BRING AN INTEL® SCALABLE IOV CAPABLE DEVICE INTO LINUX*

Xin Zeng <xin.zeng@intel.com>

Yi Liu <yi.l.liu@intel.com>

1st, Nov. 2019

Disclaimers

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request. No product or component can be absolutely secure.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

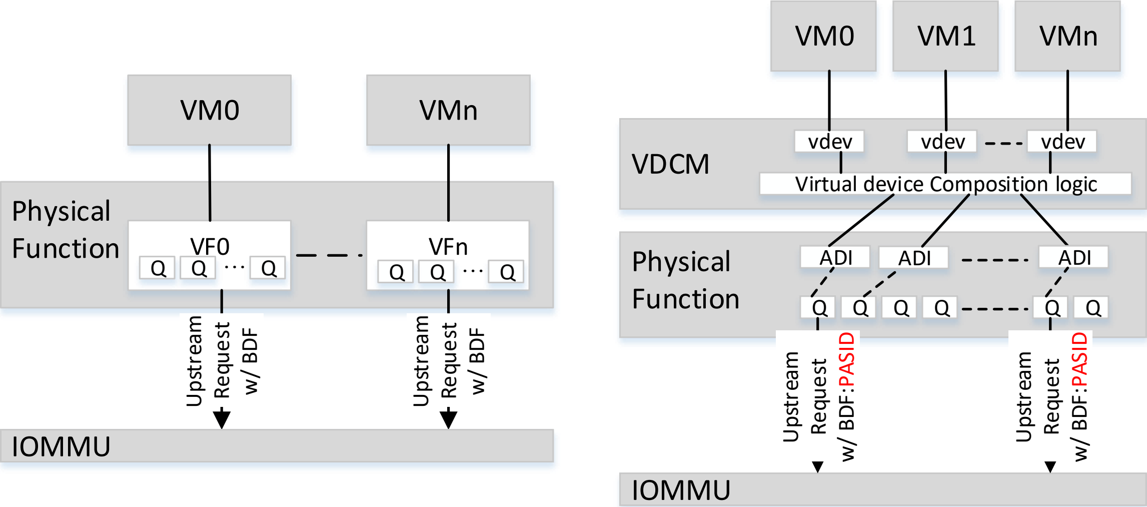
*Other names and brands may be claimed as the property of others.

Agenda

- Recap Intel® Scalable IOV technology
- The software stack in Linux*
- Develop the device driver in Linux*
- Intel® Scalable IOV & Virtual Shared Virtual Address (vSVA)
- Summary & Opens

Recap Intel® Scalable IOV technology

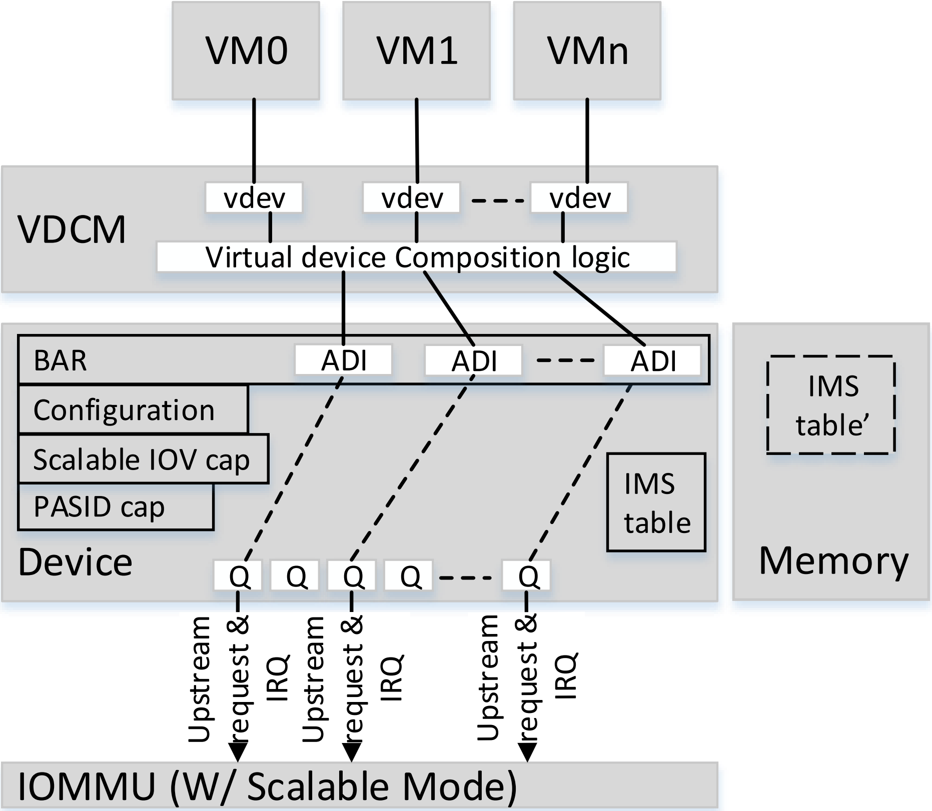
- ## Hardware
  - Spec is public by Intel in 2018
  - PASID granular DMA isolation
  - Finer assignable hardware resource
- ## Software
  - Composes assignable Virtual Device (VDEV)
  - Mediates access to hardware
- ## Combined together
  - Hardware enforced DMA isolation while keeping scalability and flexibility



