Device Assignment for VMs in Kubernetes

Martin Polednik (@mpolednik)
Software Engineer @ Red Hat
$ whoami

• Golang, Python engineer
• working on oVirt and KubeVirt
• node/host management level virtualization tech
  • device assignment w/ VFIO, (v)GPU, SR-IOV
  • NUMA, hugepages, CPU architectures
• https://mpolednik.github.io/
The Stack

- VM device assignment (VFIO)
- libvirt
- Docker
- Kubernetes
- KubeVirt
Devices & Virtualization
What even is a device?

- many memory regions!
- /sys/bus/pci/${device_address}/...
- /dev/...
VFIO 101

• PCI driver
• devices bound to it can be used in VMs
• IOMMU groups based on DMA isolation
• explained in Slicing a (v)GPU talk at DevConf.cz
  • https://www.youtube.com/watch?v=G8b9jI FN-nk
IOMMMU Groups

• group contains 1-N devices
• assignment granularity at group level
• e.g. GPU + HDMI sound card
• accessed at /dev/vfio/${N}
libvirt

- daemon & library for single-node VM management
- abstracts QEMU cmdline interface by XML
- refers to devices by their PCI address
...<devices>
  ...
  <hostdev managed="no" mode="subsystem" type="pci">
    <source>
      <address bus="7" domain="0" function="0" slot="0"/>
    </source>
  </hostdev>
  ...
...<devices>
  ...

Devices in Containers
Overview

• no special driver needed

• device path exposed to container
  • --device, --volume (?), --privileged (!?)
  • DRI, toolkits, any required endpoints

• also sets up cgroups
Overview

• sufficient unless orchestration is needed

• ... in that case, building block for Kubernetes device assignment
Devices in Kubernetes
Kubernetes 101

• orchestrate containers (in declarative way)

• pod = several containers

• pod, container, node etc. are just resources

• the talk will show resources in YAMLs
NVIDIA GPUs

- vendor-specific feature since 1.3
- `accelerators` FeatureGate
- request N GPUs
NVIDIA GPUs

spec:
  containers:
    - name: demo
      ...
  resources:
    requests:
      alpha.kubernetes.io/nvidia-gpu: 2
NVIDIA GPUs

- deprecated by device plugins
Device Plugins

- since Kubernetes 1.8
- shortened to DPI(s)
- gated behind `DevicePlugins` FeatureGate
- gRPC server(s) that exposes available resources
  - Register, Allocate, ListAndWatch
Device Plugins

- one gRPC server per tracked resource
func (dpm *DevicePluginManager) Run() {
    for _, plugin := range dpm.plugins {
        go plugin.Run()
    }
    <-dpm.stopCh
    dpm.stop()
fancy starting 50+ gRPC servers?
$ sh kubectl.sh get nodes --show-all -o json | grep -A 10 allocatable

"allocatable": {
  "cpu": "4",
  "hugepages-1Gi": "0",
  "hugepages-2Mi": "0",
  "memory": "12181600Ki",
  "mpolednik.github.io/102b_0522": "1",
  "mpolednik.github.io/111d_8018": "3",
  "mpolednik.github.io/8086_10c9": "2",
  "mpolednik.github.io/8086_10e8": "4",
  "mpolednik.github.io/8086_244e": "1",
  "mpolednik.github.io/8086_2c70": "1",
  ...
}
apiVersion: v1
kind: Pod
metadata:
  name: nginx-apparmor
spec:
  containers:
  - name: nginx
    image: nginx
  resources:
    requests:
      mpolednik.github.io/8086_10e8: 1
    limits:
      mpolednik.github.io/8086_10e8: 1
Device Plugins

• flexible
  • allows the node to advertise any resource
    • /dev/kvm is a device too!
    • and mount it into a container (not pod!)
  • still in development

• Deallocate gRPC endpoint?
KubeVirt
KubeVirt

• (not only) pet VMs in Kubernetes
• uses CRD (custom resource definition)
  • and several custom services
• based on libvirts
Devices in KubeVirt

- mix of both worlds
  - Kubernetes assignment for devices
  - VFIO within the (docker) container
  - requires custom DPI
  - + VM spec to POD spec translation
VFIO DPI

https://github.com/kubevirt/kubernetes-device-plugins (WIP)
VFIO DPI

- ensure vfio-pci is loaded
- enumerates /sys/bus/pci/devices
- for each device found
  - get vendor ID, device ID, IOMMU group
  - report it back to Kubelet (via gRPC API)
VFIO DPI

- the missing parts:
  - IOMMU group awareness (report conflicting groups as unhealthy? + DPI topology)
  - device deallocation (inotify VFIO endpoint?)
  - edge case handling (Kubelet dies, device plugin dies)
Bridging VMs and pods
What We Have (idea)

```yaml
spec:
  domain:
    devices:
      ...
      passthrough:
        - type: pci
          vendor: 1000
          device: 1000
          ...
      ...
  memory:
```
What We Need (reality)

```
spec:
  containers:
    - name: demo
  ...
  resources:
    requests:
      mpolednik.github.io/1000_1000: 1
    limits:
      mpolednik.github.io/1000_1000: 1
```
VFIO Initializer

- [https://github.com/mpolednik/k8s-vfio-initializer-plugin](https://github.com/mpolednik/k8s-vfio-initializer-plugin) (WIP)

- transform VM requirements to pod
  - in Kubernetes-native way

- probably not needed after all
That's it!*  
* almost
Is that really all?

• which devices inside pod belong to the VM?

• remember libvirt addressing?

• mount

  • /sys

  • /sys/bus/pci/devices/${device_address}

• something else?
Devices in KubeVirt

- proposal @ https://github.com/kubevirt/kubevirt/pull/593
- DPI @ https://github.com/kubevirt/kubernetes-device-plugins
- Initializer @ https://github.com/mpolednik/k8s-vfio-initializer-plugin
- comments & suggestions welcome!
Summary

- VMs in Kubernetes are real!
- and so is device assignment
Questions?

Thank you!
Slides & Blog @ https://mpolednik.github.io/