GPU Virtualization:

What we should know today

Liang Yan (lyan)
SUSE Labs

We adapt. You succeed.

Outline

1. Background
   GPU
   Virtualization
2. GPU virtualization
   Definition and Classification
   Use scenario
3. Critical techniques
   SRIOV vs MDEV
4. Current status and Future
   SUSE
   Upstream
5. 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)

GPU Purpose

Graphic Render

3D hardware acceleration

- DirectX
- OpenCL
- Vulkan

General Compute

Big Data, Machine Learning: Tensor-flow, Caffe2

CUDA	Compute Unified Device Architecture
OpenCL	Open Computing Language

GPU Structure

Fermi

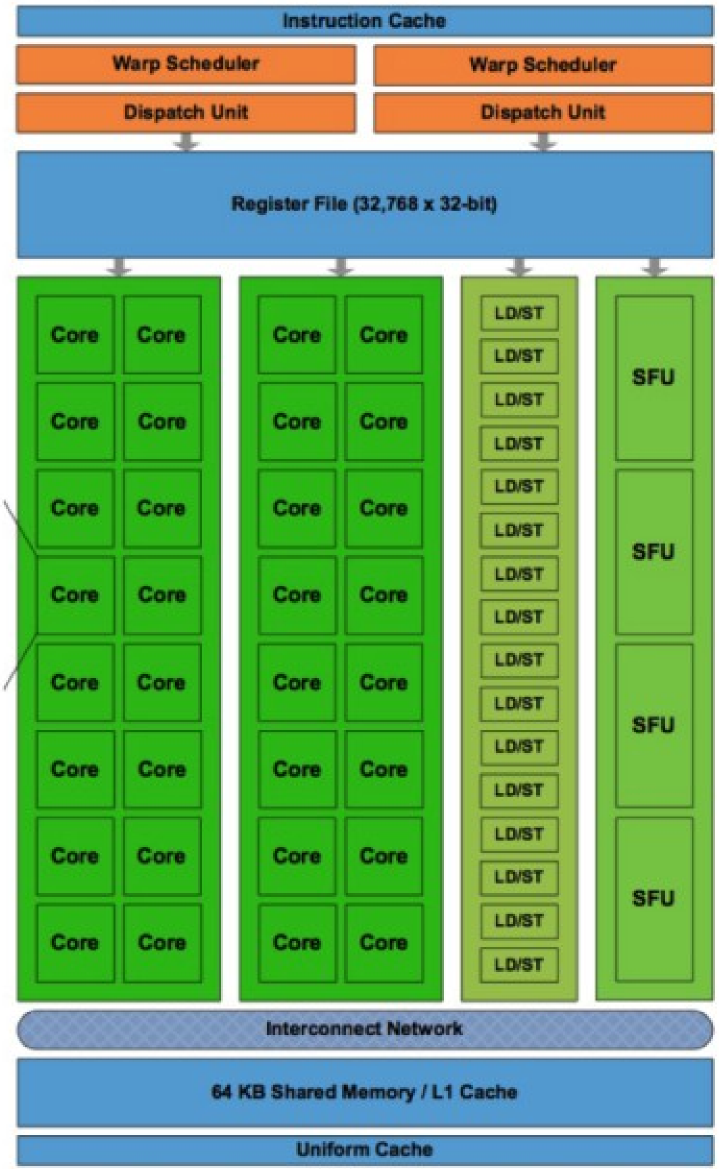
- First generation of Tesla
- Unified Architecture
- MIMD ==> but better performance with SIMD
- VLIW
- Different Storage Unit

  Register File

  L1

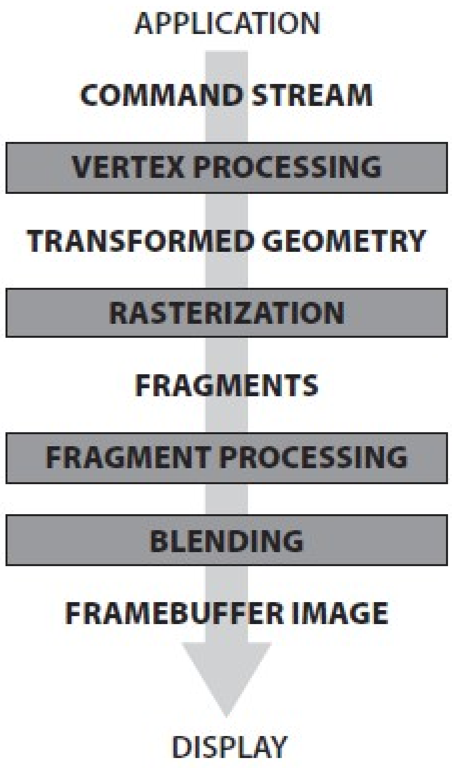
  L2

  GPU Memory VRAM



Fermi Streaming Multiprocessor (SM)