SR-IOV VS. Scalable IOV

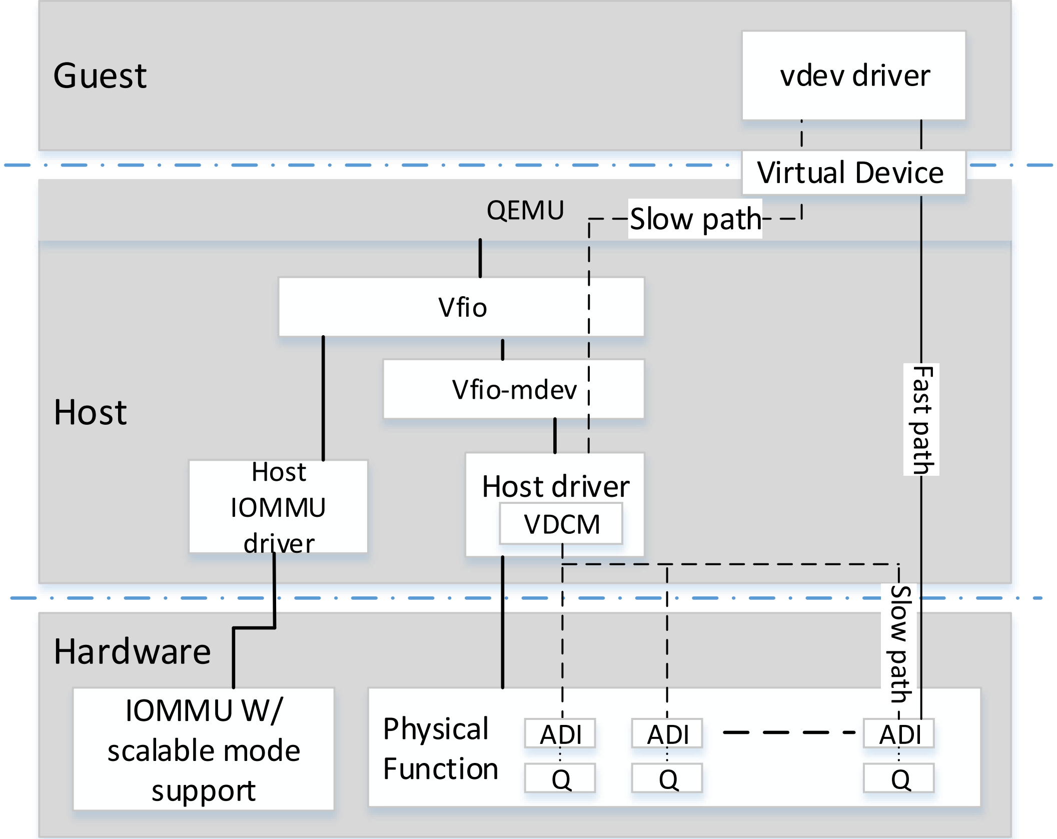
What’s new in hardware?

- **Device**
  - Assignable device interface (ADI)
    - Minimal assignable hardware unit
    - MMIO only
    - Isolated resource
  - Interrupt message storage (IMS) for ADI
  - Intel® Scalable IOV capability in DVSEC
  - PASID capability is required
- **Intel® VT-d (IOMMU)**
  - Scalable mode support



