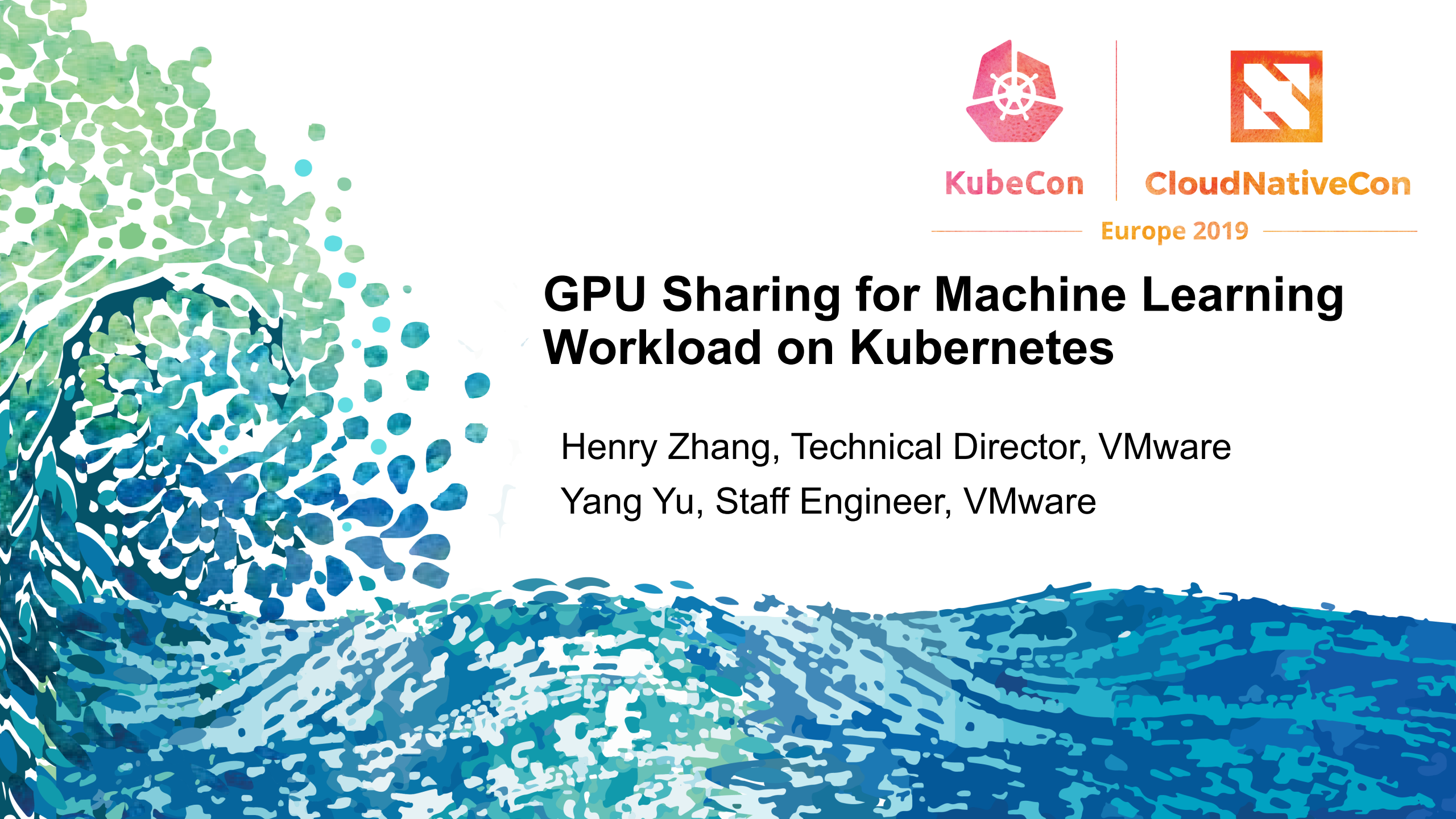


GPU Sharing for Machine Learning Workload on Kubernetes

Henry Zhang, Technical Director, VMware

Yang Yu, Staff Engineer, VMware

About Us

Henry Zhang

- Technical Director, VMware China R&D
- Founder of Project Harbor, an open source container registry hosted by CNCF
- Former evangelist of Cloud Foundry China community
- Hyperledger Cello Contributor
- Current interest: cloud computing, AI, blockchain etc.

Yang Yu

- Staff Engineer, VMware China R&D
- Working on VMware Kubernetes products
- Familiar with OpenStack's networking component Neutron
- Speaker of KubeCon Europe 2018, KubeCon China 2018

The New Era of Artificial Intelligence

- AI will be a transformative technology for organizations
  - Reduce human efforts
  - Augment human capabilities
  - Boost productivity
  - Save cost
- Widely used in business, society and our daily lives