GPU Pipeline Structure



Traditional Pipeline Structure

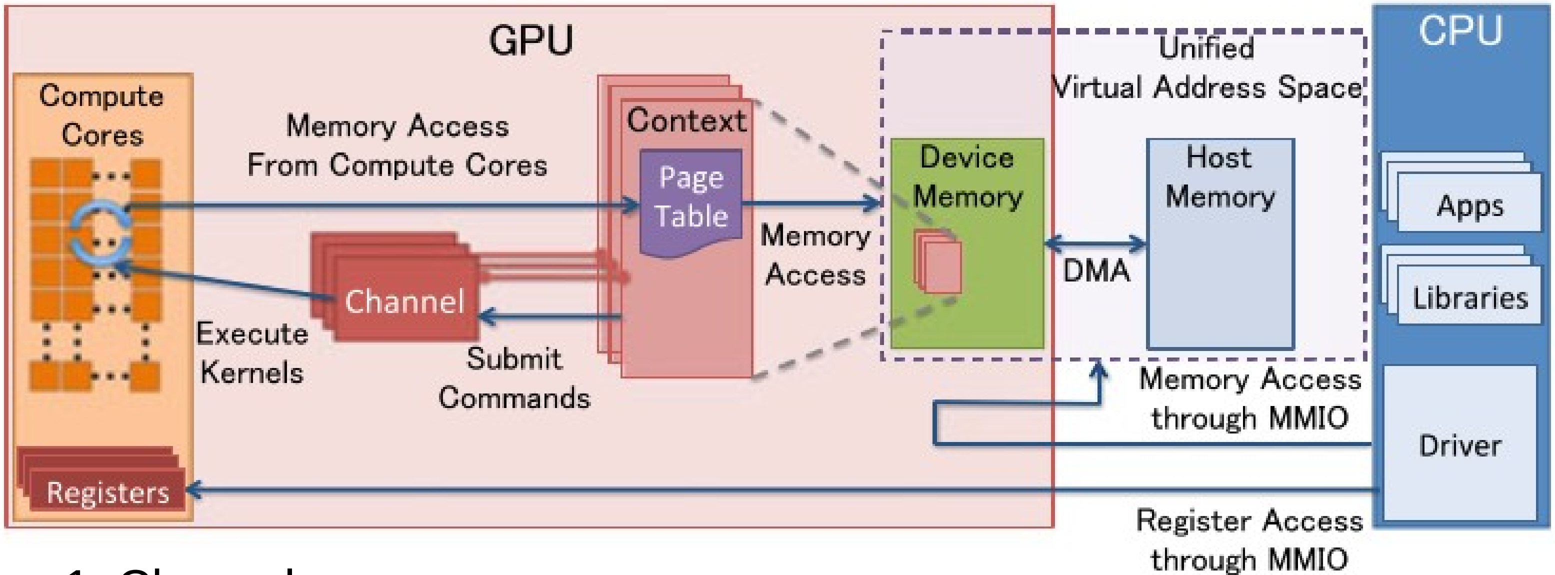
Vertex Shader
Fragment/Pixel Shader
Other Shader

=====================

New Unified Architect:

Unified shader

GPU resource management



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

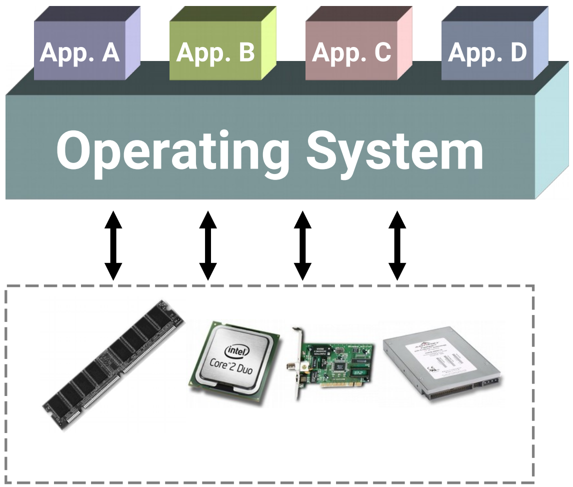
2. Context

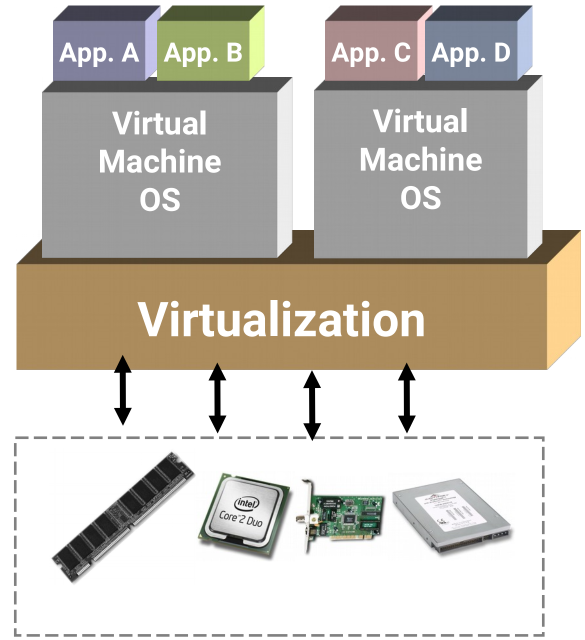
3. Memory
VRAM: frame buffer
GTT: ring buffer for channel here

Source: NVIDIA, Inc.

Virtualization

What is virtualization?





Why?
New infrastructure, fundamental of Cloud
Efficient, Security,

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

Virtualization

Basic idea:
Try to make VM access Physical Resource directly, decrease the overload by Virtualization.

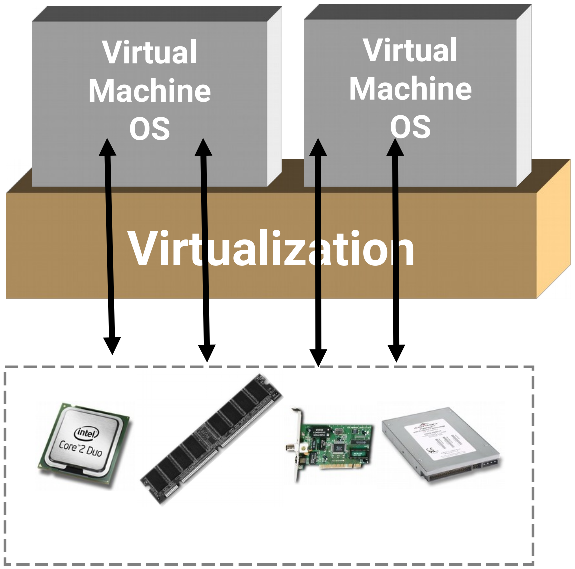
Different Stage:
Emulation (QEMU, Bachs)

Para virtualization (XEN pv, QEMU virtio)

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: 145 billions
2018: 175 billions
2019: 206 billions

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

Virtual GPU for Guest VM
Like a real GPU as much as possible

GPU virtualization, different stage like Virtualization
Software Virtualization

- Software Emulated
  CPU “trap and emulate” GPU instruction
  Slow, limited function

- API forwarding
  Frontend intercept APIs and forward
  Backend translate and send back
  Simple idea, but painful for API compatible

IO virtualization, GPU as a PCIe device today.

- GPU Passthrough
- Full GPU Virtualization

GPU Passthrough

GPU as a PCIe device today.

Full API support in Guest VM

Stable, supported by all Vendors with hardware requirement

- From SLES 12SP2
- SOC 8

Native-close performance, 95~97%

PCI resources:

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

Key Components:

- IOMMU: Hardware
- VFIO: DMA operation in userspace level

VFIO and IOMMU

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

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

Full GPU Virtualization

Run native graphics driver in VM
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,
  - More flexible for NVIDIA, RR, BOND