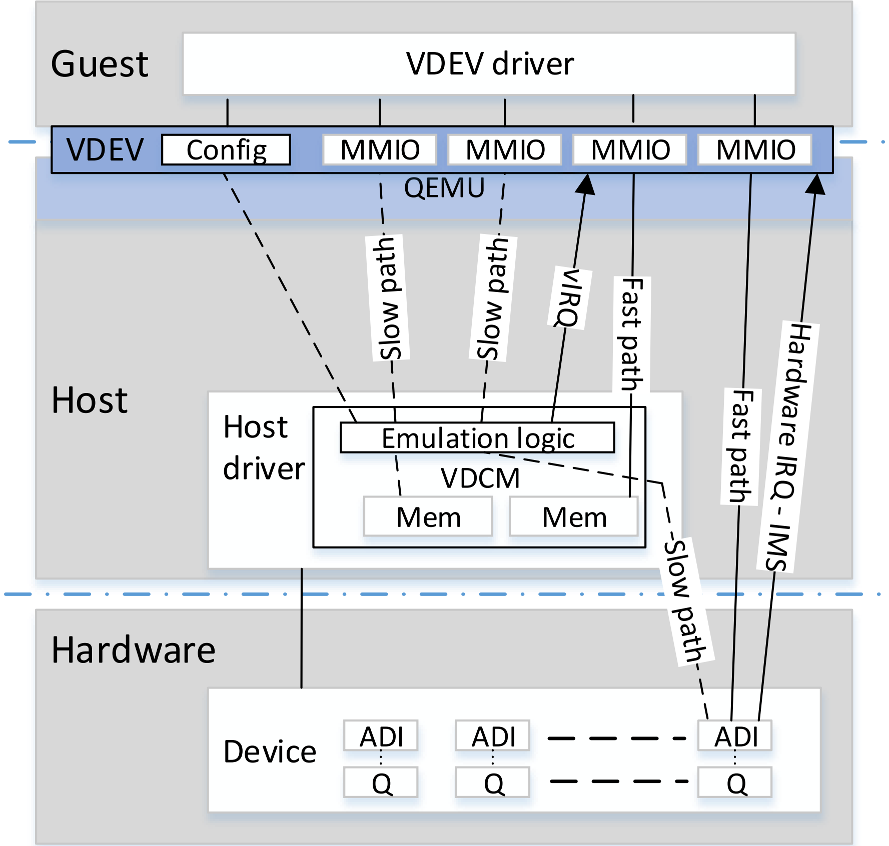
SW architecture in Linux*

- **Guest**
  - Virtual device (VDEV) driver
- **QEMU**
  - Be agnostic to Intel® Scalable IOV VDEV pass-through
- **Host**
  - Mediated Device Framework
  - Device specific Virtual Device Composition Module (VDCM)
  - IOMMU driver



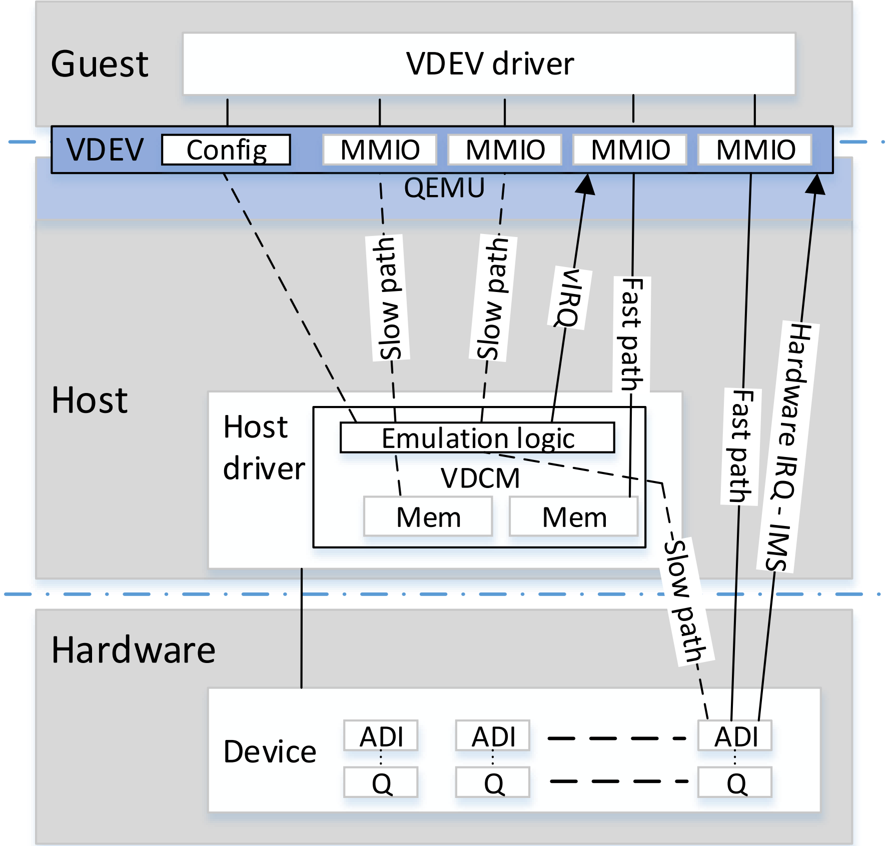
Design the VDCM

- **Determine the VDEV types**
  - The services VDEV provides
- **Organize the VDEV resources**
  - The virtual config space
  - The virtual bar regions
    - Fast path: e.g. work submission interface
    - Slow path: e.g. config, control
  - The interrupts
    - ADI Interrupt from IMS
    - Virtual interrupt



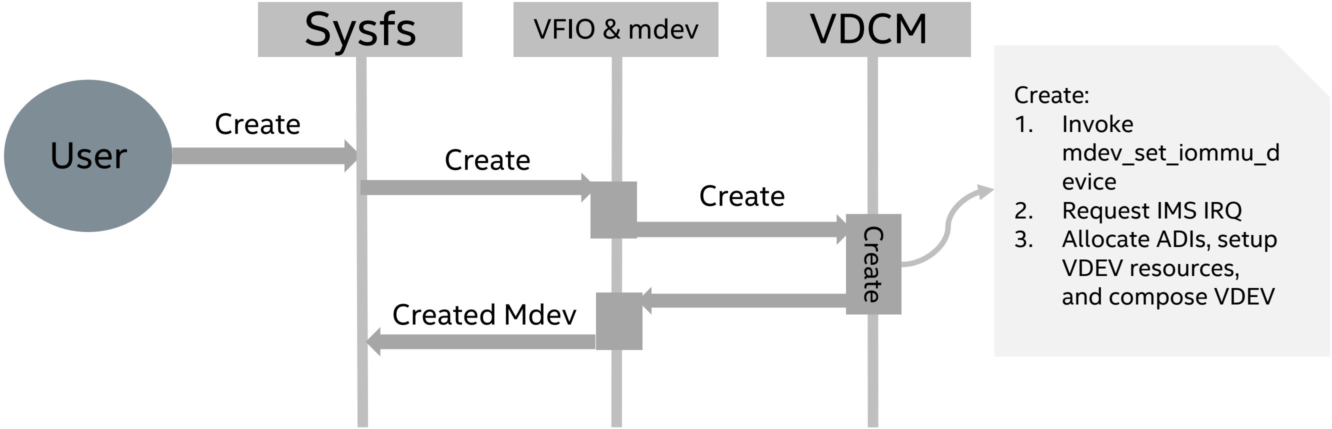
Design the VDCM (Cont'd)