Business

Education

Health care

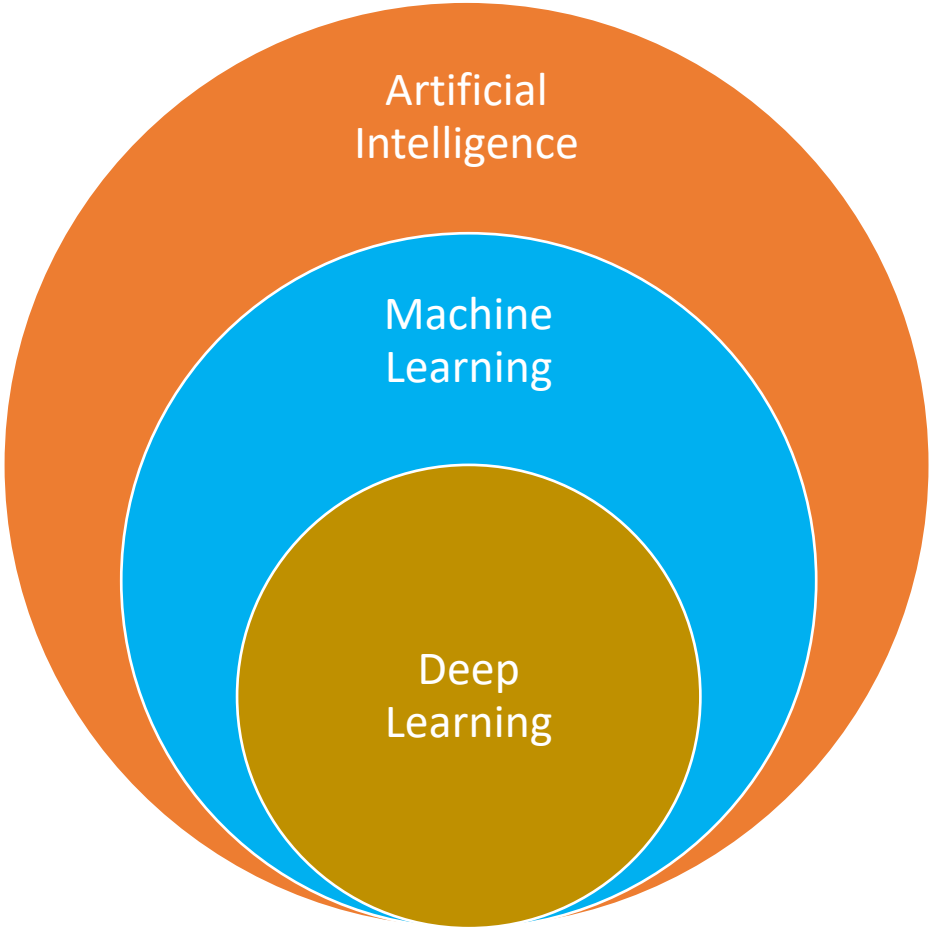
Machinery

Entertainment

Photo Source: Internet

AI Concepts

- Artificial Intelligence
  - Intelligence demonstrated or mimicked by machines
- Machine Learning
  - Statistical techniques that enable computers to use data to progressively improve performance on a specific task without being explicitly programmed
- Deep Learning
  - Machine Learning using deep (many-level) neural networks



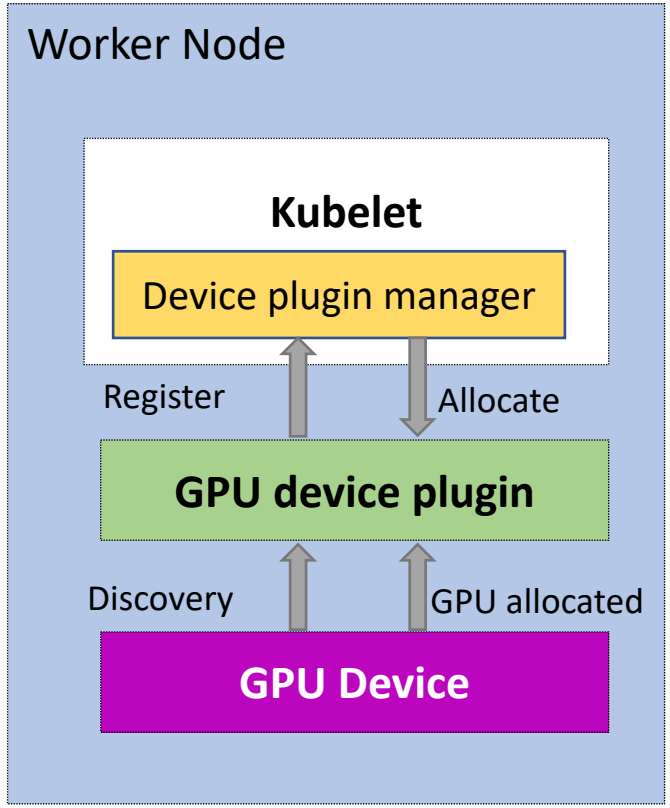
(Definitions derived from those on Wikipedia)

Kubernetes as a ML platform

- ML workload can be encapsulated in and run as container
  - Portability
  - Lightweight
- Kubernetes is the de facto standard for containerized applications
  - Scalability – distributed training, on-demand inference serving
  - Standardized workload
  - Multi-users
  - Resource management
  - Rich APIs
  - Ubiquitously available in the cloud

Using AI/ML Compute Accelerators in Kubernetes

- Device plugin for AI/ML chips
  - GPU, FPGA, ASIC
- Limitation of GPU Scheduling in Kubernetes
  - Exclusive assignment
  - No fractional assignment
- The problem of “Model Stuffing”
  - Stuff multiple models in a single container in order to share a GPU

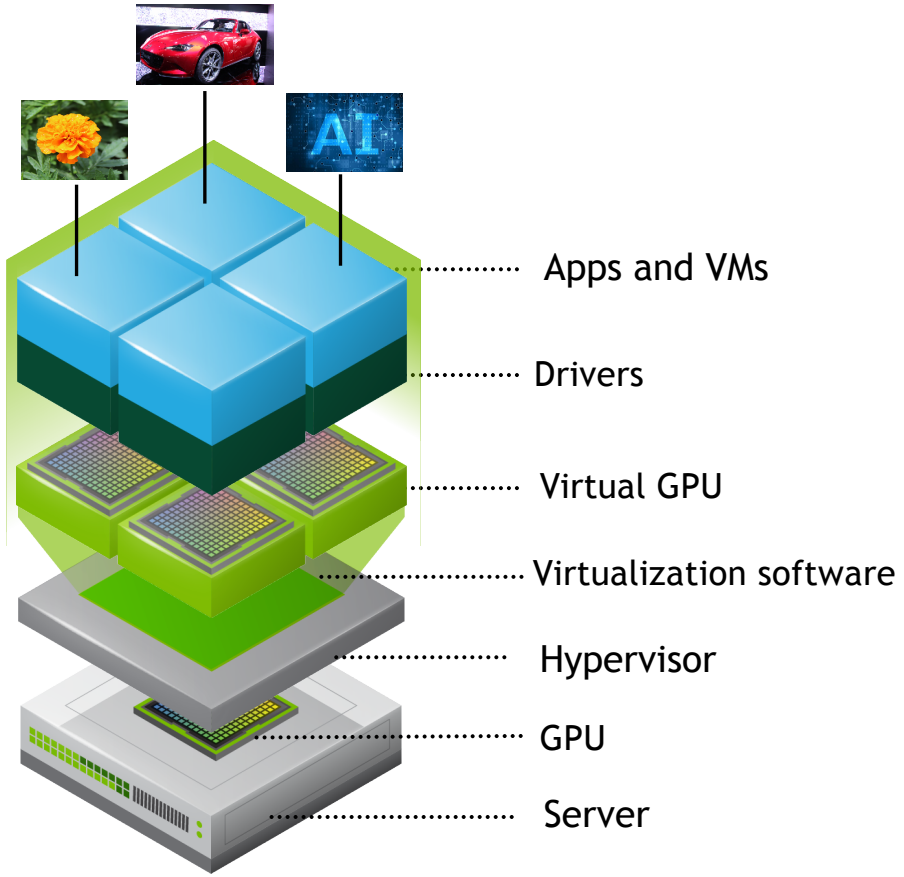


The Need of Sharing GPU in Kubernetes

- Increased utilization
  - Multiple tenants
  - Multiple types of ML workload
- Improved flexibility
  - Concurrent pipelines
  - Fine-grained GPU assignment
- Existing solutions
  - No isolation
  - No QoS

