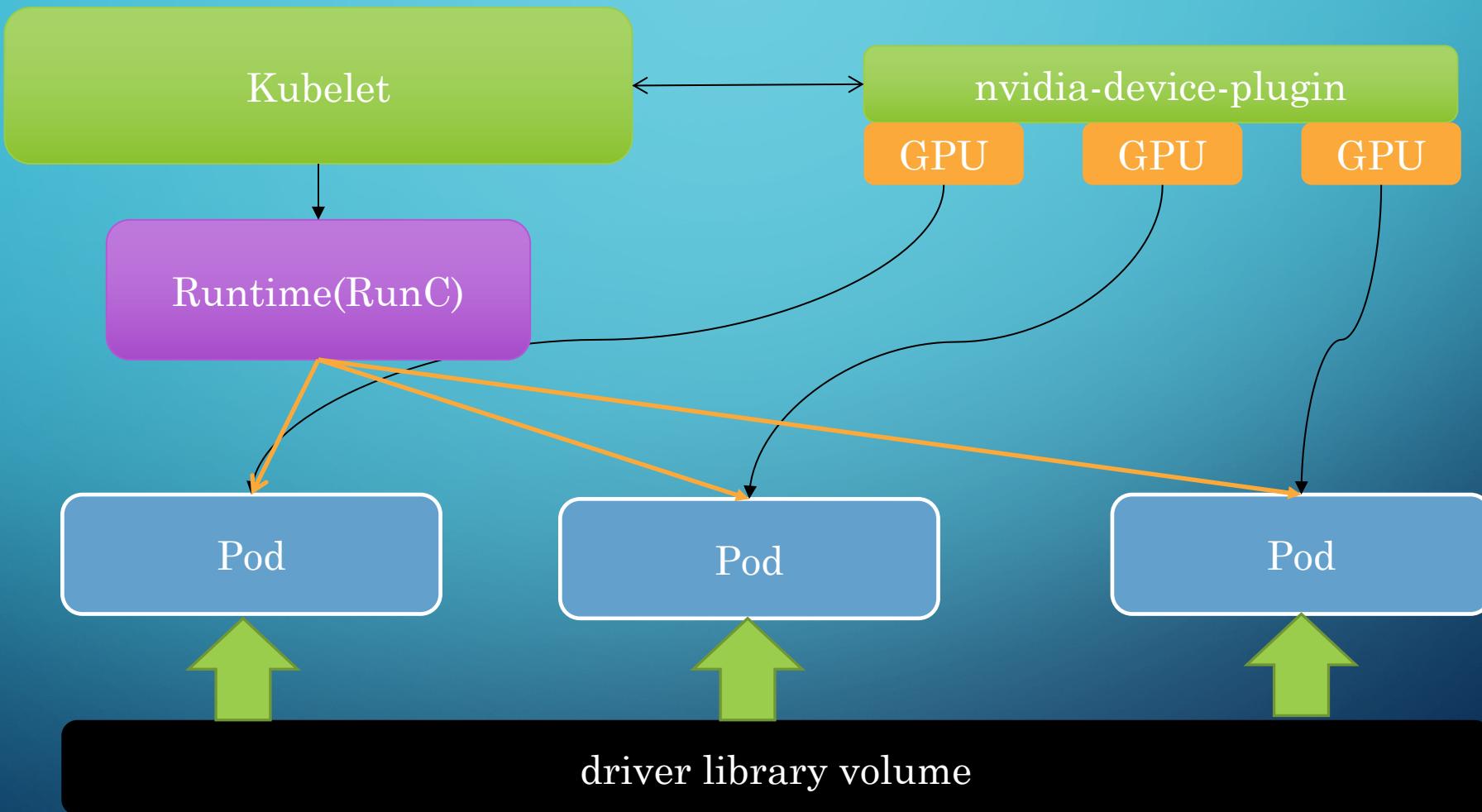


# 通过kata容器在kubernetes集群中支持vGPU

周威（腾讯IEG） 2020/11



# 原有GPU架构



# GPU的虚拟化

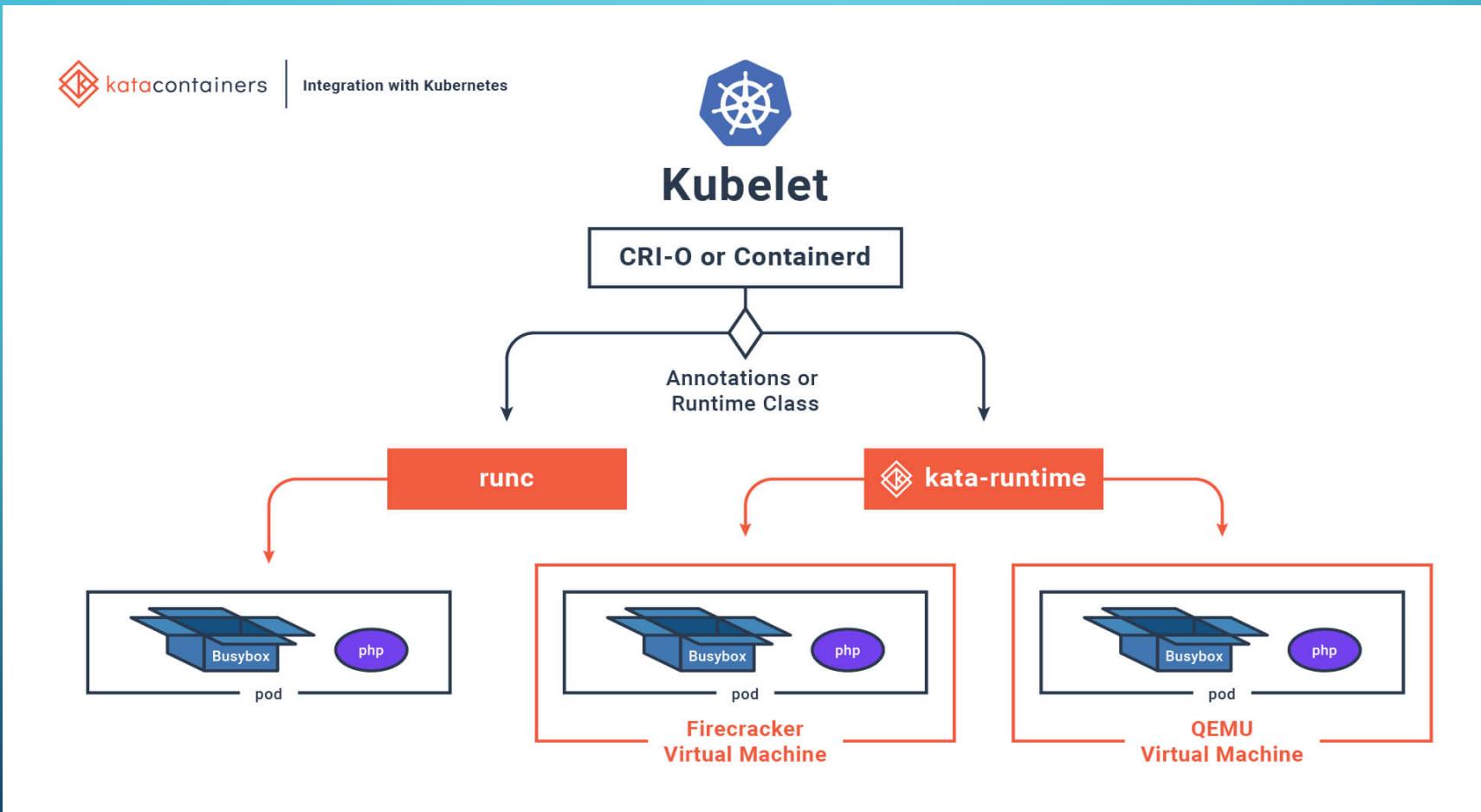
根据我们的监控数据，很多情况下用户申请了GPU资源却并不能将GPU用满，而GPU作为一种宝贵的计算资源，如果能通过一些手段让多个用户共享一块GPU卡的话无疑能节省很多成本。现有的GPU虚拟化方案主要有MPS和GRID两种

方案	MPS	GRID
原理	容器中的多个cuda进程共享GPU context，母机上运行守护进程进行管理	GRID通过vfio技术创建了一个资源隔离的vGPU，然后将vGPU分配给vm实例
优势	直接支持RunC容器，改造成本小	1.成熟稳定，公有云GPU虚拟化大部分使用该方案 2.支持GPU按资源隔离
缺点	1.无法隔离资源使用率(Volta+) 2.由于共享了GPU context，一旦出错会影响所有实例	1.只支持虚拟机(kvm) 2.只能按照固定配置进行切分