- **Design the VDEV-VDCM communication channel**
  - Software based mechanism
    - Memory backed virtual MMIO
  - Hardware based mechanism
    - Mailbox MMIO
- **Compose and manage VDEVs**



VDEV lifecycle management with VDCM

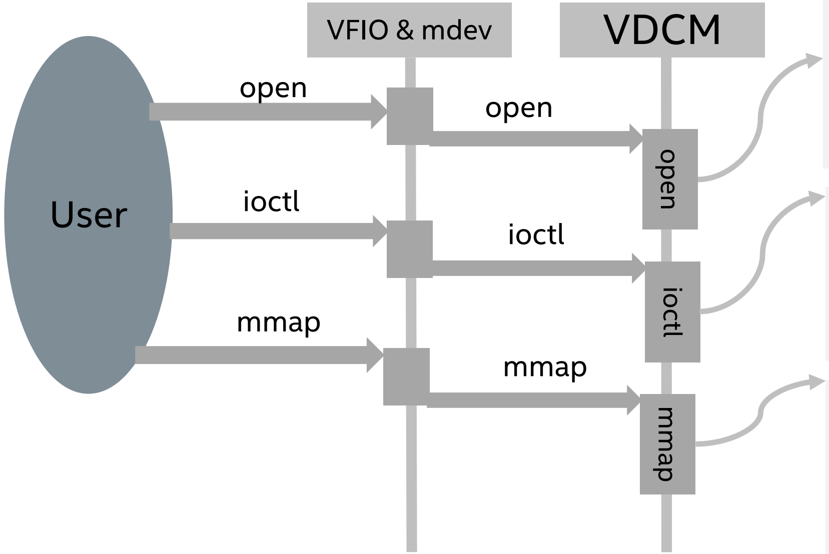
- **Create**
  - <Vdev type/uuid>
- **Assign**
- **Run-time access**
- **Reset**
- **Release**
- **Remove**



Create:
1. Invoke mdev_set_iommu_device
2. Request IMS IRQ
3. Allocate ADIs, setup VDEV resources, and compose VDEV

VDEV lifecycle management with VDCM

- Create
  - <Vdev type/uuid>
- **Assign**
- Run-time access
- Reset
- Release
- Remove



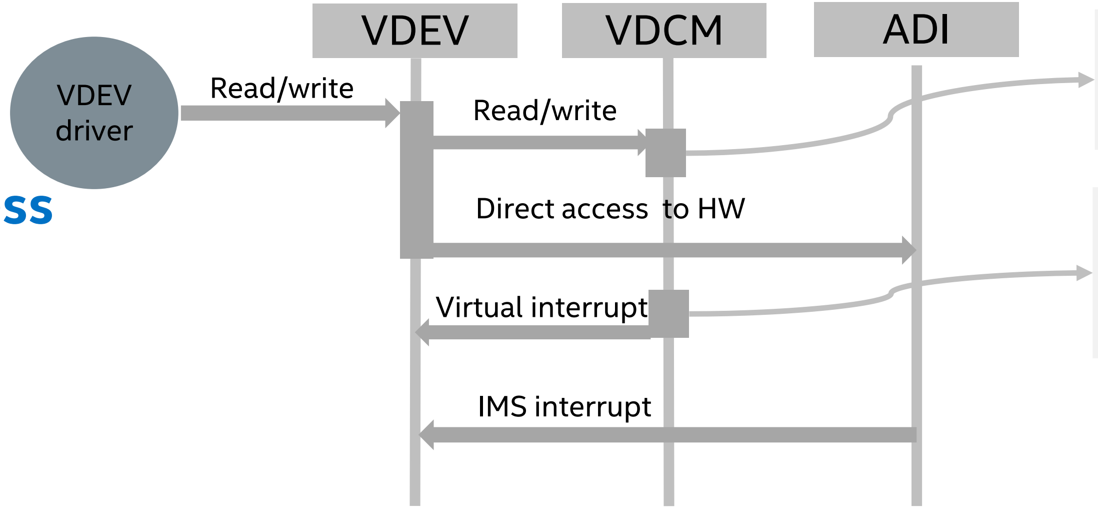
open:
1. Query default pasid from AUX domain
2. Program pasid to ADI in hardware

Ioctl:
1. Respond VDEV resource
2. Program IMS vector
3. Handle VDEV reset

mmap:
1. Map designated fast path resource to user

VDEV lifecycle management with VDCM

- Create
  - <Vdev type/uuid>
- Assign
- **Run-time access**
- Reset
- Release
- Remove