GPU Virtualization

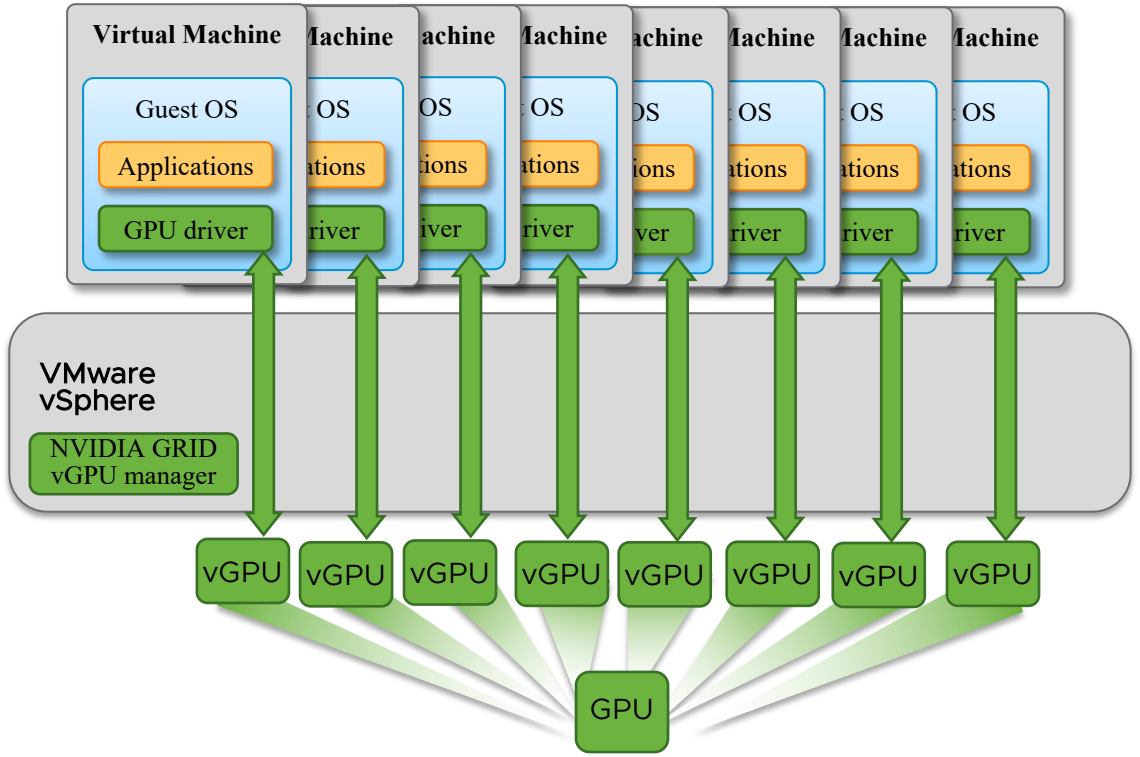
- GPU Virtualization is similar to CPU virtualization (VM)
  - NVIDIA, AMD, Intel
  - Sharing GPU resources between VMs
  - VM level isolation
  - QoS ready
- Hypervisor support
  - vSphere, KVM, Xen Server



Source: NVIDIA

vSphere using Compute Accelerators

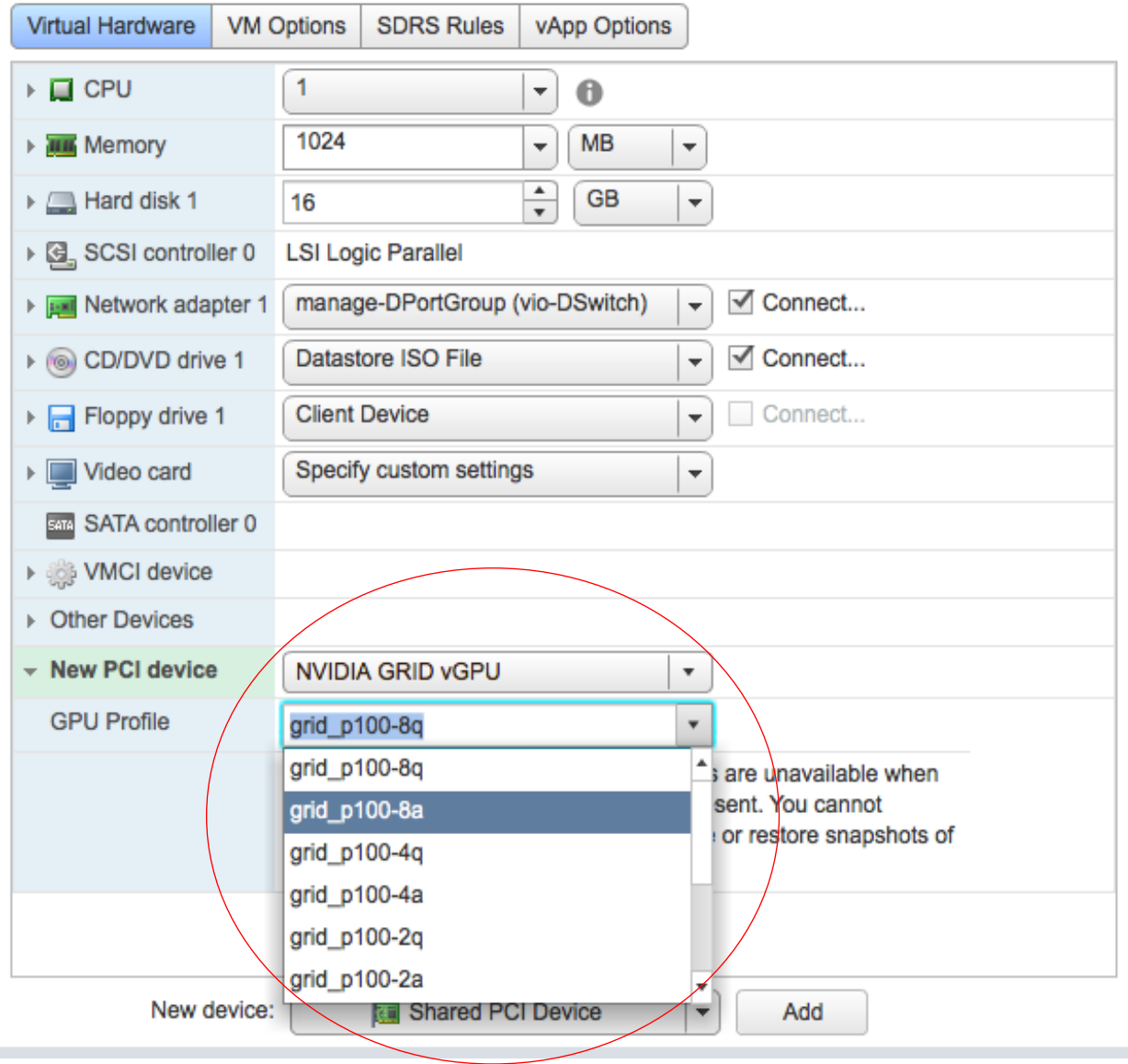
- Multiple VMs share a physical GPU
- Allow one or more vGPUs per VM
- vGPU VMs can use snapshots and vMotion



