Let's run GPU virtualization in SUSE Linux

Liang Yan (lyan)
SUSE Labs

Outline

1. Background
   GPU
   Virtualization
2. GPU virtualization
   Definition and Classification
   Use scenario
3. Critical techniques
   SRIOV vs MDEV
4. Demo
5. Current status and Todo
   SUSE
   Outside Provider
6. Q&A

Background

GPU Graphic Process Unit

- 1980’s – No GPU. PC used VGA controller
- 1990’s – Add more function into VGA controller
- 1997 – 3D acceleration functions:
  Hardware for triangle setup and rasterization
  Texture mapping
  Shading
- 2000 – A single chip graphics processor (beginning of GPU term)
- 2005 – Massively parallel programmable processors
- 2007 – CUDA (Compute Unified Device Architecture)

Choice: AMD, Intel, Nvidia

GPU Purpose

Graphic Render

3D hardware acceleration

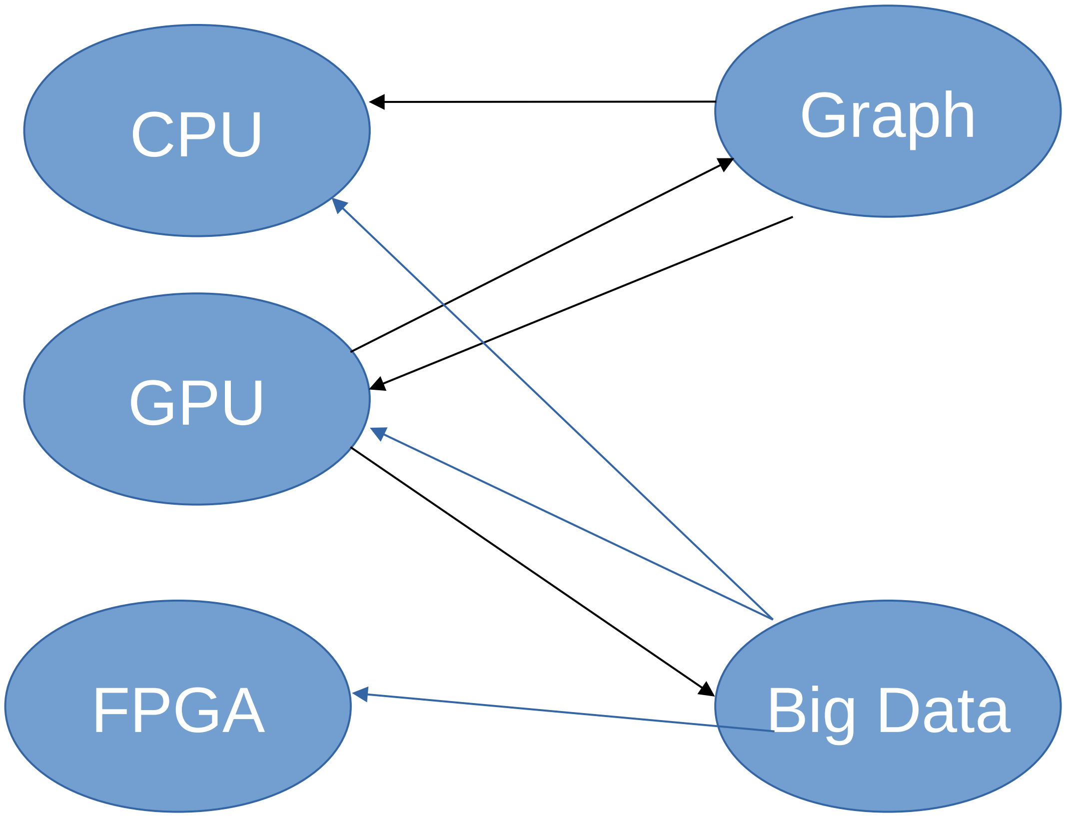
- DirectX
- OpenGL
- Vulkan

General Compute

Big Data, Machine Learning: Tensor-flow, Caffe2

CUDA	Compute Unified Device Architecture
OpenCL	Open Computing Language

Everything depends



What is really necessary for Big Data?
Data has pattern for some specific instruction, Hardware could optimize it.