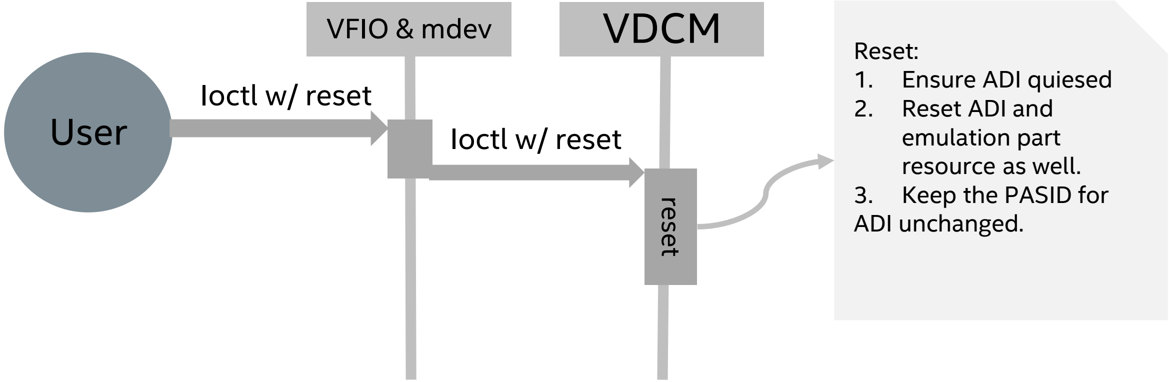
read/write:

1. Emulate the VDEV slow path access

VDCM emulates the virtual interrupt and inject it to the VM.

VDEV lifecycle management with VDCM

- Create
  - <Vdev type/uuid>
- Assign
- Run-time access
- **Reset**
- Release
- Remove

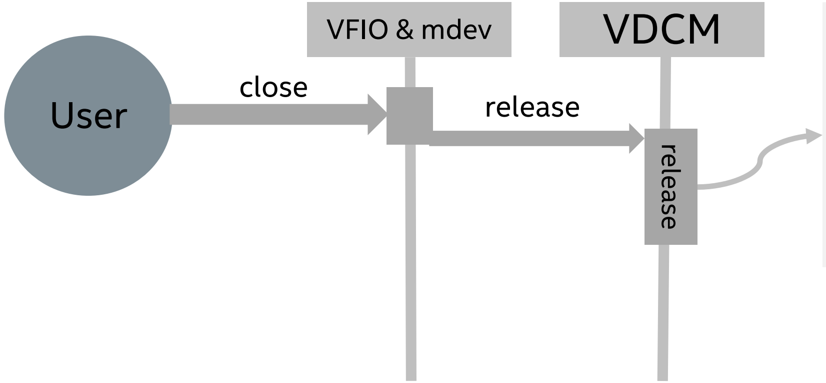


Reset:

1. Ensure ADI quiesed
2. Reset ADI and emulation part resource as well.
3. Keep the PASID for ADI unchanged.

VDEV lifecycle management with VDCM

- Create
  - <Vdev type/uuid>
- Assign
- Run-time access
- Reset
- **Release**
- Remove

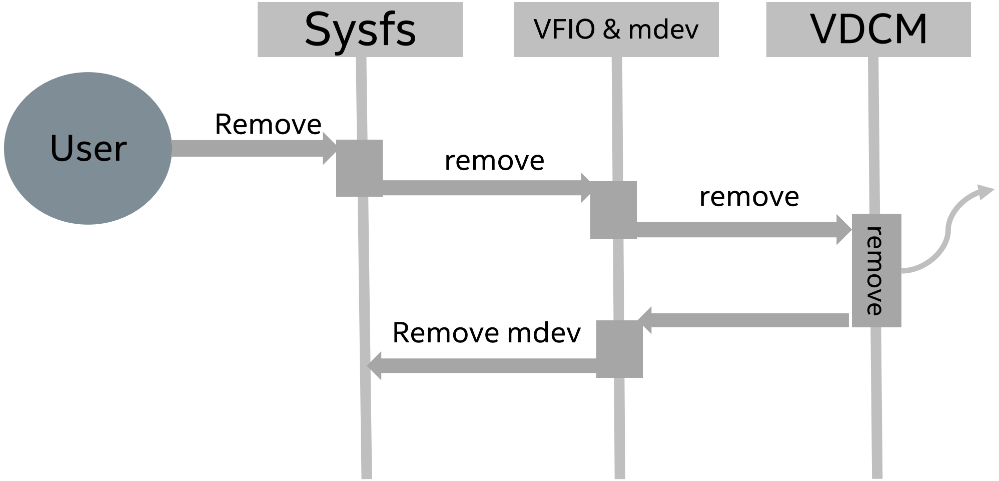


release:

1. Invalidate pasid in ADI
2. Reset VDEV to ensure no further activities in hardware
3. Clean up VDEV resource allocated after opening.