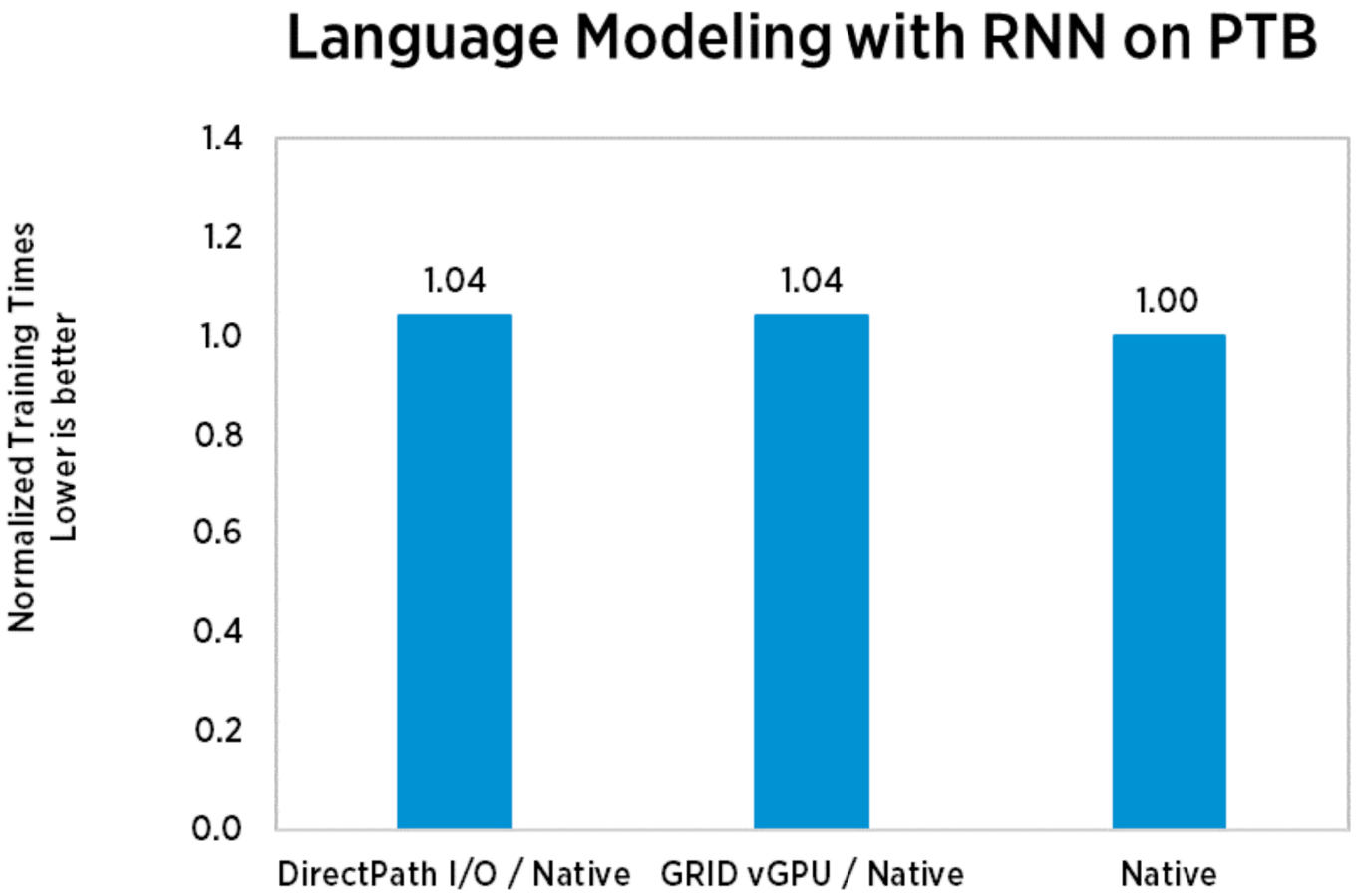
NVIDIA GRID vGPU

vGPU in vSphere

- Split the physical GPU by fixed framebuffer
- vGPUs sharing the same GPU compute engine
- Scheduling
  - Best Effort
  - Fixed Share
  - Equal Share