GPU Structure

Fermi

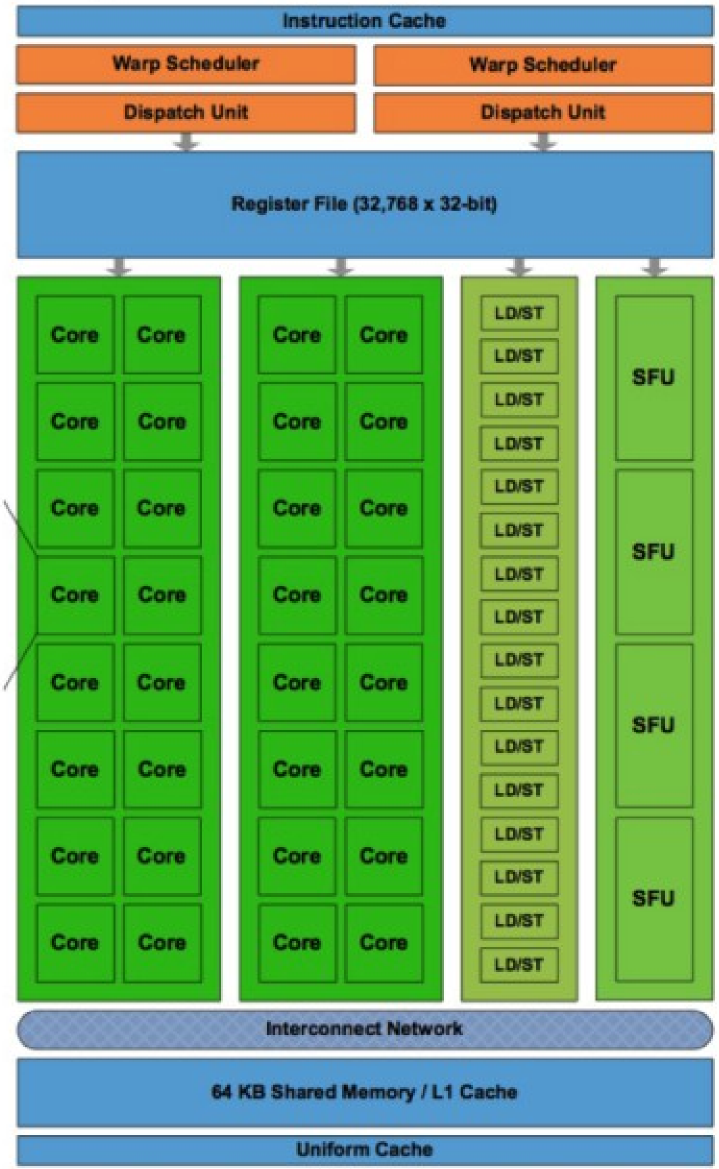
- First generation of Tesla
- Unified Architecture
- MIMD
- VLIW