VDEV lifecycle management with VDCM

- Create
  - <Vdev type/uuid>
- Assign
- Run-time access
- Reset
- Release
- **Remove**

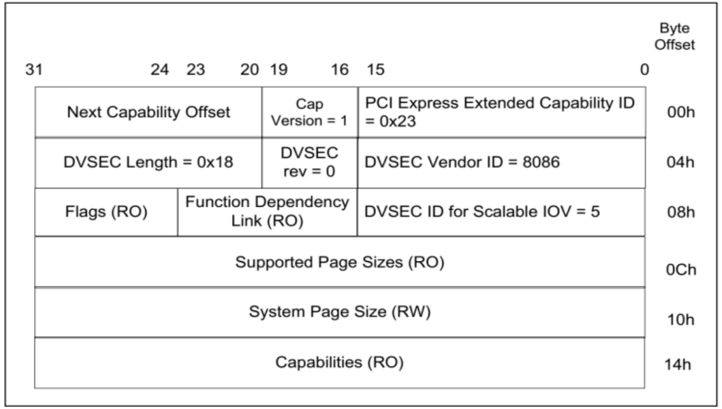


remove:

1. Remove iommu device from VDEV
2. Release IMS IRQ
3. Release ADIs and the associated resources
4. Any state need to be cleaned up

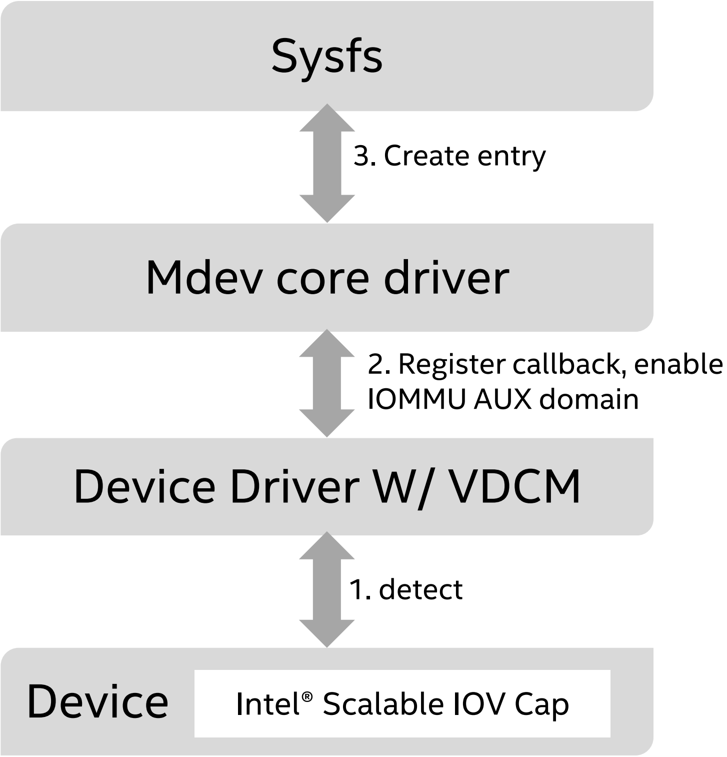
What’s missing? - Discover Intel® scalable IOV capability!

- **Software**
  - Detects Intel® scalable IOV capability
  - Registers parent operations to mdev core driver
- **Hardware**
  - Presents Intel® capability in DVSEC



31 … 24	23 … 20	19 … 16	15 … 0	Byte Offset
Next Capability Offset		Cap Version = 1	PCI Express Extended Capability ID = 0x23	00h
DVSEC Length = 0x18		DVSEC rev = 0	DVSEC Vendor ID = 8086	04h
Flags (RO)	Function Dependency Link (RO)		DVSEC ID for Scalable IOV = 5	08h
Supported Page Sizes (RO)				0Ch
System Page Size (RW)				10h
Capabilities (RO)				14h

Intel® Scalable IOV capability



Sysfs

3. Create entry

Mdev core driver

2. Register callback, enable IOMMU AUX domain

Device Driver W/ VDCM

1. detect

Device

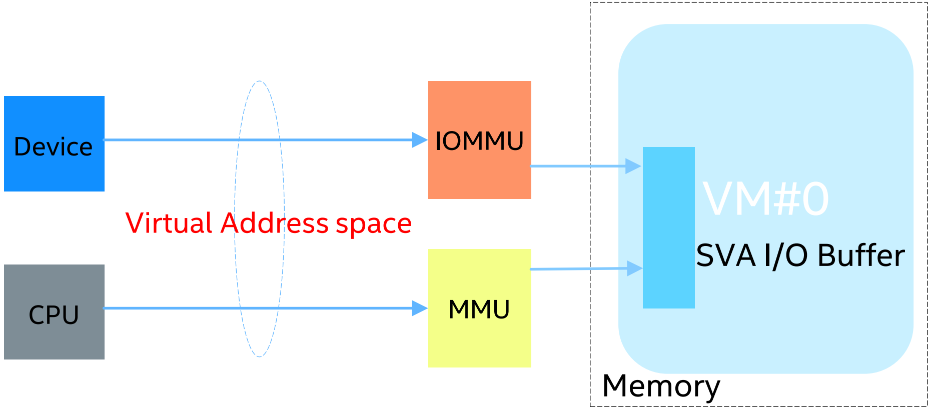
Intel® Scalable IOV Cap

➢ Intel® Scalable IOV & vSVA

- ➢ Recap vSVA
- ➢ VDCM to support vSVA

Recap vSVA

- Shared Virtual Addressing (SVA) is a hardware feature that allows address space sharing between CPU and I/O device for memory access.