# Why not KubeVirt?

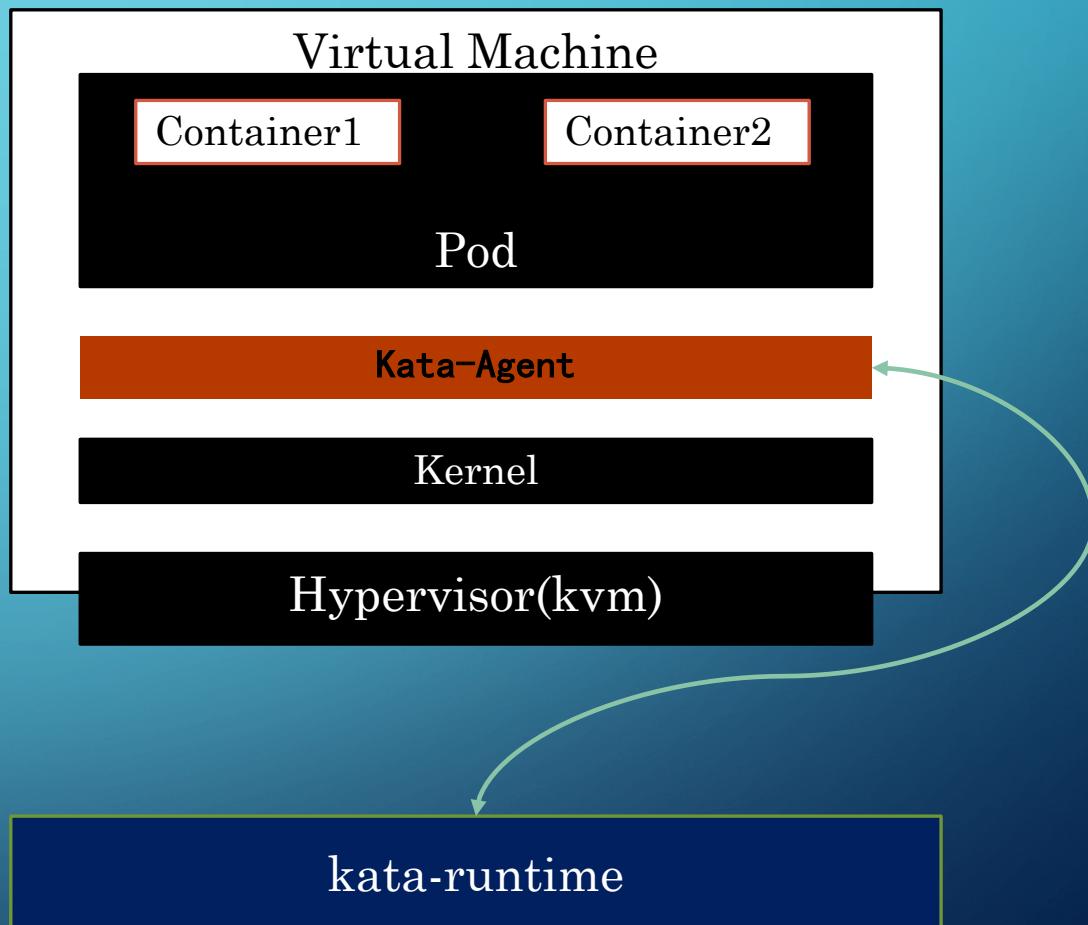
- 有一个官方提供的vGPU插件
- 我们希望使用的是容器(随起随停, 使用image), 而不是VM
- 业务已经熟悉kubernets的各种workload (pod, deployment...)
- KubeVirt 中 vGPU需要预先创建, 并且无法更改类型

