Intel Arch SIG Meeting

Scalable IOV Introduction

Jane Lv & Jun Tian (SATG/SSE Linux OS Enabling team)

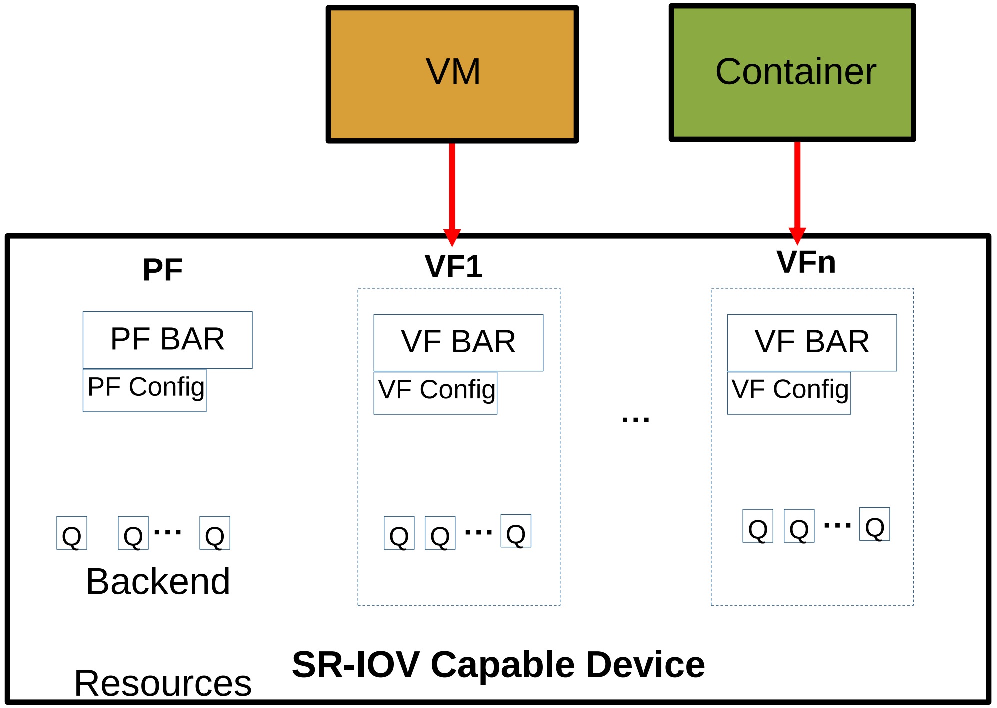
intel®

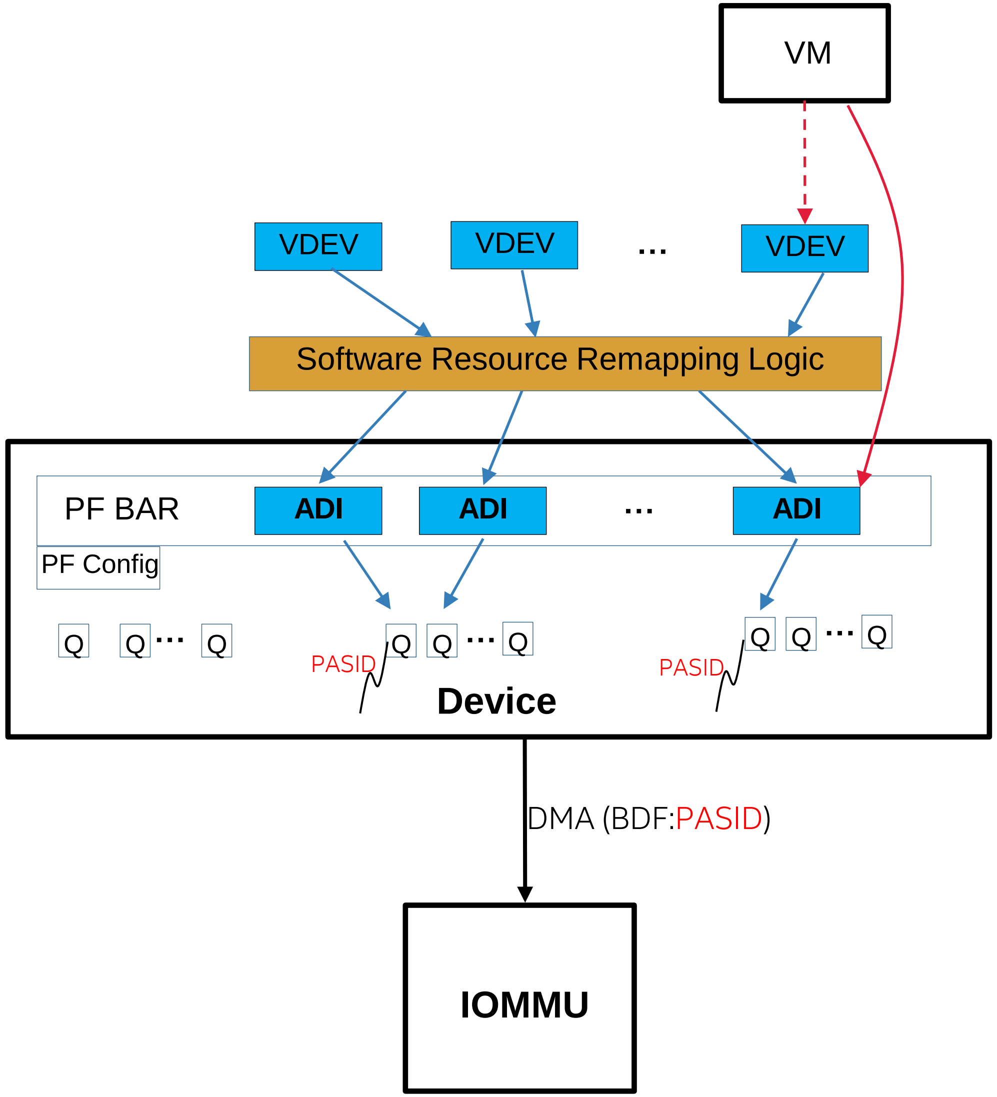
Agenda

- SIOV Introduction
- Upstream Refactoring

SIOV Introduction

SIOV vs SR-IOV





PCI Single Root I/O Virtualization (SR-IOV)
- Physical Function & Virtual Function
- VF directly assignable to VM/container

Scalable IOV
- Assignable Device Interface (ADI), Queues, Q pairs, contexts with PASID
- PASID-granule DMA isolation
- Compose ADIs into Virtual Device (Vdev)