GPU/vGPU Performance Comparison

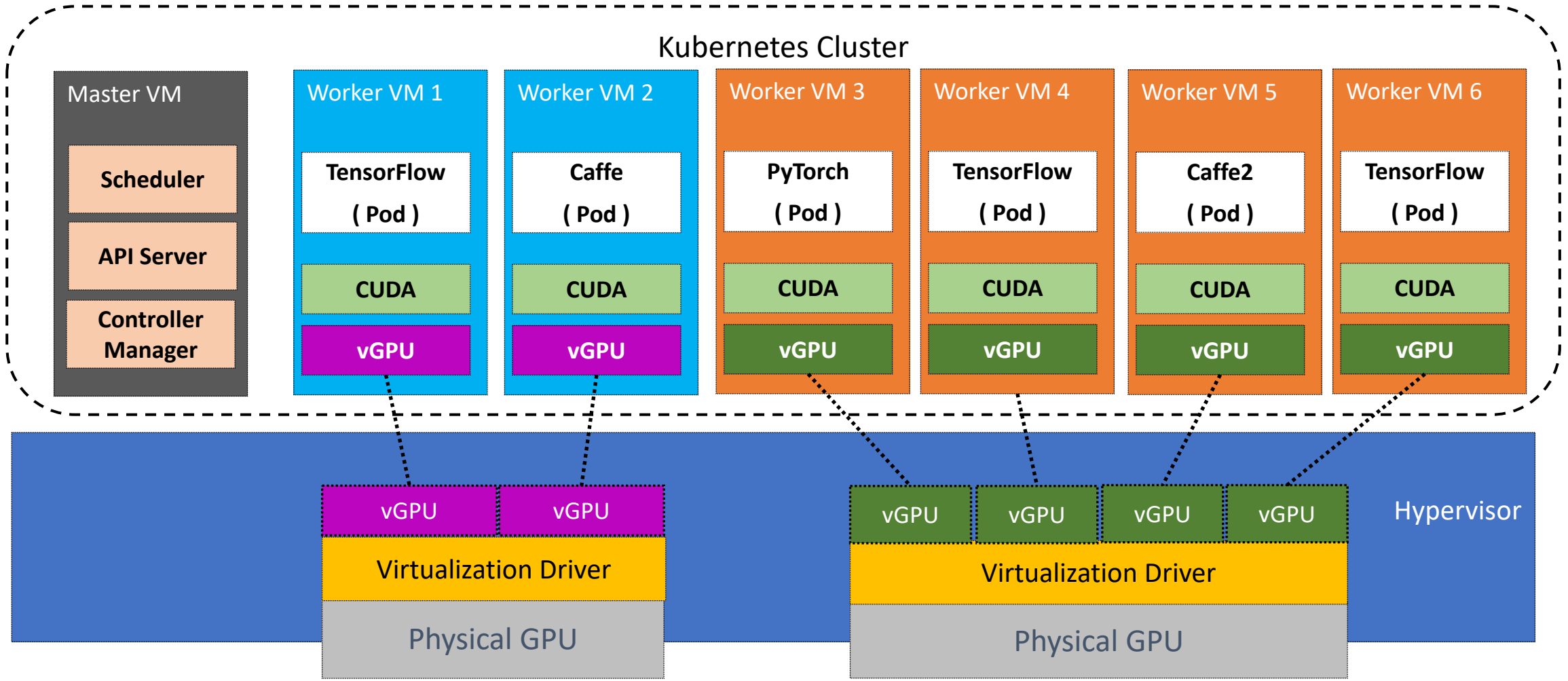


- GPU access mode
  - DirectPath I/O (passthrough)
  - vGPU
  - Native
- Little impact to performance (<4%)

Source: https://blogs.vmware.com/performance/2017/10/episode-3-performance-comparison-native-gpu-virtualized-gpu-scalability-virtualized-gpus-machine-learning.html

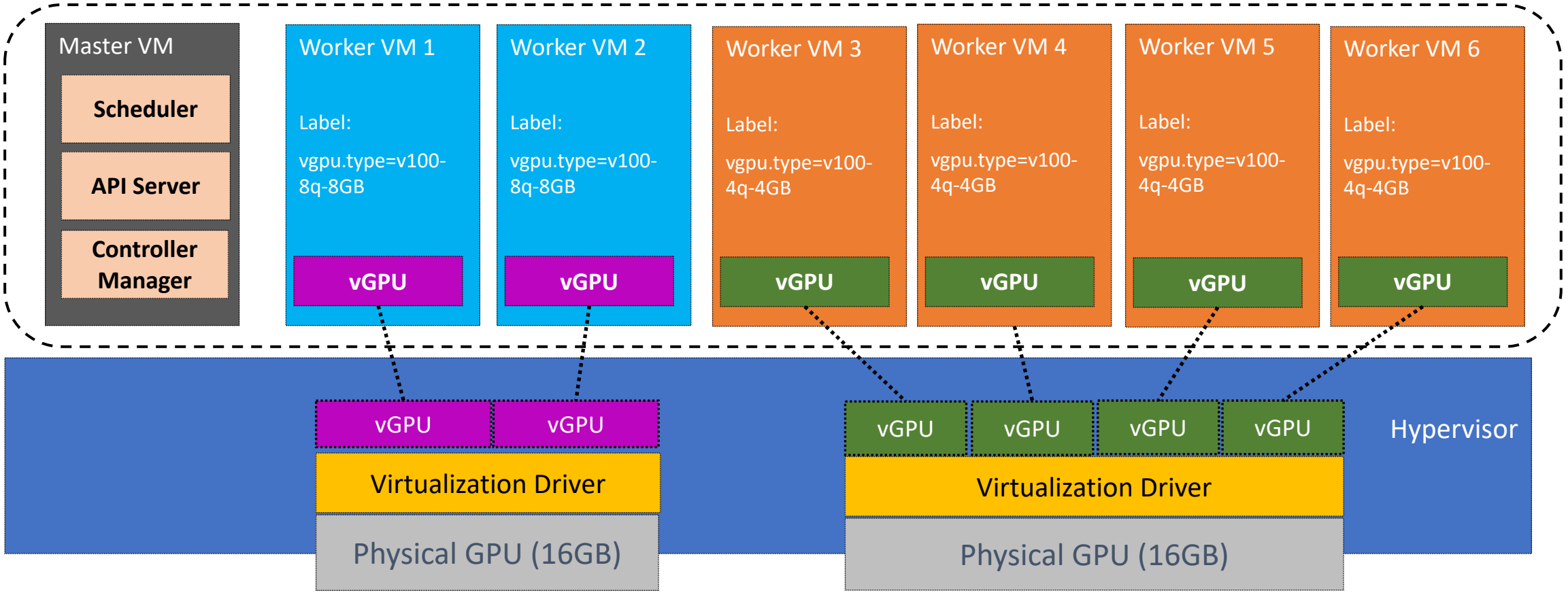
ML Jobs Sharing GPU in Kubernetes

- Flexibility, finer granularity, isolation, multi-tenant



Using Virtual GPU in Kubernetes

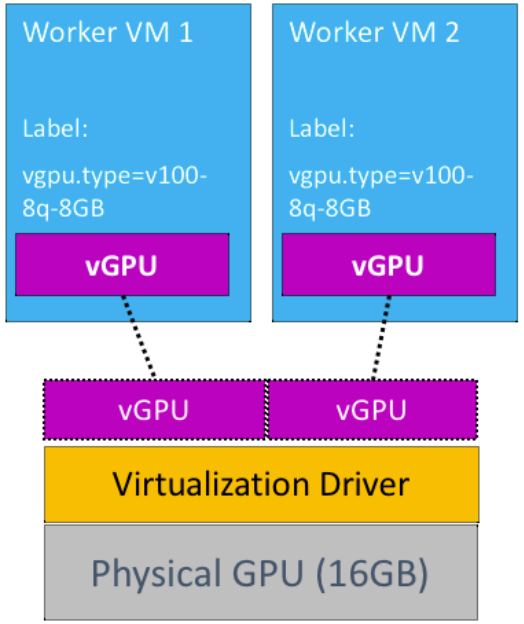
- Provision worker nodes with vGPU devices
- Label worker nodes with different vGPU profiles
  - For example: `kubectl label node <worker_node_1> vgpu.type=v100-8q-8GB`



Using Virtual GPU in Kubernetes(Cont'd)

- Pod definition
  - Request GPU resources
  - 1 GPU mapped to a fractional physical GPU
    - e.g. 1/2 GPU or 1/4 GPU
  - Node selector for scheduling to specified vGPU profile
- Isolation
- QoS

```
kind: Pod
apiVersion: v1
metadata:
  name: gpu-pod
spec:
  containers:
  - name: gpu-container
    image:
tensorflow/tensorflow:1.10.0-rc1-
gpu-py3
    imagePullPolicy: IfNotPresent
    command: ["python"]
    args: ["-u", "-c", "import
tensorflow”]
    resources:
      requests:
        nvidia.com/gpu: 1
      limits:
        nvidia.com/gpu: 1
  restartPolicy: Never
  nodeSelector:
    vgpu.type: v100-8q-8GB
```