- Different Storage Unit
  Register File

  L1

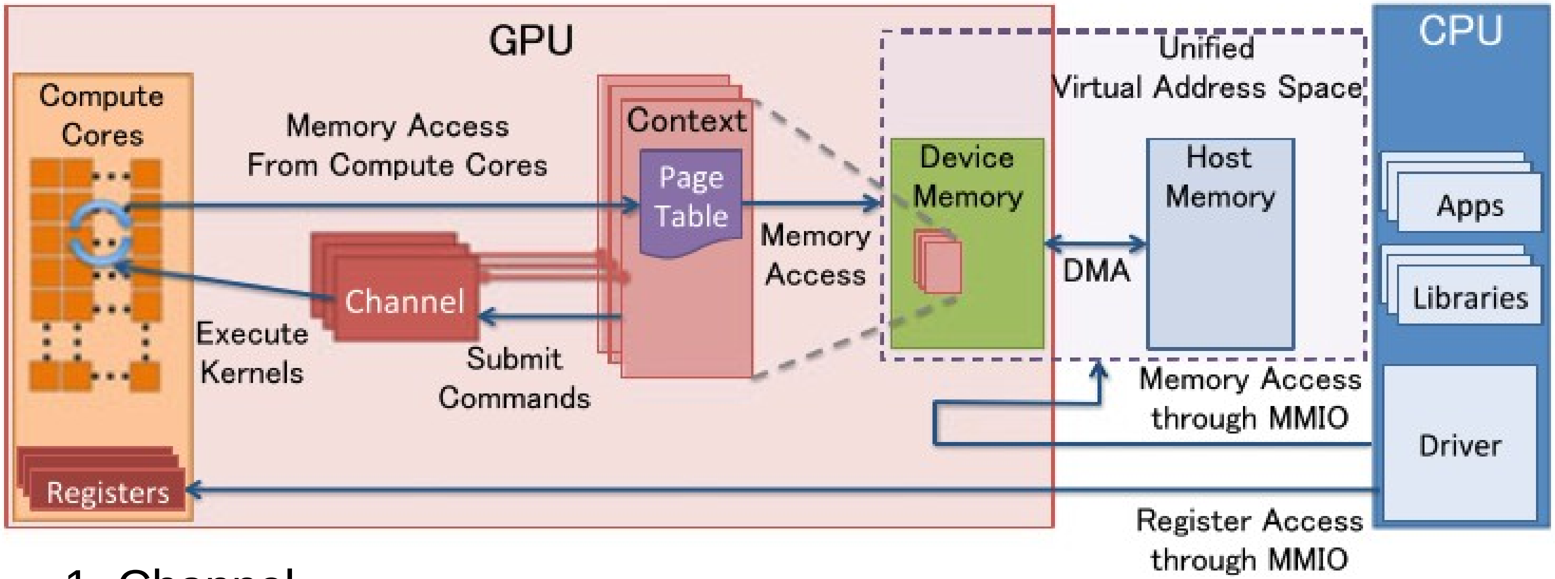
  L2

  GPU Memory VRAM



Fermi Streaming Multiprocessor (SM)

GPU resource management



1. Channel
a command submission system,which is used to launch GPU programs, start DMA operations or synchronize CPU and GPU

