memory: 56Gi
```

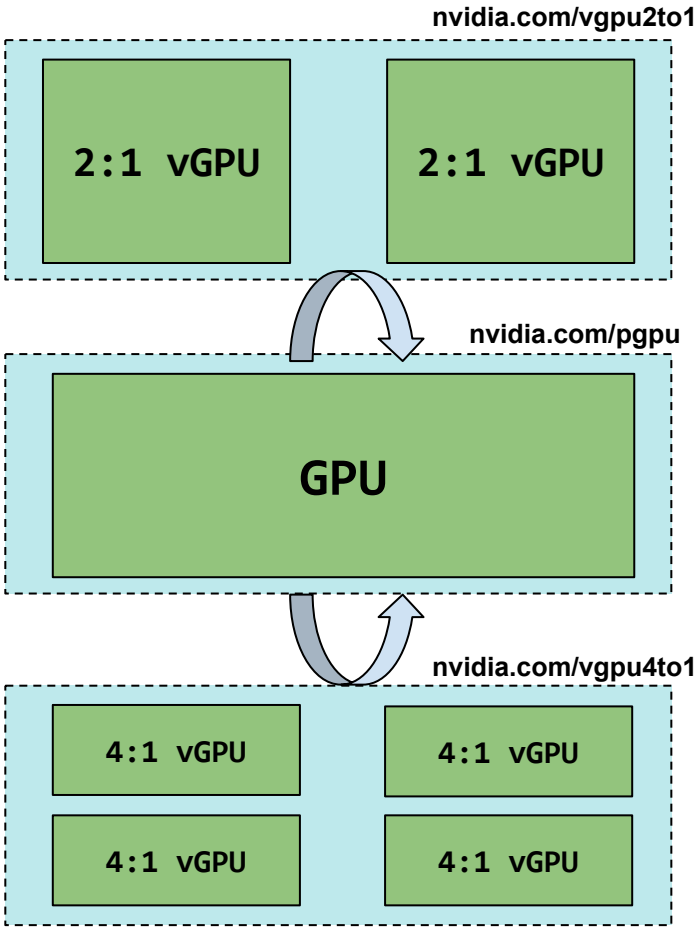
```yaml
apiVersion:v1
kind: Pod
...
spec:
  containers:
  - name: virt-launcher
    resources:
      requests:
        nvidia.com/cpu: 2
        nvidia.com/gpu: 1
        memory: 56Gi
```



Hiding the mess!!!!

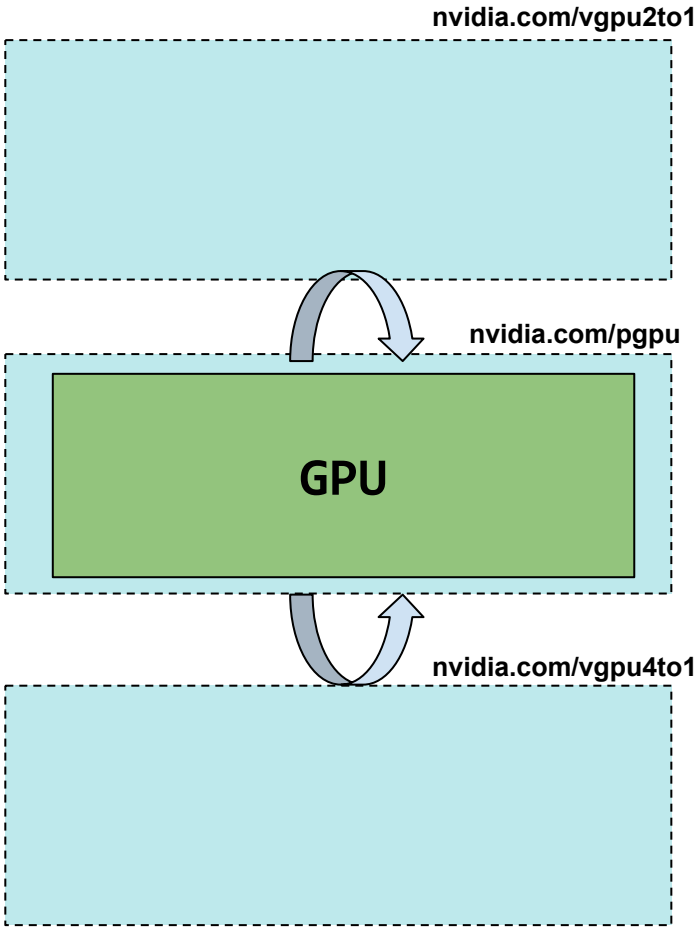
Multiple Device Plugins for the same resource

- Each GPU type needs its own plugin registered with kubelet.
- The different plugins need to coordinate to maintain state.