SIOV Advantages

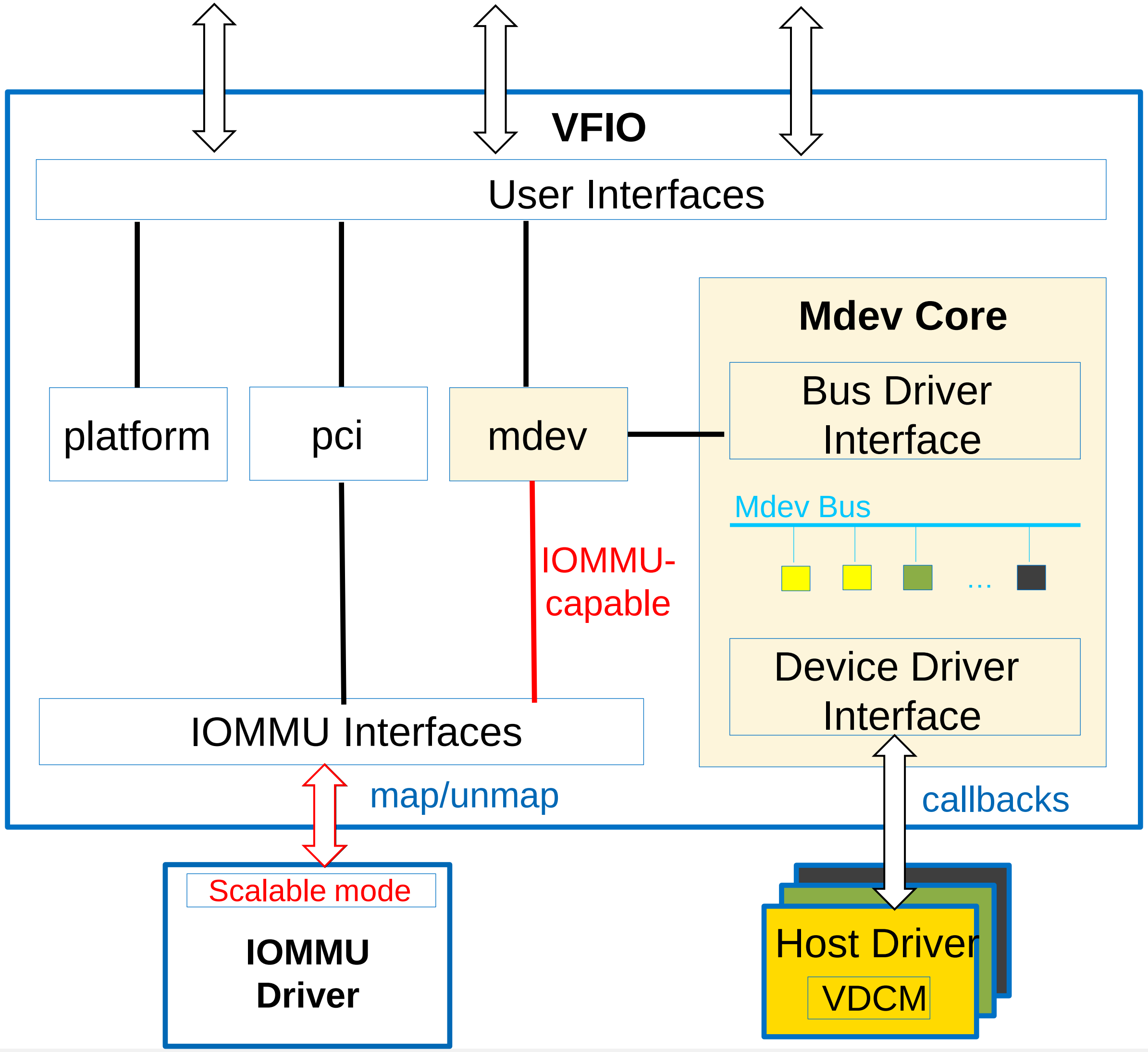
Usage Point-of-view

- Hyper-scale usages
  - E.g. scale to 1000+ VMs/containers
- Dynamic resource management
  - E.g. user-defined sharing granularity, over-provisioning, etc.
- Composability
  - E.g. VM live migration, snapshot, etc.

Technical Point-of-view

- A hardware-assisted mediated pass-through architecture
  - Slow-path operations emulated by SW
  - Fast-path resources dynamically provisioned for direct access
  - Hardware-enforced DMA isolation between fast-path resources
- Finer-grained device sharing than SR-IOV
  - Think about each TX/RX queue pair is now assignable
- Supports any type of devices
  - Integrated or discrete
- Utilizes existing PCIe capabilities
  - e.g. Process Address Space ID (PASID)

SIOV Framework (OOT)



- IOMMU-capable mdev
  - ✓ Link to iommu_domain (tagged by PASID)
  - ✓ Allow PASID-granule iommu map/unmap
- Opt-in by VDCM
  - ✓ When a mdev is created

SIOV Concepts

- Work Queues (WQ) - On device storage to queue descriptors to the device. Requests are added to a WQ by using new instructions to write to the memory mapped “portal” associated with each WQ.
  - Dedicated WQ (DWQ) - A single client owns this exclusively and can submit work to it.
    - A client using DWQ submits work descriptors using the **MOVDIR64B** instruction. Posted write, do not exceed the configured length of WQ
  - Shared WQ (SWQ) - Multiple clients can submit work to the SWQ.
    - Clients using shared work queues submit work descriptors using either ENQCMDS (from supervisor mode) or **ENQCMD** (from user mode).
    - These instructions indicate via the EFLAGS.ZF bit whether the request was accepted.
    - A SWQ requires PASID and can only run with SVA support