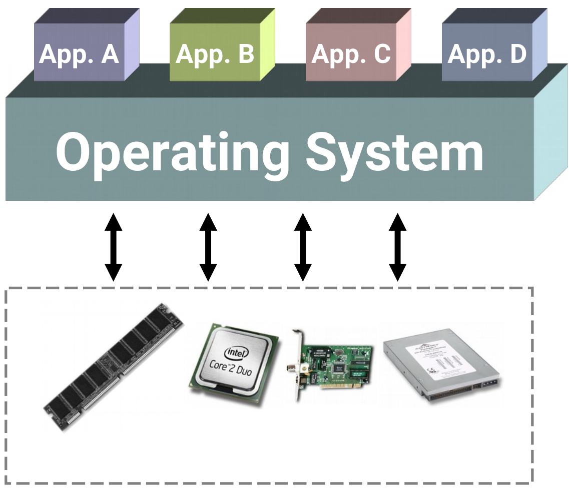
2. Context

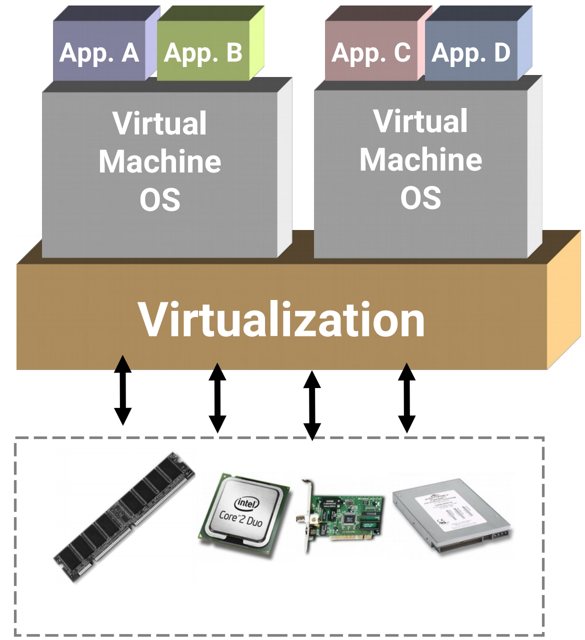
3. Memory
    VRAM: frame buffer
    GTT: ring buffer for channel here

Source: NVIDIA, Inc.

Virtualization

What is virtualization?





Why?

New infrastructure, fundamental of Cloud

Efficient, Security, Manageable, More “Software Define”

Choices:

KVM, XEN, Cirtrix XEN-Server, VMWare Vsphere, Hyper-V

Virtualization

Basic idea:

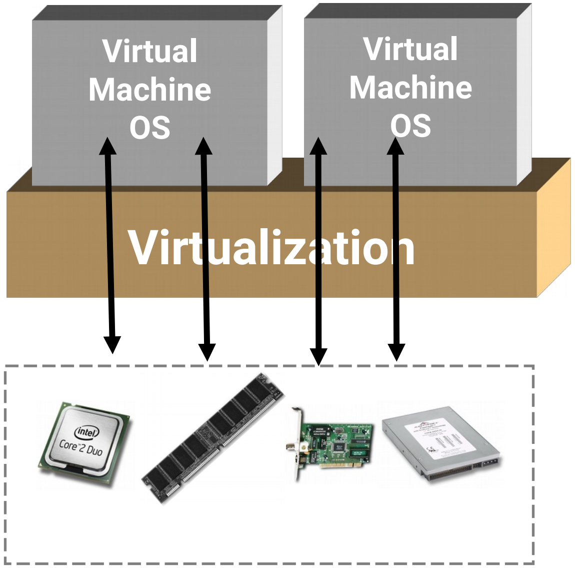
1. Emulation (QEMU, Bachs)

2. Para virtualization (XEN pv, QEMU virtio)

3. Full (Hardware assistant) virtualization
   CPU:
   VT-x Root and None-Root Mode

   Memory:
   EPT/NPT

   IO device
   VT-d/ AMD-Vi /SMMU
   SR-IOV/MR-IOV



GPU Virtualization

GPU Virtualization

Huge market:

Gartner Report:

2017: $ 145B
2018: $ 175B
2019: $ 206B

User Cases:

Auto driver: Tesla
Medical area:
Finance: wall street
Electronic Commerce: Delivery Transport, Recommend Sale
Language Translate:

GPU Virtualization

GPU Virtualization in Cloud, providing machine learning service

Google Colaboratory

Paperspace Gradient

FloydHub Workspace

Lambda GPU Cloud

AWS Deep Learning AMIs

GCP Deep Learning VM Images

GPU Virtualization

GPU virtualization, Software Implementation

- Software Emulated
  CPU “trap and emulate” GPU instruction,Slow, limited function
- API forwarding
  Intercept API, Simple idea, but painful for API compatible

IO virtualization, GPU as a PCIe device today.

- GPU Passthrough
- Full GPU Virtualization

Para?

VirtIO-GPU

GPU Passthrough

GPU as a PCIe device. NVlink?