- SR-IOV: the generic software(VFIO/IOMMU/QEMU) changes is in community
- Scalable IOV reuses the generic software arch with SR-IOV

SVA in KVM: https://www.youtube.com/watch?v=Kq_nfGK5MwQ

vSVA in Scalable IOV and SR-IOV

- **Different DMA transaction types in PCI Express* hardware**
  - SVA transaction targets to a MMU managed address space
  - Non-SVA transaction targets to an IOMMU managed address space

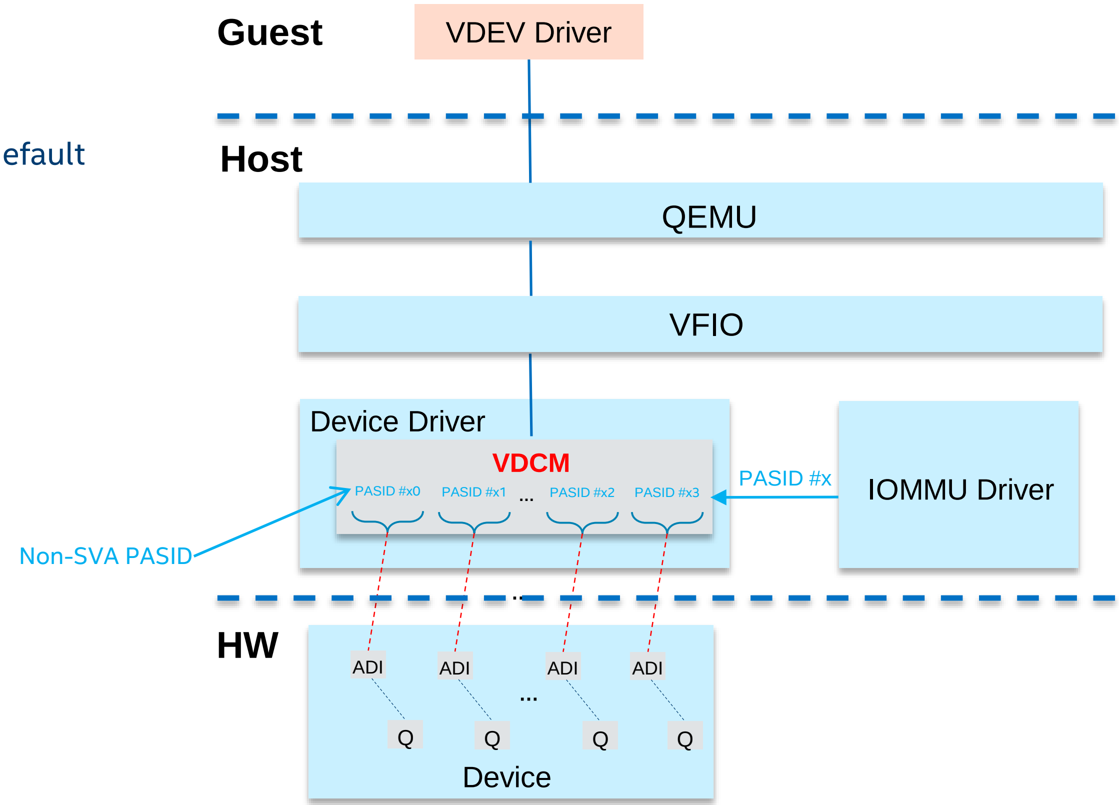
memory requests differences in PCIe*

	SR-IOV	Scalable IOV
SVA transaction	w/ PASID	w/ PASID
Non-SVA transaction	w/o PASID	w/ PASID