Full GPU Virtualization

vGPU Investments Upstream

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

Intel has no VRAM

AMD has IOMMU support

SRIOV 97%
MDEV 80~90%

Full GPU Virtualization

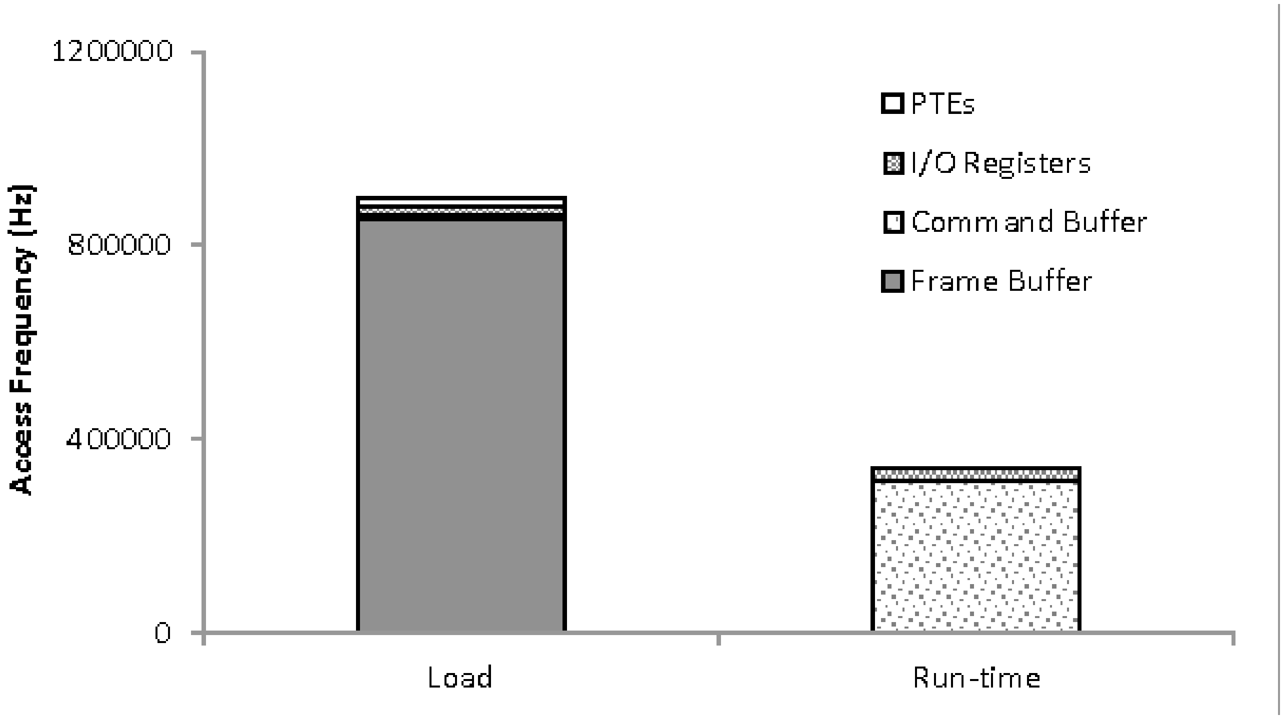
Nvidia
Tesla Series: Volta Pascal Maxwell M6 M10 M60 P4 P6 P40 P100 V100
GRID: Kepler K1 K2 (VDI and application virtualization)
http://www.nvidia.com/object/grid-certified-servers.html

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

Intel

Haswell(3VMs) Broadwell(7VMs) Skylake, Kaby Lake
https://github.com/intel/gvt-linux/wiki

SRIOV vs Mdev


1200000
800000
400000
0
Access Frequency (Hz)
PTEs
I/O Registers
Command Buffer
Frame Buffer
Load
Run-time

SR-IOV devices

supported by standard VFIO PCI (Direct Assignment)

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

Mediated devices

non SR-IOV, require vendor-specific drivers to mediate sharing Leveraging existing VFIO framework, UAPI Vendor driver - Mediated Device – managing device’s internal I/O resource

MEDIATED DEVICE FRAMEWORK

Mediated core module (new) Mediated bus driver, create mediated device Physical device interface for vendor driver callbacks Generic mediate device management user interface (sysfs)

Mediated device module (new) Manage created mediated device, fully compatible with VFIO user API

VFIO IOMMU driver (enhancement) VFIO IOMMU API TYPE1 compatible, easy to extend to non-TYPE1

Registers VFIO MDEV as driver
Vendor driver registers devices
Vendor driver registers Mediated CBs
User writes mdev sysfs to create mdev device
QEMU calls VFIO API to add VFIO dev to IOMMU container, group, get fd back
QEMU access device fd and present it into VM

A mediated pass-through solution for graphics virtualization
- Pass-through performance critical resources
- Trap-and-emulate privileged operations

Maintain a device model per VM

Current Status and Future

Most of the work is done by Intel Nvidia IBM and RH unfortunately

SUSE

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

- GPU passthrough stage for Cloud
- GPU virtualization for CAAS

Upstream

- Remote display
- IOMMU compatible
- Live Migration
- Scalability

1. GPU virtualization in CAAS
Nvidia-docker 2.0
Kubernetes
Kata-Container

2. People would be more interested in PAAS or EVEN SAAS
Provide a machine learning environment

cuDNN(Deep Neural Network)

A lot of startups are working on this: unicorn company

Question?

Thank you.

We adapt. You succeed.

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

SUSE®

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

Unpublished Work of SUSE. All Rights Reserved.

This work is an unpublished work and contains confidential, proprietary, and trade secret information of SUSE. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

General Disclaimer

This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.

257-000020-001

We adapt. You succeed.