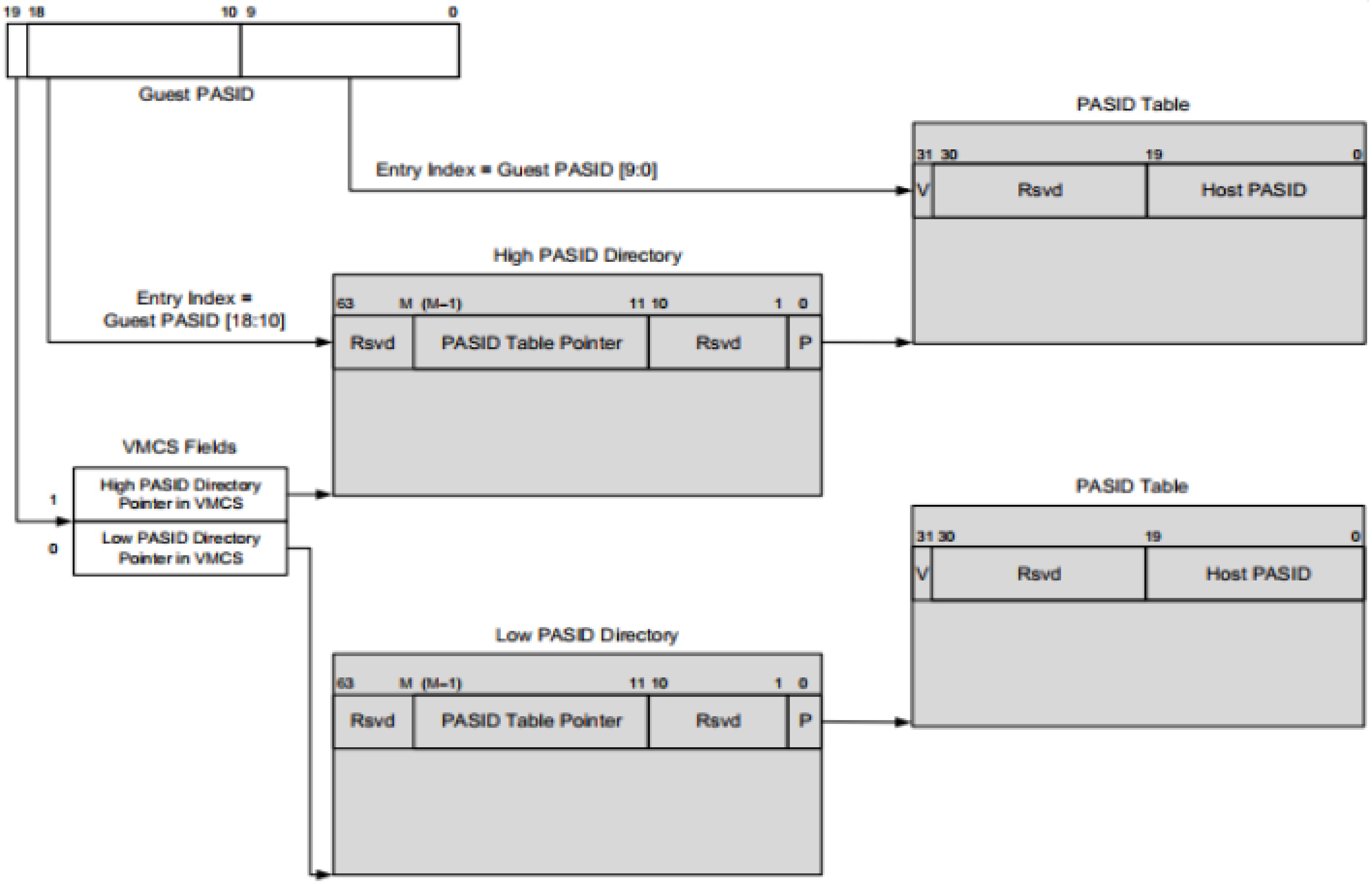
Shared WQ in VM ← PASID in VM ← PASID Virtualization ← vIOMMU