# Kata与CRI接口

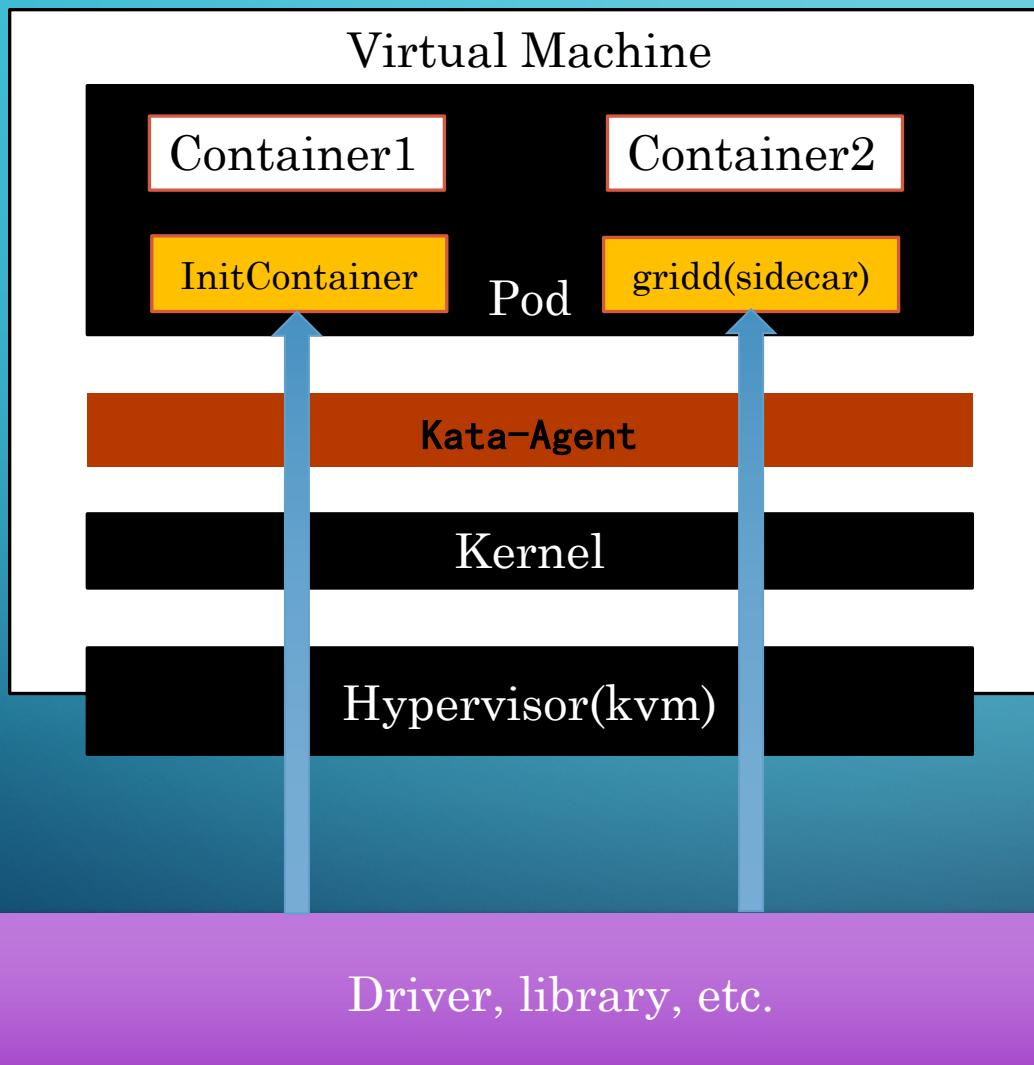


# How Kata works?

官方的kata容器并不能直接支持vGPU，一个原因是Hypervisor提供了一个独立的内核空间，所以需要在这一层集成vGPU驱动，另一个需求是，vGPU需要一个常驻进程nvidia-gridd来获取license