VDCM should deal with the PASID differences

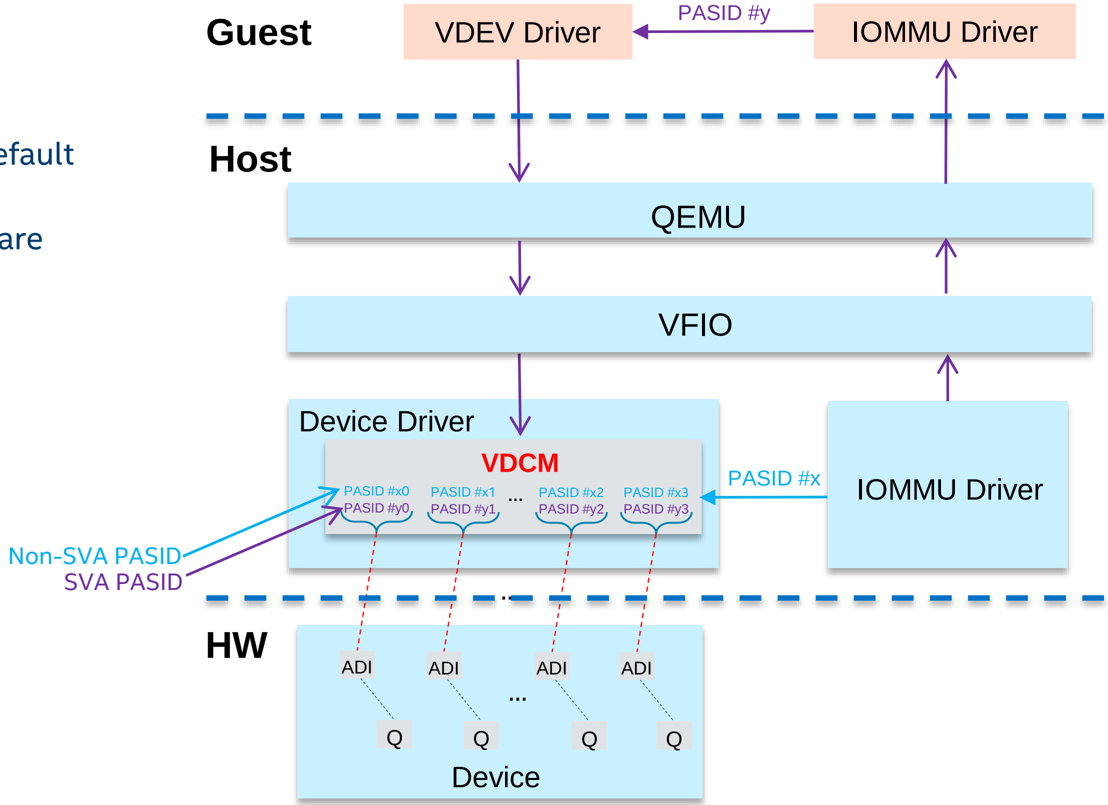
VDCM to support vSVA

- **PASID Management**
  - **Track the PASIDs**
    - Non-SVA PASID: AUX domain default PASID



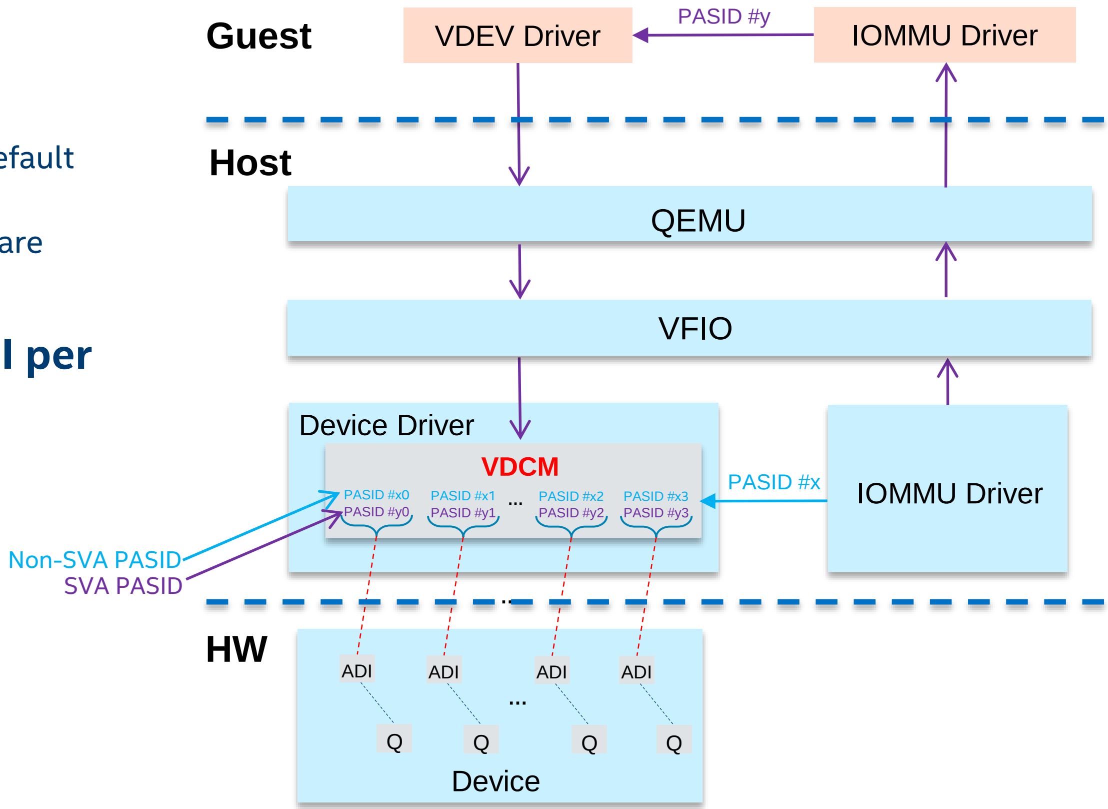
VDCM to support vSVA

- **PASID Management**
  - **Track the PASIDs**
    - Non-SVA PASID: AUX domain default PASID
    - SVA PASID: set from guest software



VDCM to support vSVA

- **PASID Management**
  - **Track the PASIDs**
    - Non-SVA PASID: AUX domain default PASID
    - SVA PASID: set from guest software
  - **Switch the PASID for the ADI per guest operation**
    - Enable vSVA or disable vSVA
  - **Do PASID drain/reset**
    - SVA PASID free



VDCM to support vSVA (Cont.)

- **Compose VDEV with SVA related Capabilities**
  - PCI Express* PASID/ATS/PRS capability structures in VDEV configuration space
  - Native software should have enabled the above capabilities

- **Handle IOMMU page fault**
  - Register fault handler to iommu driver
  - Notify user space client

Summary

- Intel® Scalable IOV enforces DMA isolation at PASID granularity
- Intel® Scalable IOV brings more scalability and flexibility
- To develop the VDCM for the PASID granular DMA isolation capable technology like Intel® Scalable IOV
  - Determine your virtual device types
  - Organize your virtual device resource into slow path and fast path
  - Take care the hardware interrupt (IMS in Intel® Scalable IOV) and PASID programming
  - Leverage existing mediated device framework for VDEV management
  - Emulate SVA capabilities stuff and handle IOMMU page fault to support vSVA

QUESTIONS?

intel®

experience
what's inside™