PCI resources:
PCI configure space, ROM, BARs(PIO, MMIO)

Full API support in Guest VM
Stable, supported by all Vendors with hardware requirement
- From SLES 12SP2
- SOC 8

Native-close performance, 95~97%

Only for One VM and lack of flexibility

Full GPU Virtualization

Run “native” graphics driver in VM, Full API, 3D
Achieve good performance and moderate multiplexing capability

- Split
  - Time Slices
  - framebuffer memory

- Isolate
  - Give a neat access between VM and Host Physical Device
  - IOMMU/Mdev and VFIO
  - DMA
  - Interrupt

- Schedule
  - Efficient and Robust
  - Pretty fix for AMD, hardware implementation
  - More flexible for NVIDIA, RR, BOND

Full GPU Virtualization

Upstream

- NVIDIA (GRID)
- Intel (GVT-G)
- AMD(GIM)

Intel has no VRAM

AMD has IOMMU support

Full GPU Virtualization

Nvidia
Tesla Series: Volta Pascal Maxwell M6 M10 M60 P4 P6 P40 P100 V100
GRID: Kepler K1 K2 (VDI and application virtualization)
http://www.nvidia.com/object/grid-certified-servers.html

AMD
FirePro S7150 S7150x2
Radeon Pro V320 V340
Radeon Instinct MI6 MI8 MI25(Machine learning interface, CUDA compatible with HIP) MI50/60 MI100
https://lists.freedesktop.org/archives/amd-gfx/2016-December/004075.html

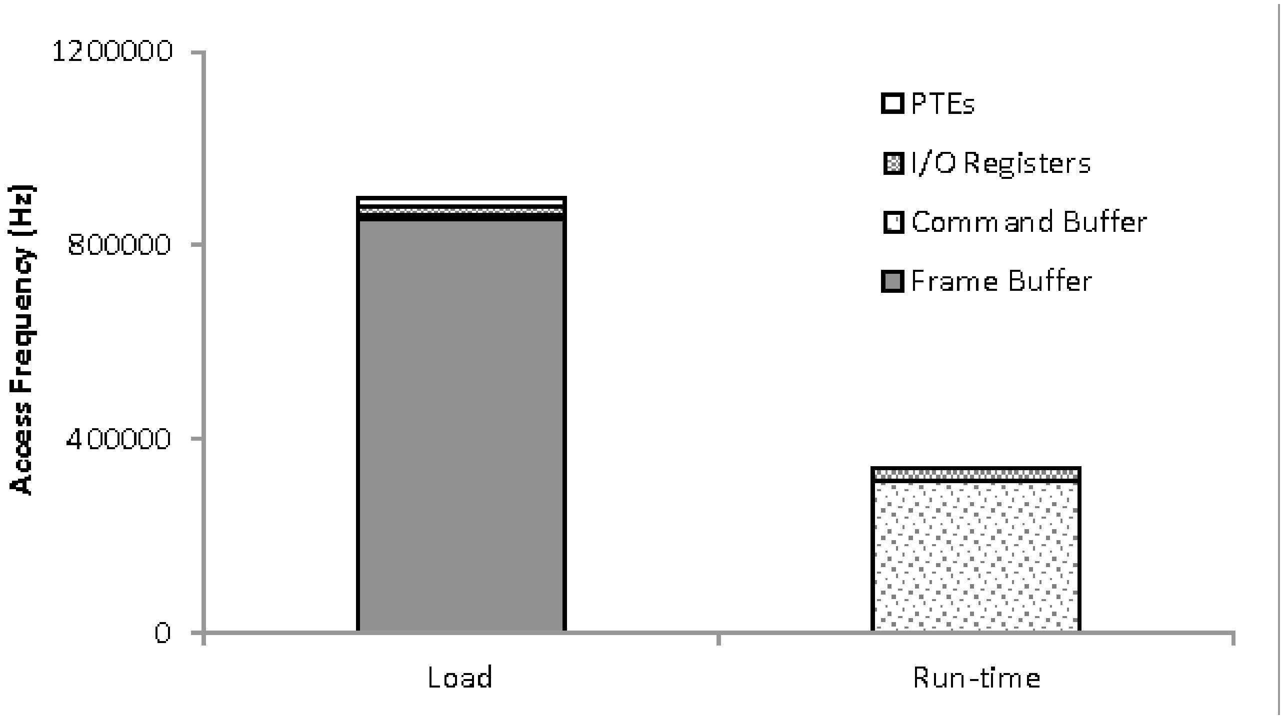
Intel
Haswell(3VMs) Broadwell(7VMs) Skylake, Kaby Lake
2020, dedicated GPU

https://github.com/intel/gvt-linux/wiki

SRIOV vs Mdev

GPU real Workload:

This is why GPU virtualization is doable



VFIO and IOMMU

Key Components:

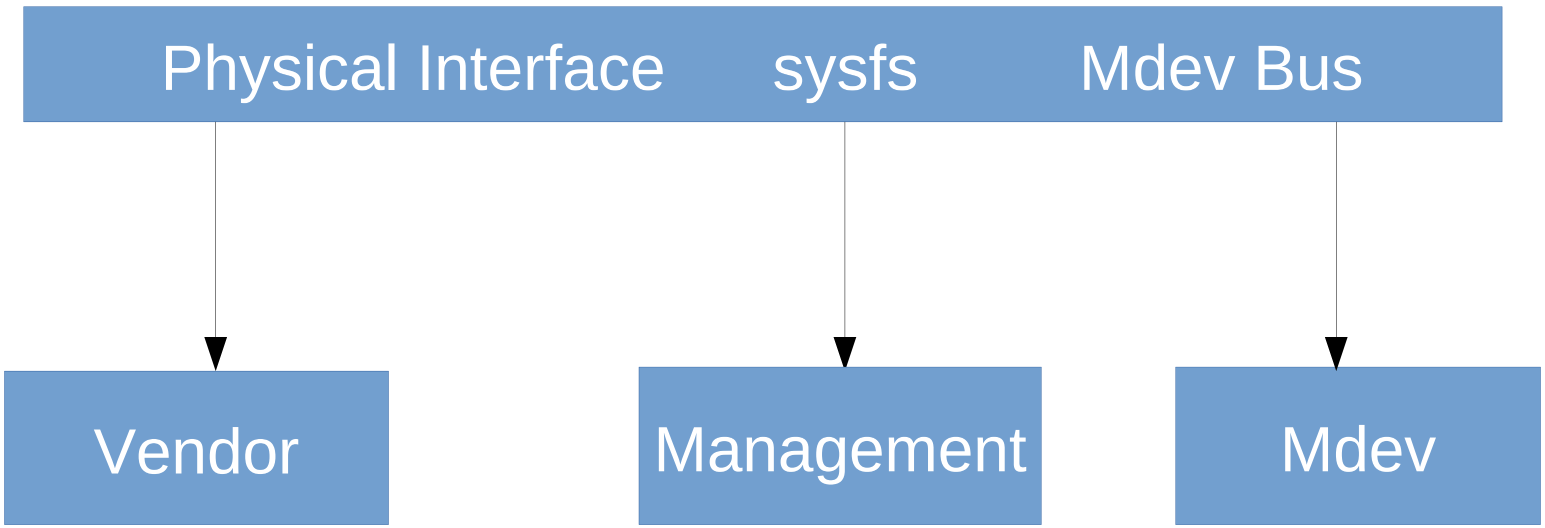
IOMMU: Hardware
- DMA remapping
- Interrupt remapping

VFIO: userspace driver for PCI device

Configure space	QEMU emulated with VFIO
PIO	I/O bitmap of VMCS
MMIO	EPT
Interrupt	IOEVENTFD IRQFD IOMMU
DMA	IOMMU GPA <==> HPA

SR-IOV	Mediated
SRIOV 97%	MDEV 80~90%
supported by standard VFIO PCI (Direct Assignment)	non SR-IOV, require vendor-specific drivers to mediate sharing Leveraging existing VFIO framework,
Established QEMU VFIO/PCI driver, KVM agnostic and well-defined UAPI Virtualized PCI config /MMIO space access, interrupt delivery Modular IOMMU, pin and map memory for DMA	UAPI Vendor driver - Mediated Device – managing device's interna I/O resource