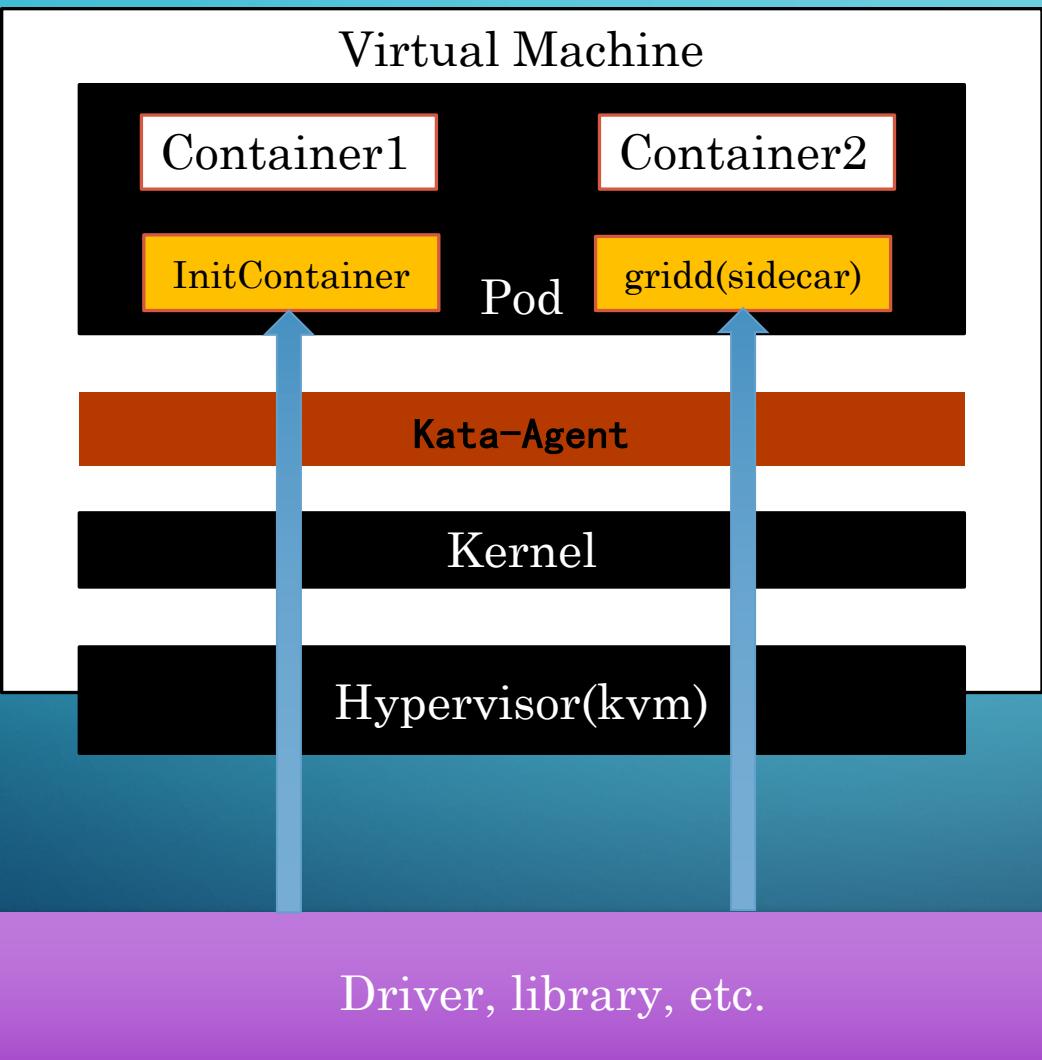


# How Kata works?



```
./nvidia-installer \
--ui=none -q \
--kernel-source-path=%{KERNEL_SRC} \
--utility-prefix=%{buildroot}/%{NVIDIA_LIBRARY} \
--utility-libdir=lib64 \
--x-prefix=%{buildroot}/%{NVIDIA_LIBRARY} \
--x-library-path=%{buildroot}/%{NVIDIA_LIBRARY}/lib64 \
--opengl-prefix=%{buildroot}/%{NVIDIA_LIBRARY} \
--kernel-install-
path=%{buildroot}/%{NVIDIA_LIBRARY}/lib/modules/%{KERNEL_
VERSION}-%{kernel_suffix}/kernel/drivers/video \
--opengl-libdir=lib64 \
--override-file-type-
destination="NVIDIA_MODPROBE:%{buildroot}/%{NVIDIA_LIBRAR
Y}/bin" \
-k %{KERNEL_VERSION}-%{kernel_suffix} \