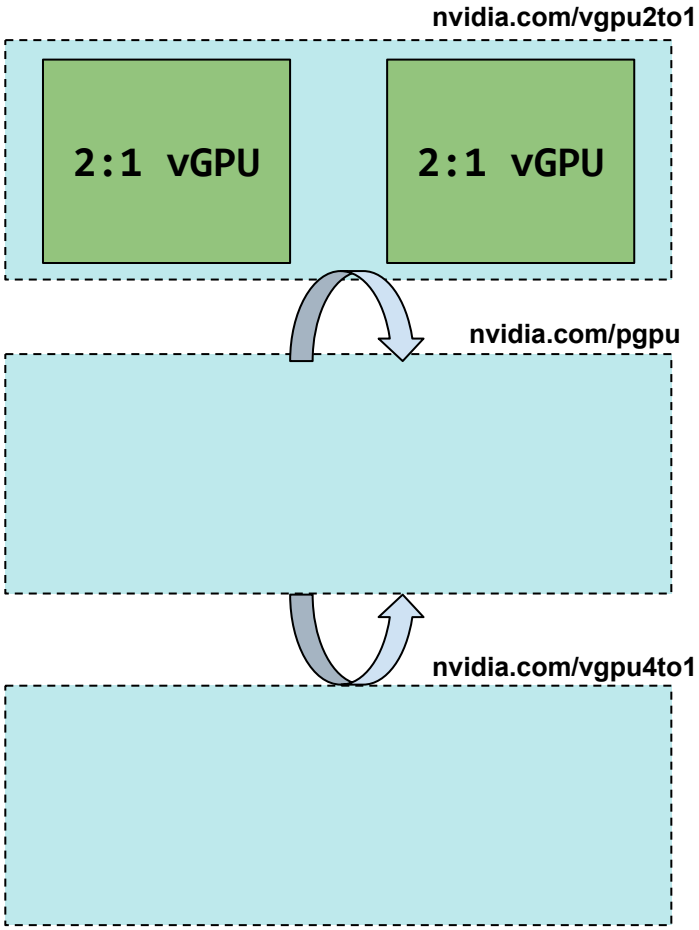
Multiple Device Plugins for the same resource

- If **nvidia.com/pgpu** is allocated, then the vGPUs should be deadvertised.



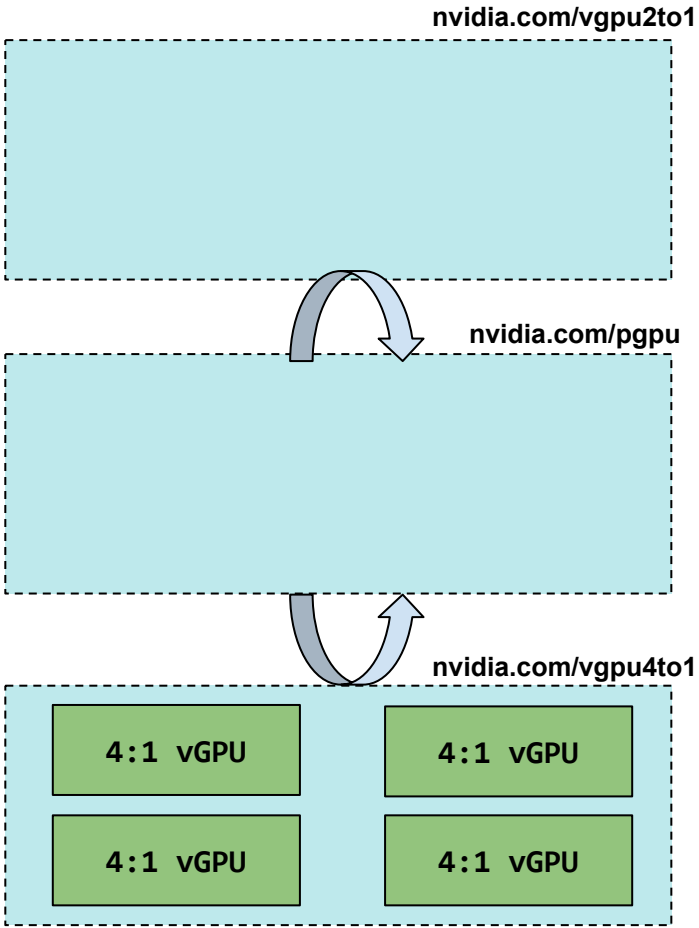
Multiple Device Plugins for the same resource

- If **nvidia.com/vgpu2to1** is allocated, then the full-GPU and the 4:1 vGPU needs to be deadvertised.



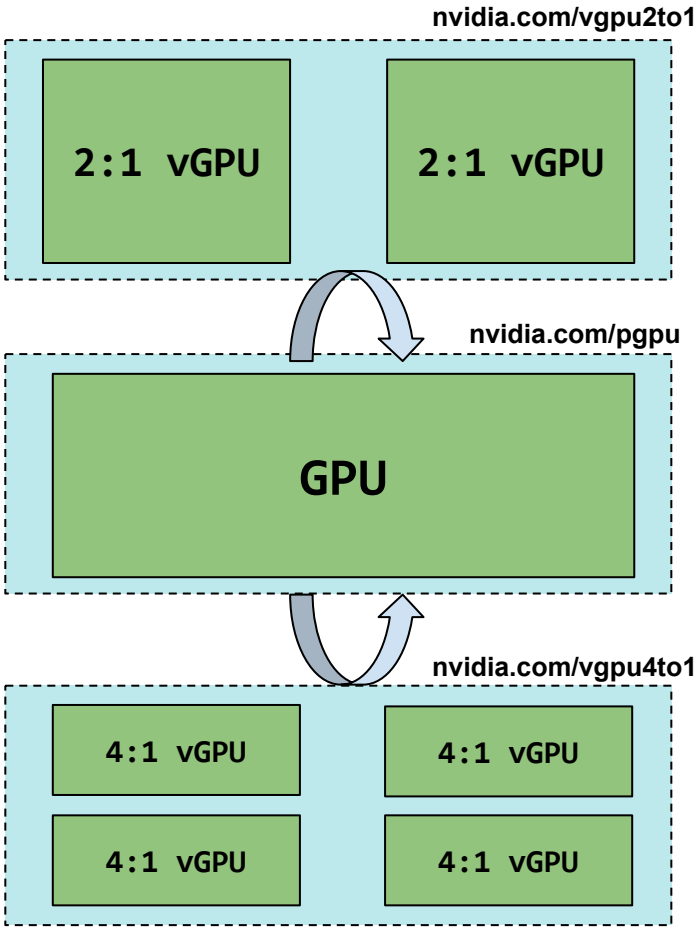
Multiple Device Plugins for the same resource