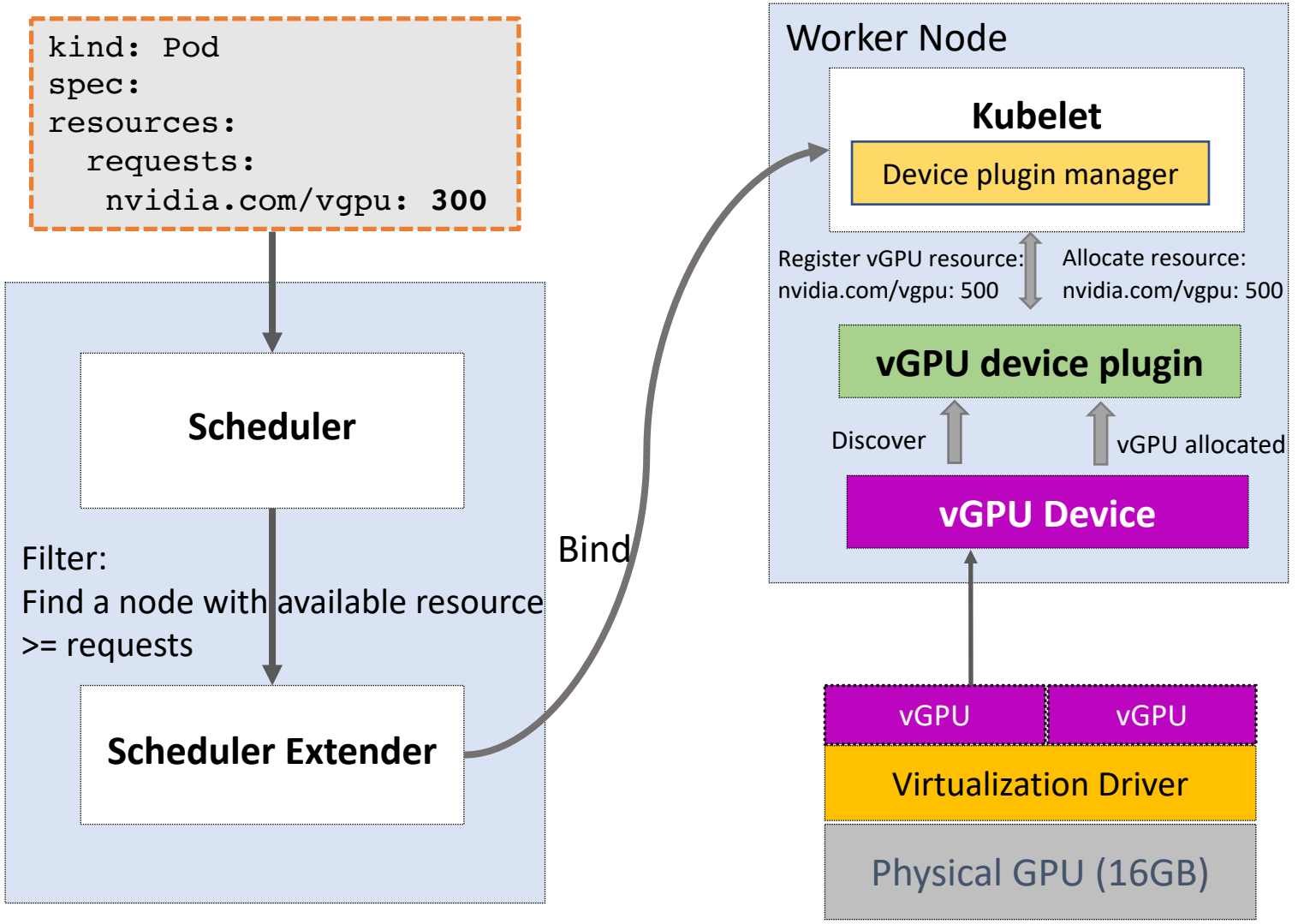


DEMO1

- Pods using and sharing vGPU

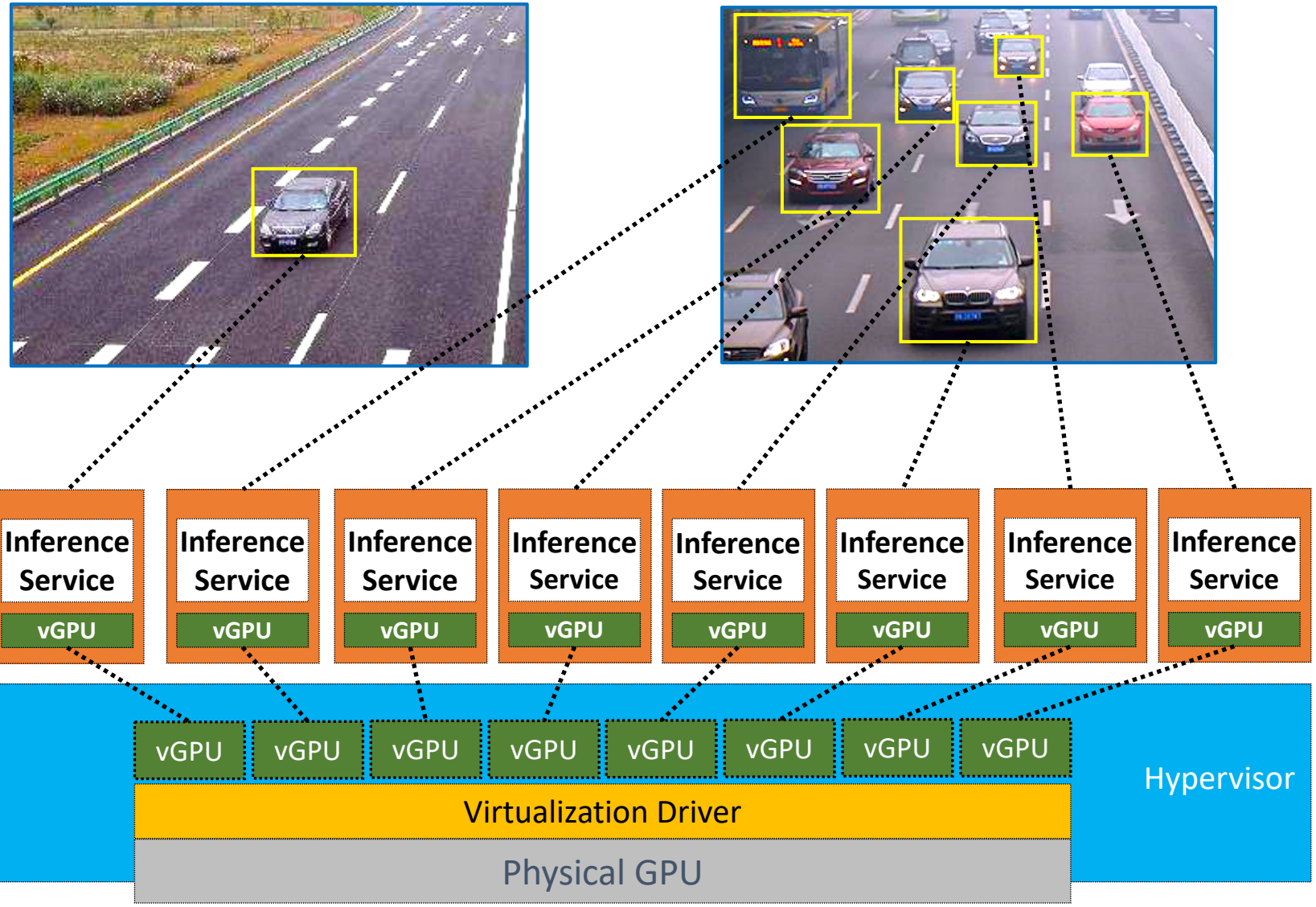
Fractional GPU in Kubernetes

- Implementing fractional GPU
  - 1000 = 1 physical GPU
  - 500 = 0.5 physical GPU
- Device plugin reports discovered extended resource