--no-distro-scripts \
--no-check-for-alternate-installs \
--no-install-libglvnd \
--no-kernel-module-source \
--install-compat32-libs \
--compat32-prefix=%{buildroot}/%{NVIDIA_LIBRARY} \
--compat32-libdir=lib \
cp gridd.conf.template
%{buildroot}/%{NVIDIA_LIBRARY}/etc/nvidia/gridd.conf
```

# How Kata works?



## initContainers:

- name: nvidia  
image: busybox  
command:
  - /bin/sh
  - -c
  - "modprobe nvidia;modprobe nvidia-drm modeset=1;nvidia-modprobe -u -m"

## resources:

### limits:

tencent.com/vgpu-492: 1

### securityContext:

#### capabilities:

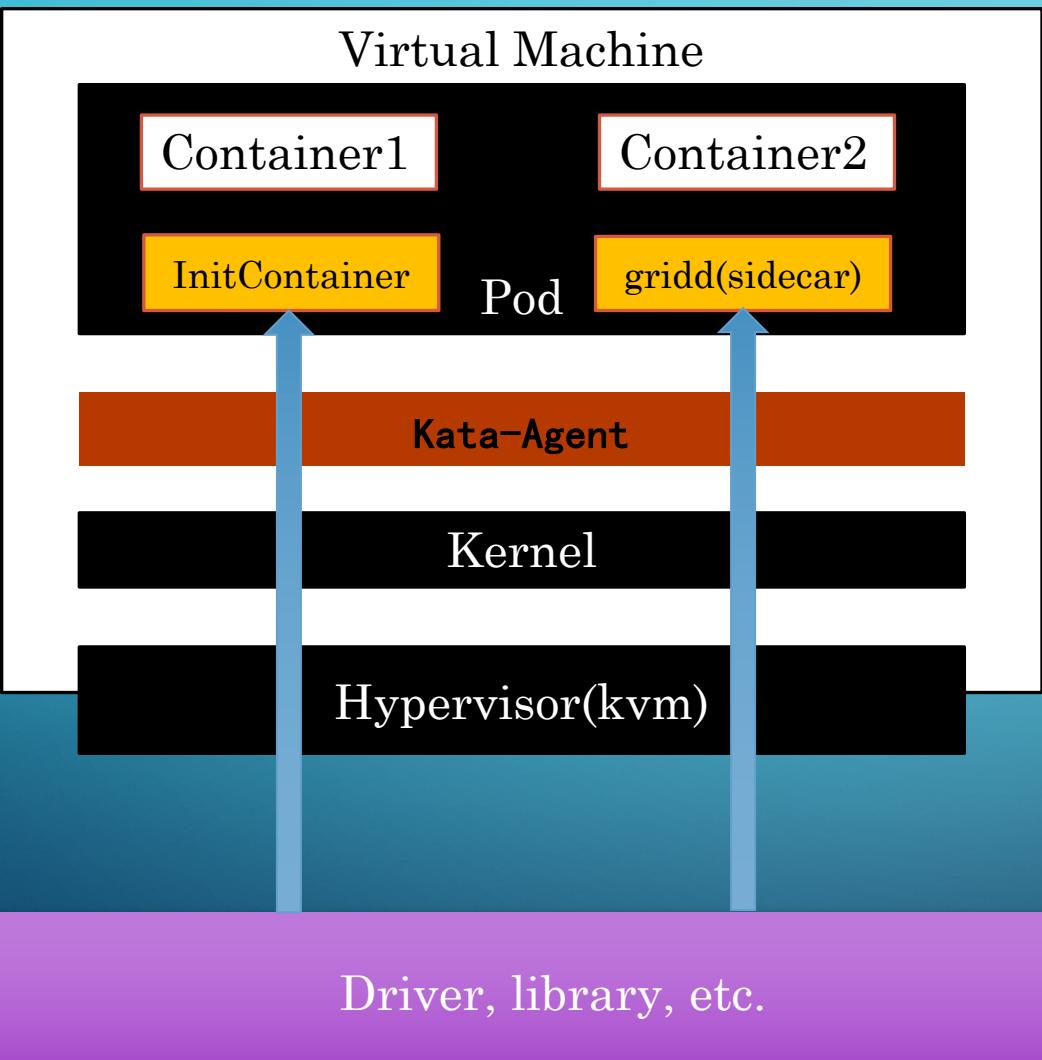
##### add:

- all

### volumeMounts:

- name: dev  
mountPath: /dev
- name: nvidia-driver  
mountPath: /lib/modules  
subPath: lib/modules  
readOnly: true
- name: nvidia-driver  
mountPath: /var/lib/nvidia  
readOnly: true

# How Kata works?



## containers:

- name: gridd
- image: centos:7
- command:
  - /bin/bash
  - -c
  - "nvidia-gridd; sleep infinity"

## securityContext:

### capabilities:

- add:
  - all

## volumeMounts:

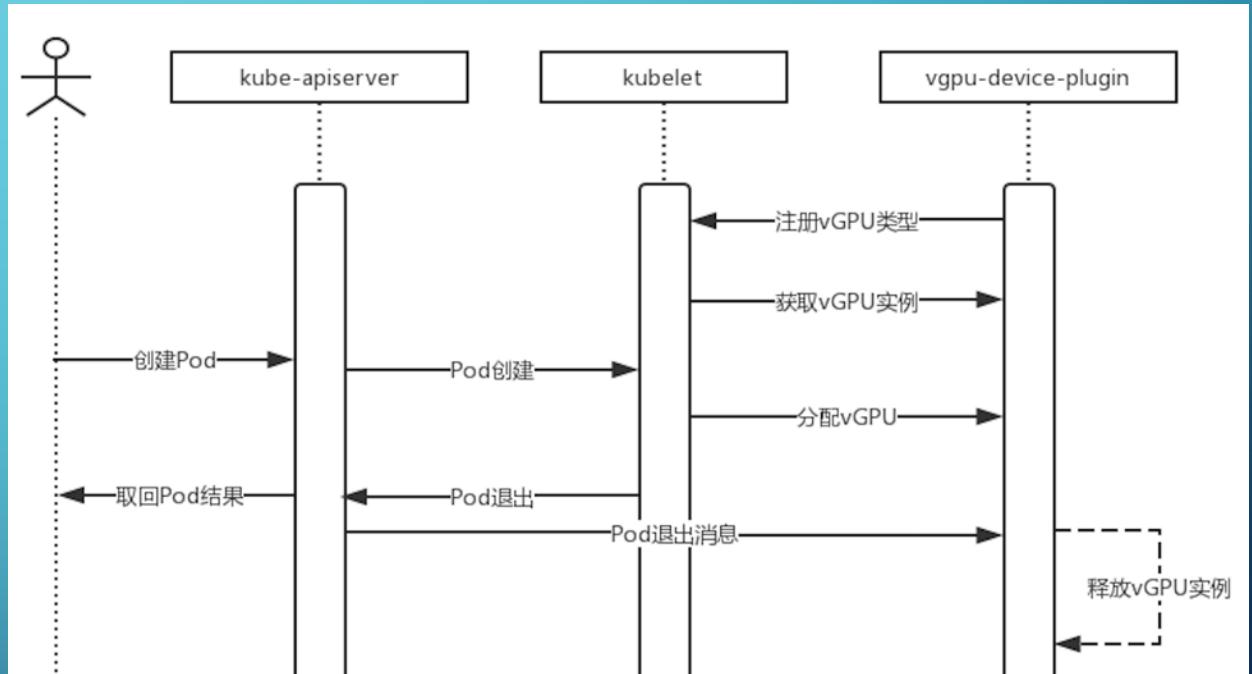
- name: dev
- mountPath: /dev
- name: nvidia-driver
- mountPath: /usr/share/nvidia
- readOnly: true
- name: nvidia-driver
- mountPath: /etc/nvidia
- subPath: etc/nvidia
- readOnly: true