- If **nvidia.com/vgpu4to1** is allocated, then the full-GPU and the 2:1 vGPU needs to be deadvertised.



What happens when GPU is freed up ?!

- DP framework does not have a deallocate API
  - On Deallocation, the different plugins need to reset their device states



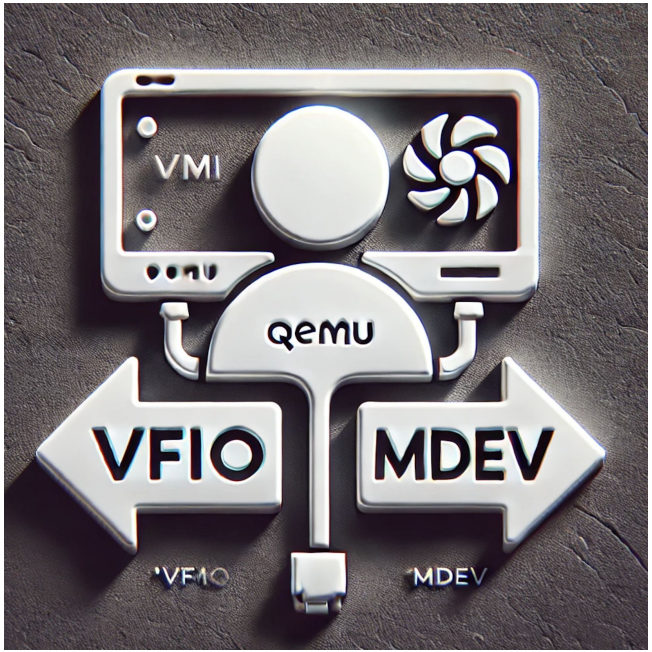
No Deallocate API

- DP framework does not have a deallocate API
  - On Deallocation, the different plugins need to reset their device states