ENQCMD/ENQCMDS

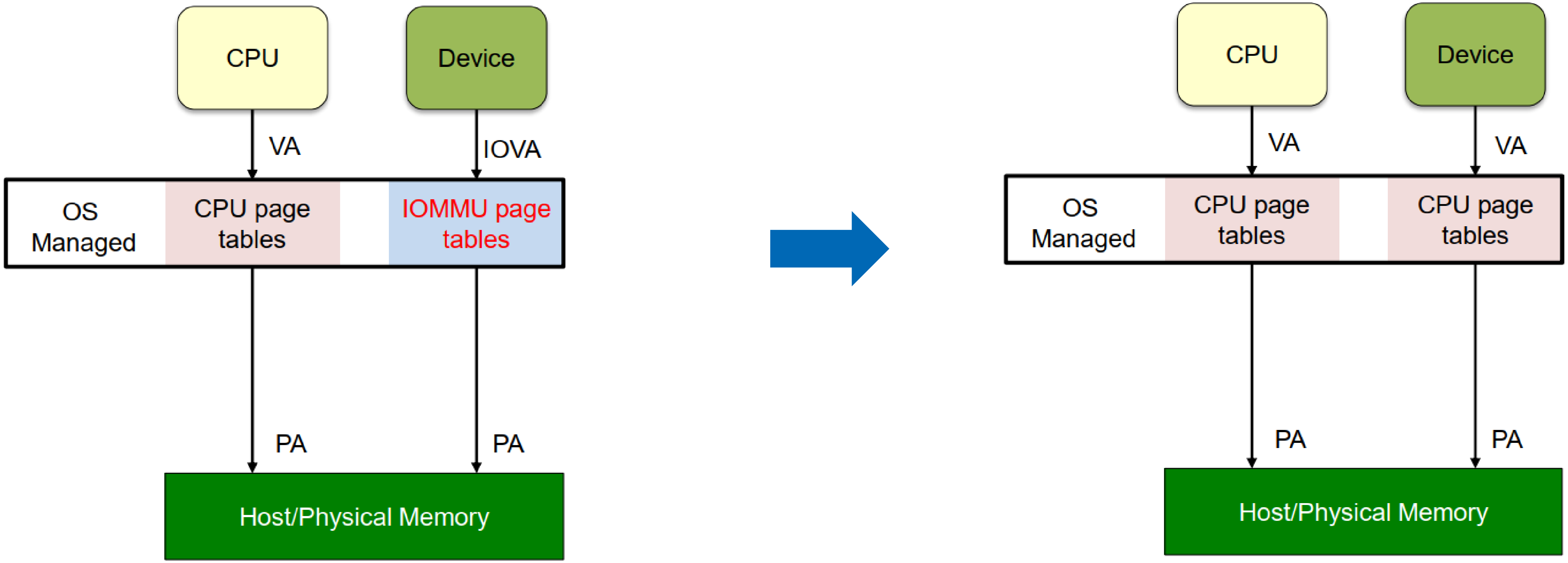
- New instruction on Intel® Platforms
- Atomically submit a work with PASID
  - Obtains PASID from IA32_PASID MSR
  - Enqueue store 64B command to enqueue register in device MMIO
- IA32_PASID is managed by XSAVE as PASID supervisor state
- Non-Posted instruction which carries back a status
  - ZF flag indicates if the command was accepted by device
  - Allows user to retry
- ENQCMDS for Supervisor
  - PASID from command data

ENQCMD in VM

- New feature in VMX on Intel® Platforms
- Use PASID Translation Table for guest PASID to host PASID translation
- Trigger VM-Exit if fails to translate guest PASID



Shared Virtual Address



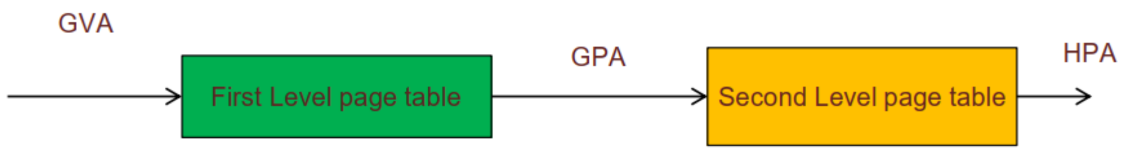
Provide system and device memory unified programming model

SVA Concepts

- Process Address Space ID (PASID)
  - Identify process address space
- First-level translation
  - DMA requests with PASID
  - For SVA transaction from endpoint device (GIOVA/GVA->GPA)
- Second-level translation
  - DMA requests without PASID
  - For normal DMA transaction from endpoint device (GPA->HPA)