  - e.g. nvidia.com/vgpu
- Extend Kubernetes scheduler to filter extended resource



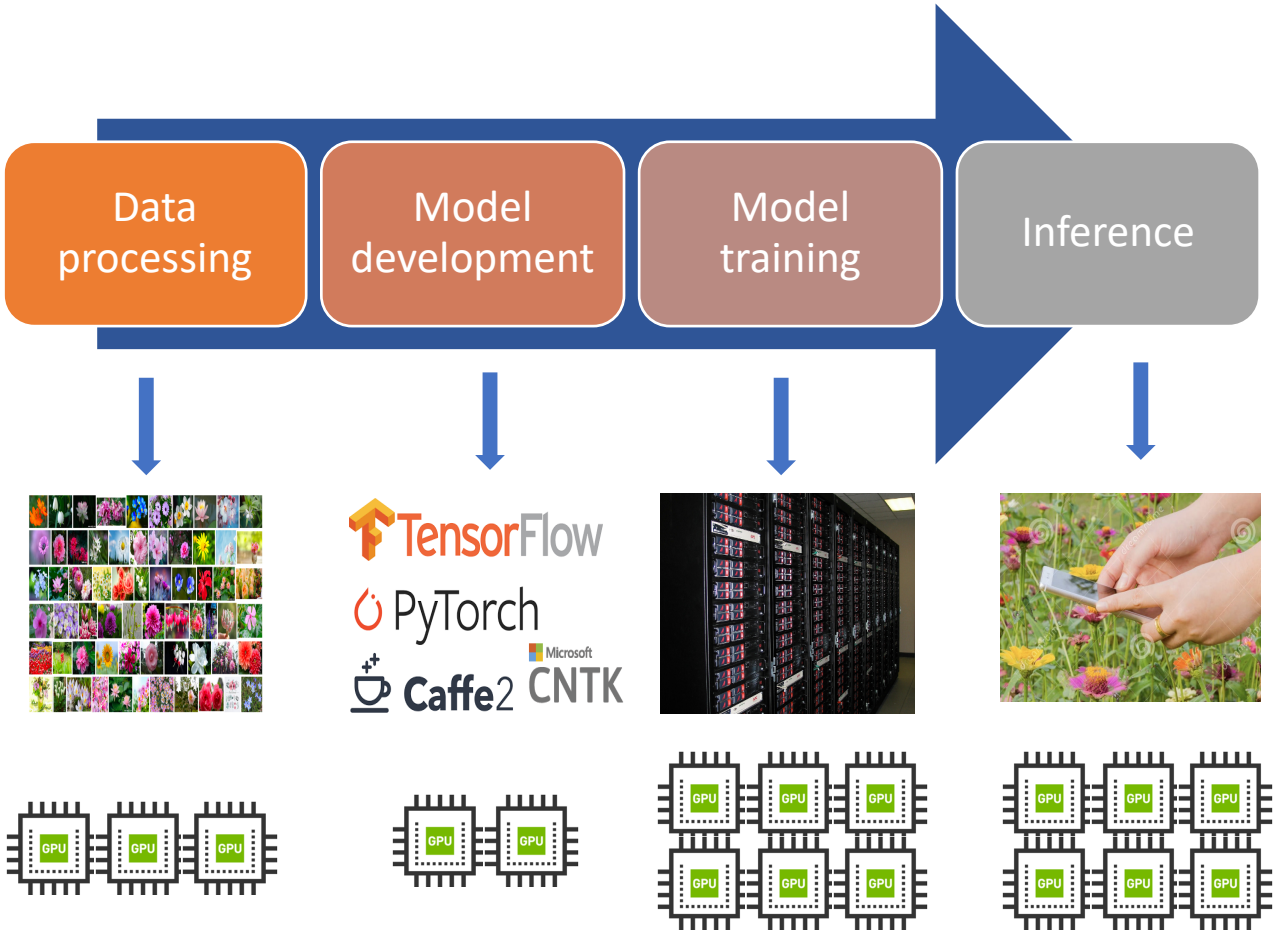
Use Case (1): Scalability

- Scaling out inference service (microservice style)
- Sharing physical GPU
- Shortened response time