```
// DevicePlugin is the service advertised by Device Plugins
service DevicePlugin {
  rpc GetDevicePluginOptions(Empty) returns (DevicePluginOptions) {}

  rpc ListAndWatch(Empty) returns (stream ListAndWatchResponse) {}

  rpc GetPreferredAllocation(PreferredAllocationRequest) returns (PreferredAllocationResponse) {}

  rpc Allocate(AllocateRequest) returns (AllocateResponse) {}

  rpc PreStartContainer(PreStartContainerRequest) returns (PreStartContainerResponse) {}
}
```

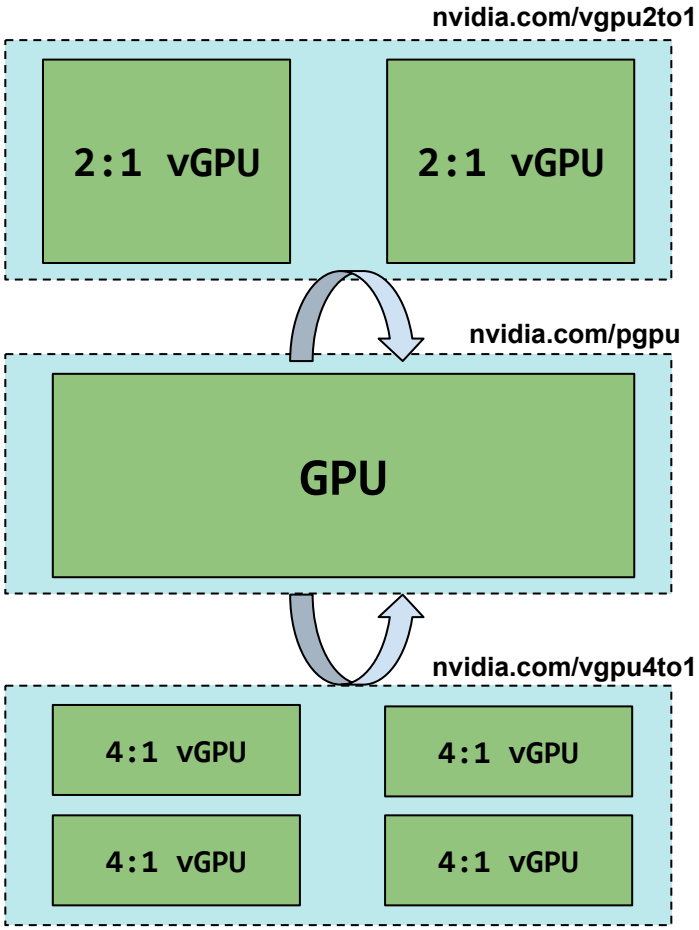
Pod Startup Latency

- Kubelet does allocations and configuration serially
- Device reconfiguration is costly
  - Specifically with GPUs in Virtual Machines
- Makes starting workloads latency sensitive



Racy scheduler

- Device plugin (DP) work at node level
  - Could race with cluster components like scheduler