# Mutating Admission Webhook

Mutating Admission Webhook是一种准入控制器，可以用来修改发送到apiserver的对象，我们可以用它来对使用vGPU的Pod进行修改，自动插入InitContainer和sidecar容器（也可以用于给用户容器插入volume），从而做到对用户无感知，可以开箱即用。

# vgpu device plugin

GRID通过vfio技术虚拟出不同规格的vGPU后，kubernetes需要将这些vGPU分配给不同的容器。这里就会涉及到vGPU的调度过程，我们专门开发了一个vgpu-device-plugin用以管理vGPU。



# device plugin 接口

```
service DevicePlugin {
    // ListAndWatch returns a stream of List of Devices
    // Whenever a Device state change or a Device disappears, ListAndWatch
    // returns the new list
    rpc ListAndWatch(Empty) returns (stream ListAndWatchResponse) {}

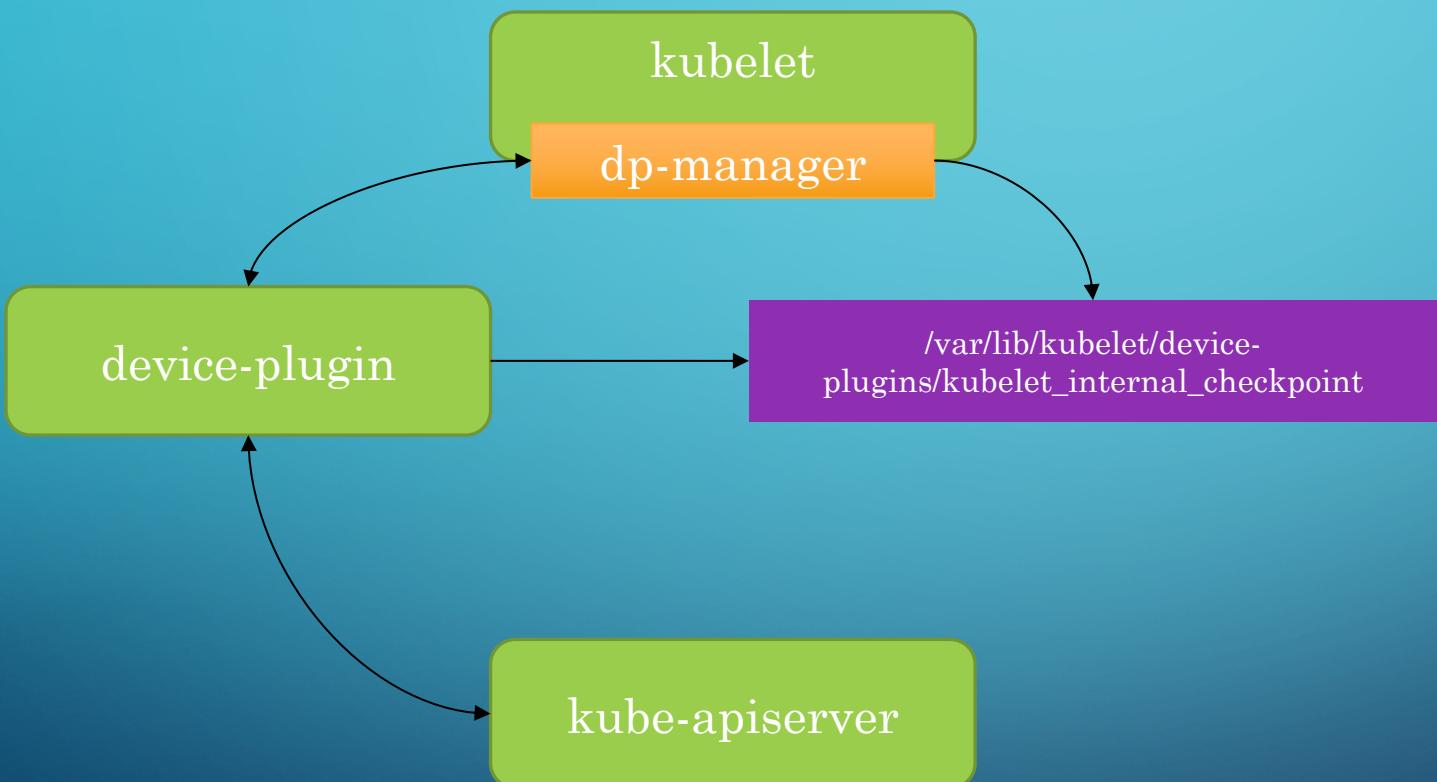
    // Allocate is called during container creation so that the Device
    // Plugin can run device specific operations and instruct Kubelet
    // of the steps to make the Device available in the container
    rpc Allocate(AllocateRequest) returns (AllocateResponse) {}
}
```