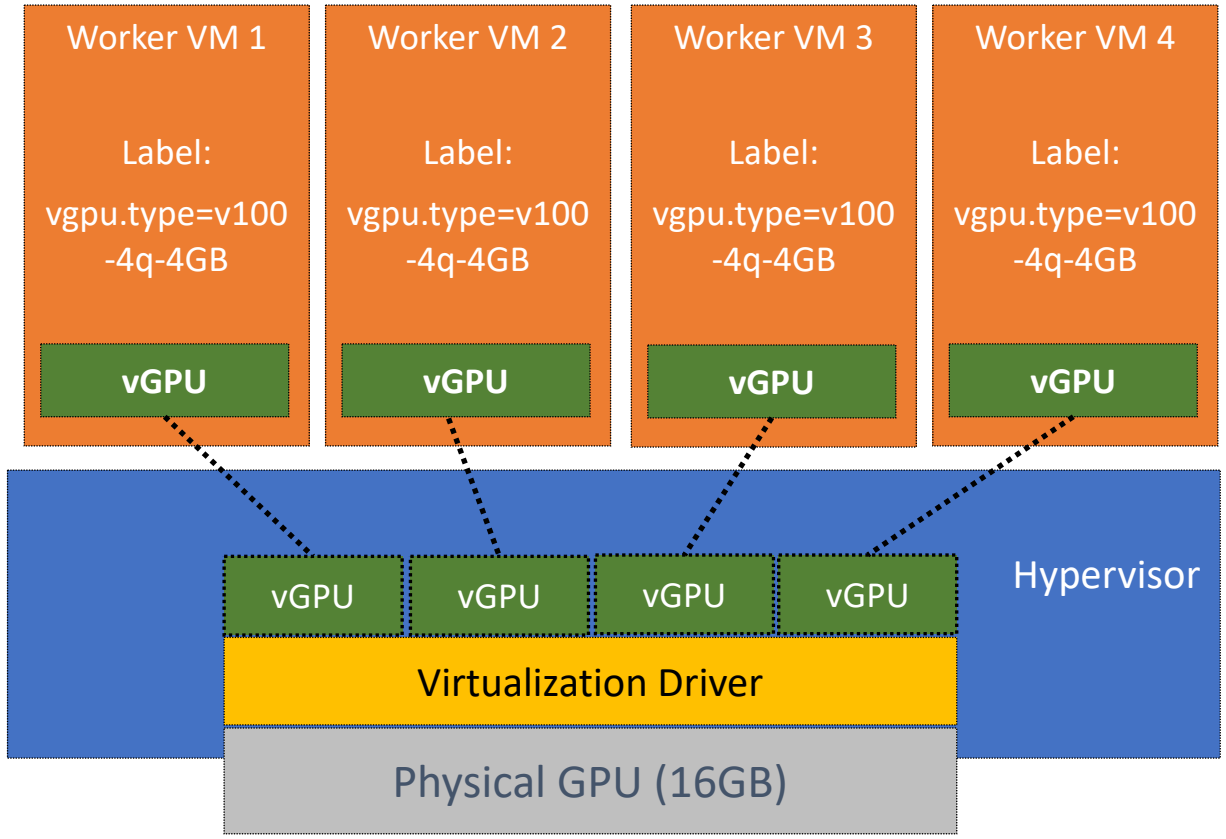
Use Case (2):Mixed ML Workload

- A university has diverse ML workload for a large group
  - Research/teaching/development
  - Training/inference/data processing
- Sharing GPUs between workloads and users
  - Utilization / density
  - Multi-tenant
  - Software-defined sharing policy



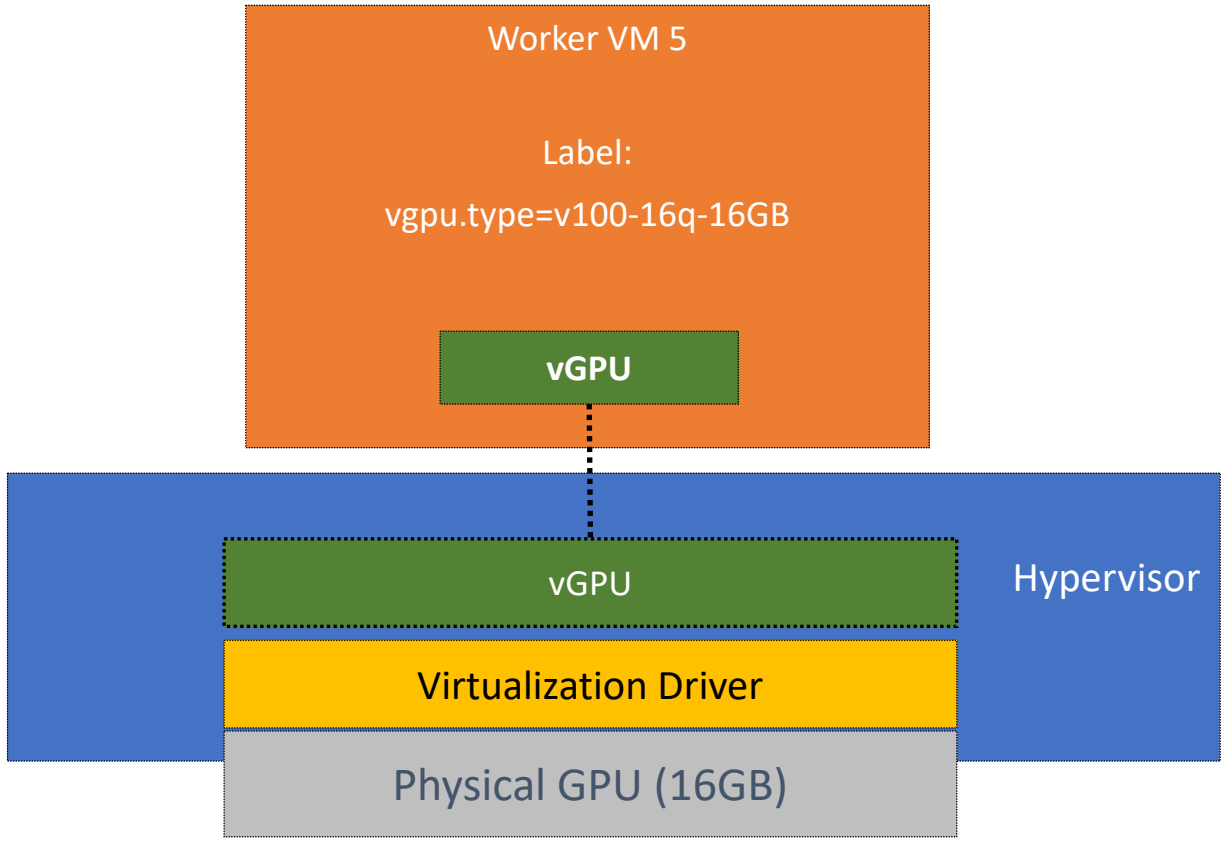
Use Case (2):Mixed ML Workload(cont'd)

- vGPU Profile switching
- Day time:
  - Worker Node 1~4 with vGPU of v100-4q-4GB



Use Case (2):Mixed ML Workload(cont'd)

- vGPU Profile switching
- Day time:
  - Worker Node 1~4 with vGPU of v100-4q-4GB
- Night time:
  - Suspend Worker Node 1~4
  - Resume Worker Node 5 with vGPU of v100-16q-16GB



DEMO2

- Switching vGPU profiles of worker nodes

Summary

- Kubernetes is suitable to run ML workload
- Kubernetes can leverage GPU virtualization for ML applications
  - Utilization
  - Scalability
  - Isolation
  - QoS
  - Suspend/ resume/live migration/snapshot/cloning
- Reference
  - https://github.com/yuyangbj/K8vGPUSharing

Thank you!