  - Until pod admission happens, scheduler will still see the old device states and makes uninformed scheduling decisions.
- And many more...



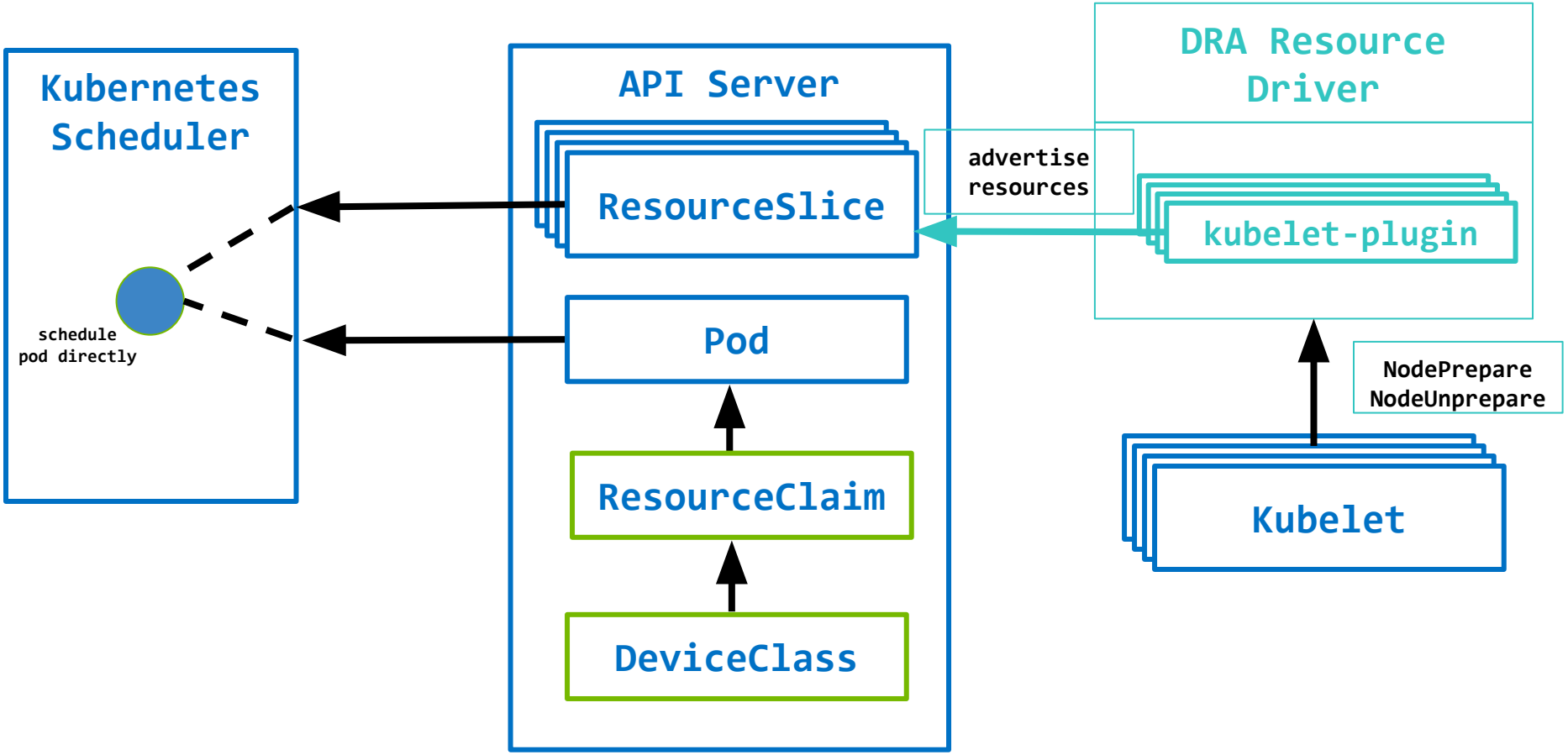
KubeCon | CloudNativeCon

North America 2024

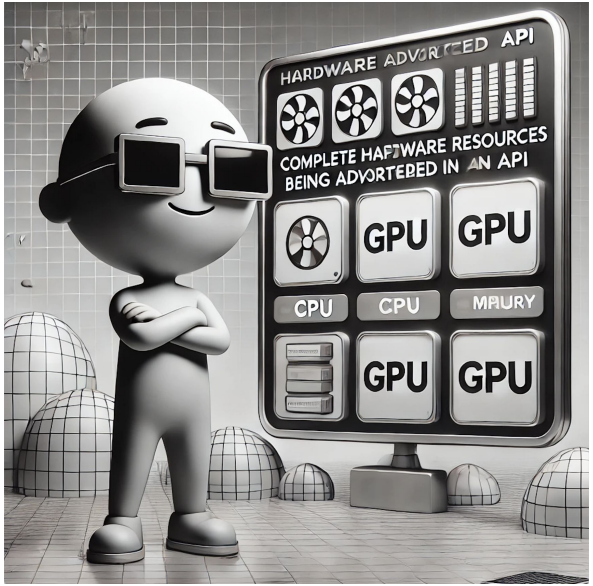
DRA to the rescue

What is DRA?

- Dynamic Resource Allocation (DRA)
  - An API for requesting resources in a pod
- Generalization of PVC API for generic resources like GPUs
- Similar to Device Plugins, third-party drivers manages the resources
- Kubernetes does the allocation

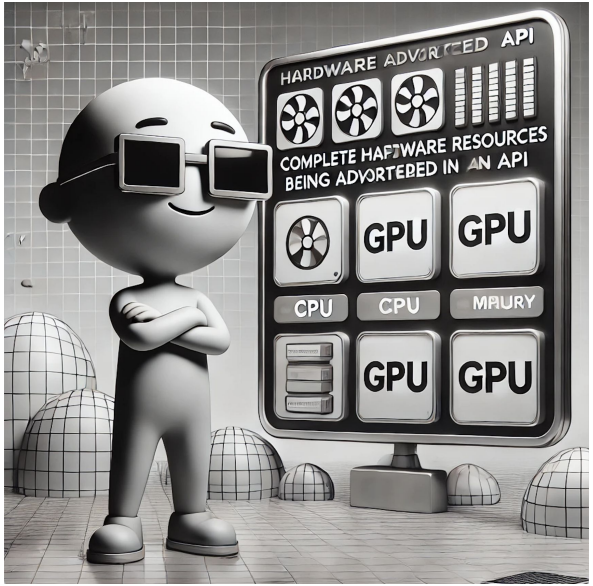

Kubernetes Scheduler
schedule pod directly
API Server
ResourceSlice
Pod
ResourceClaim
DeviceClass
advertise resources
DRA Resource Driver
kubelet-plugin
NodePrepare
NodeUnprepare
Kubelet

Advertised resources: GPU