这个接口让我们遇到了下面几个困难

- 在List请求上报设备数量时，vGPU设备是还没有创建的
- device-plugin不知道设备何时会被销毁，以及分配给了哪个容器
- kubelet会重用销毁了Pod的vGPU，而不会再通知device-plugin

我们认为这是dp接口的设计缺陷，因此用了几个workaround来解决它

# 获取设备和vGPU的绑定关系



device-plugin 通过监听 device plugin manager 生成的 checkpoint 文件的变化来获取 vGPU 和 pod 的对应关系，然后再通过 apiserver 获取该 pod 的销毁事件，从而判断 vGPU 的销毁时机。

# vGPU信息注册

```
[root@TENCENT64 ~]# ls /sys/class/mdev_bus/0000\:81\:00.0/mdev_supported_types/ -lh
total 0
drwxr-xr-x 3 root root 0 Nov  3 15:44 nvidia-277
drwxr-xr-x 3 root root 0 Nov  3 15:44 nvidia-278
drwxr-xr-x 3 root root 0 Nov  3 15:44 nvidia-279
drwxr-xr-x 3 root root 0 Nov  3 15:44 nvidia-490
drwxr-xr-x 3 root root 0 Nov  3 15:44 nvidia-491
drwxr-xr-x 3 root root 0 Nov  3 15:44 nvidia-492
```

Capacity:

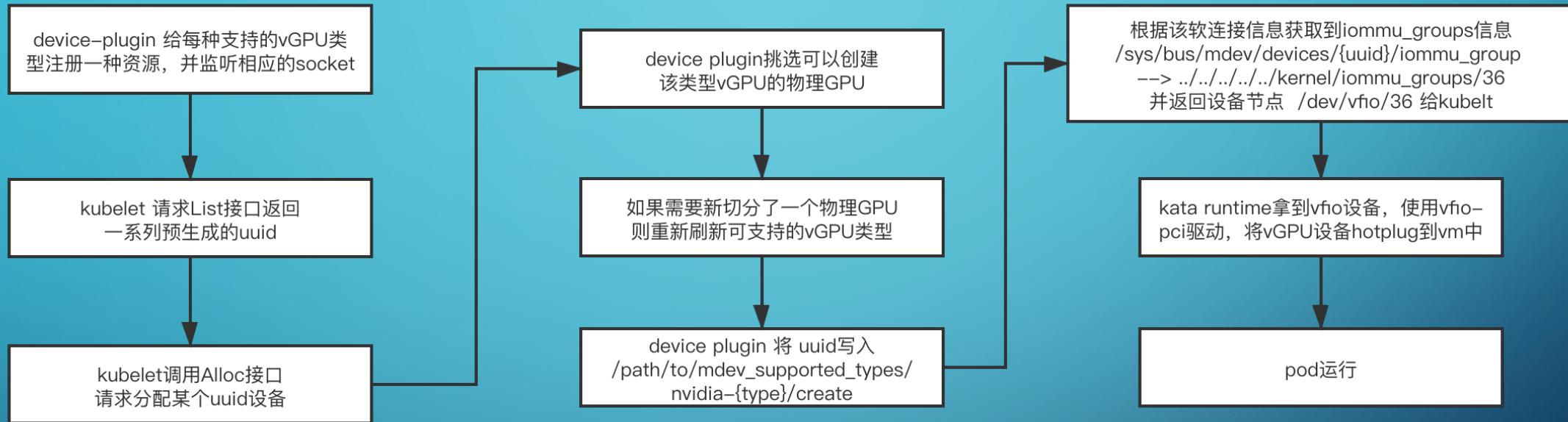
cpu:	128
ephemeral-storage:	735871936Ki
memory:	526233376Ki
pods:	61
tencent.com/vgpu-277:	1
tencent.com/vgpu-278:	1
tencent.com/vgpu-279:	1
tencent.com/vgpu-490:	1
tencent.com/vgpu-491:	1
tencent.com/vgpu-492:	1



Capacity:

cpu:	128
ephemeral-storage:	735871936Ki
memory:	526233376Ki
pods:	61
tencent.com/vgpu-277:	0
tencent.com/vgpu-278:	0
tencent.com/vgpu-279:	0
tencent.com/vgpu-490:	0
tencent.com/vgpu-491:	0
tencent.com/vgpu-492:	32

# 创建与销毁



销毁流程比较简单，device plugin 监听到pod销毁事件，则将分配给它的vGPU销毁掉，通过写入该路径来完成：  
`/sys/bus/mdev/devices/{uuid}/remove`



## 总结

通过这些手段，开发者可以直接通过创建kubernetes工作负载的方式来实现GPU虚拟化功能。

相比于传统的vGPU使用方式，借助kata，我们可以减少额外的资源开销，并且简化开发流程，这些都有助于提升我们整体的GPU使用率。