MEDIATED DEVICE FRAMEWORK



i915.ko
nvidia.ko

create
destroy

Vfio-mdev.ko
Vfio API compatible
TYPE1 IOMMU

MEDIATED DEVICE FRAMEWORK

Initialize:

Vendor device register
Vfio Mdev   driver register

Life Cycle:
User writes mdev sysfs to create mdev device

Attach to VM:
QEMU calls VFIO API to add VFIO dev to IOMMU container, group, get fd back

QEMU access device fd and bind it to VM

Demo

Host:

- BIOS enable VT-x & VT-d
- Kernel : >=4.10
- Hardware: limited but more and more coming
- Management driver installment
- Create vGPU device

Hypervisor:

- OVMF firmware, not seabios
- LifeCycle management
- Attach to VM

Guest:

- Guest Kernel Driver
- Development Environment setup

Intel:

i915.enable_gvt=1

```
${GVT_DOM} = 0000\:00
$GVT_PCI = 0000\:00\:02.0
# ls /sys/devices/pci${GVT_DOM}/$GVT_PCI/mdev_supported_types
i915-GVTg_V5_4  i915-GVTg_V5_8
```

UUIDgen

```
# echo "$GVT_GUID" >
"/sys/devices/pci${GVT_DOM}/$GVT_PCI/mdev_supported_types/$GVT_TYPE/create"
```

QEMU command line:

```
 -M graphics=off \
 -display gtk,gl=on \
 -device vfio-pci,sysfsdev=/sys/bus/mdev/devices/$uuid,display=on
```

NVIDIA:

Nothing special needs to do here, make sure register with License.
Install NVIDIA Virtual GPU Manager Driver

```
rpm -iv NVIDIA-vGPU-****.rpm
# lsmod | grep vfio
nvidia_vgpu_vfio      27099  0
nvidia             12316924  1 nvidia_vgpu_vfio
vfio_mdev             12841  0
mdev                  20414  2 vfio_mdev,nvidia_vgpu_vfio
vfio_iommu_type1      22342  0
vfio                  32331  3 vfio_mdev,nvidia_vgpu_vfio,vfio_iommu_type1
```

```
uidgen
cd /sys/class/mdev_bus/domain\:bus\:slot.function/mdev_supported_types/
echo "uuid"> subdirectory/create
```

Libvirt:

```
<hostdev mode='subsystem' type='mdev' model='vfio-pci'>
  <source>
    <address uuid='uuid'/>
  </source>
</hostdev>
```

AMD:

intel_iommu=on
amd_iommu=on

Blacklist amdgpu driver on the host system

Install GIM (GPU-IOV Module)

You will see available GPU VF devices, no different with a normal GPU passthrogh device.

GIM will come with a guest driver too.

Current Status and ToDo

SUSE

- Intel KVMGT technical ready
- Nvidia GRID technical ready
- AMD MxGPU technical ready

- GPU virtualization for Cloud
- GPU virtualization for CAAS

Outside

- Remote display
- IOMMU compatible
- Live Migration
- Scalability: Schedule Algorithm **

GPU for Container

Question?

Thank you.

REFERENCE

VGPU ON KVM

An Introduction to PCI Device Assignment with VFIO - Alex Williamson,

Red Hat [Qemu-devel] [PATCH v7 0/4] Add Mediated device support

[libvirt] [RFC] libvirt vGPU QEMU integratio

https://yq.aliyun.com/articles/590909?spm=a2c4e.11153940.blogcont599189.23.f2016d7bXPo7TD

https://zhuanlan.zhihu.com/p/35489035

We adapt. You succeed.

Corporate Headquarters
Maxfeldstrasse 5
90409 Nuremberg
Germany

+49 911 740 53 0 (Worldwide)
www.suse.com

Join us on:
www.opensuse.org

257-000020-001

We adapt. You succeed.