```yaml
apiVersion: resource.k8s.io/v1alpha3
kind: ResourceSlice
metadata:
  generateName: kind-1.31-dra-control-plane-gpu.example.com-
  name: kind-1.31-dra-control-plane-gpu.example.com-drr27
spec:
  driver: gpu.example.com
  nodeName: kind-1.31-dra-control-plane
  pool:
    ...
  devices:
  - basic:
      attributes:
        type:
          string: gpu
        pciAddress:
          string: 0000:01:00.0
        dra.example.com/numa:
          string: 0
    name: pgpu-0
...
```

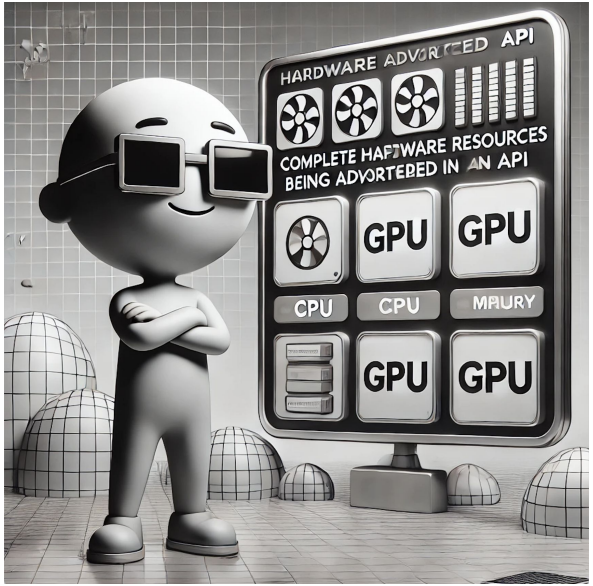
Contd ...




```
...
  - basic:
      attributes:
        type:
          string: vgpu
        uuid:
          string: 8e942949-f10b-d871-09b0-ee0657e28f90-0
        class:
          string: vgpu-2-to-1
        dra.example.com/numa:
          string: 0
    name: vgpu-2-to-1-0
```

```
...
  - basic:
      attributes:
        type:
          string: vgpu
        uuid:
          string: 8e942949-f10b-d871-09b0-ee0657e28f90-0
        class:
          string: vgpu-4-to-1
        dra.example.com/numa:
          string: 0
    name: vgpu-4-to-1-0
```

Advertised resources: CPU




```
apiVersion: resource.k8s.io/v1alpha3
kind: ResourceSlice
metadata:
  name: kind-1.31-dra-worker-cpu.example.com-c4jmq
spec:
  devices:
  - basic:
      attributes:
        architecture:
          string: amd64
        id:
          int: 0
        parentID:
          int: 0
        dra.example.com/numa:
          int: 1
    name: cpu-0
  - basic:
...
        id:
          int: 15
        parentId:
          int: 0
...
```

NUMA aligned pGPU and CPU

```yaml
apiVersion: resource.k8s.io/v1alpha3
kind: ResourceClaimTemplate
metadata:
  namespace: aligned-devices
  name: pgpu-test
spec:
  spec:
    devices:
      requests:
      - name: pgpu
        deviceClassName: vfiopci.example.com
        selectors:
        - cel:
            expression: "device.attributes['gpu.example.com'].type == 'gpu'"
      - name: cpu-full-numa
        deviceClassName: cpu.example.com

      constraints:
      - requests: ["pgpu", "cpu-full-numa"]
        matchAttribute: "dra.example.com/numa"
```

NUMA aligned 2:1 vGPU and CPU

```yaml
apiVersion: resource.k8s.io/v1alpha3
kind: ResourceClaimTemplate
metadata:
  namespace: numa-aligned-devices
  name: vgpu-test
spec:
  spec:
    devices:
      requests:
      - name: vgpu
        deviceClassName: vgpu2to1.example.com
        selectors:
        - cel:
            expression: "device.attributes['gpu.example.com'].type == 'vgpu' &&
device.attributes['gpu.example.com'].class == 'vgpu-2-to-1-0'"
      - name: cpu-half-numa
        deviceClassName: cpu.example.com

    constraints:
    - requests: ["vgpu", "cpu-half-numa"]
      matchAttribute: "dra.example.com/numa"
```

DEMO

Going forward with DRA ...

- Partitionable Devices: KEP 4815
- Resource Health: KEP-4680
- Potential of other attribute standardization: KEP 4817
- Support DRA in KubeVirt: design proposal

Q & A