Translation Types

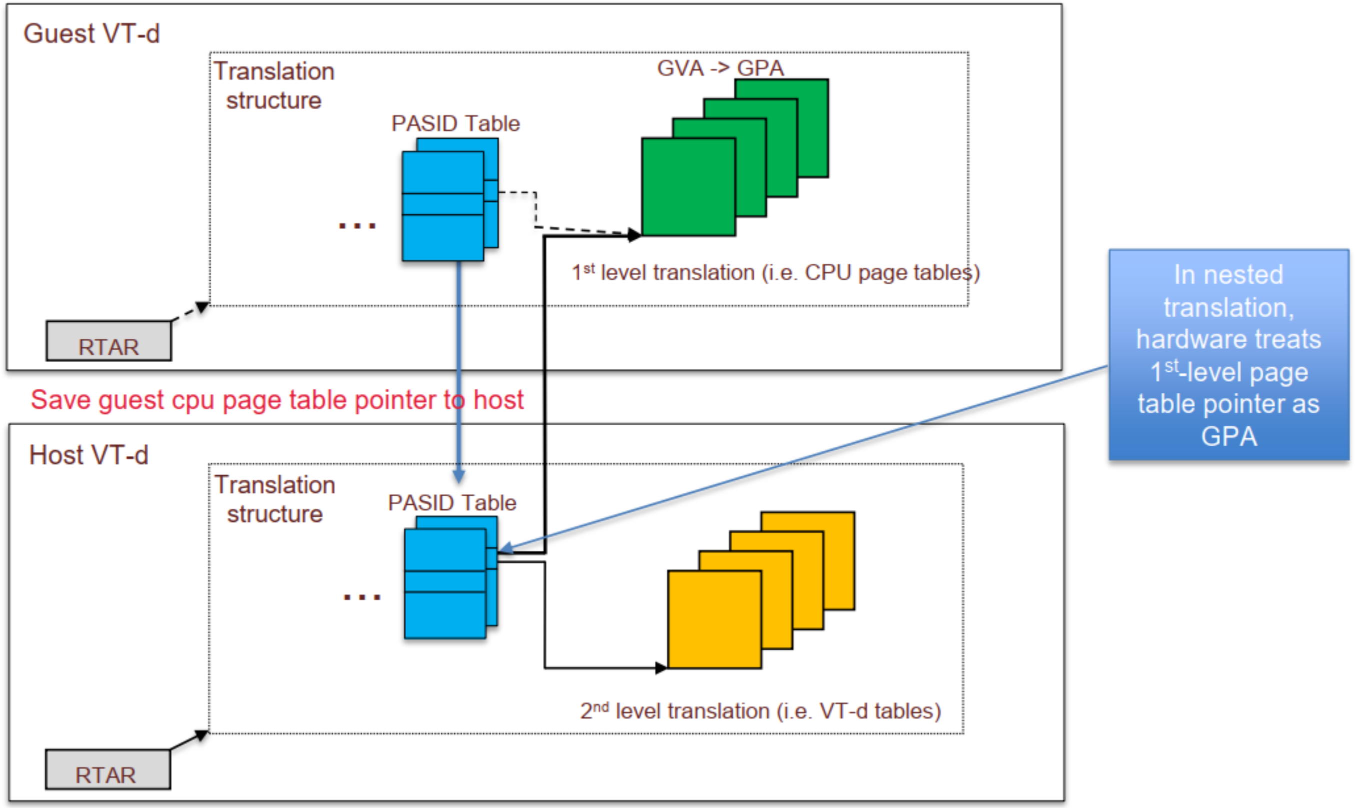
- First-Level translation
- Second-Level translation
- Nested translation - Virtualization environment
  - Need a vIOMMU with SVA capability)
  - Vendor specific
- Pass-Through (address translation bypassed)



SVA in VM

Enable SVA in VM a.k.a vSVA

- vIOMMU emulation in QEMU
- VFIO for programming host IOMMU
- IOMMU Driver with new API to expose VFIO for guest SVA (vSVA)
- Enable nested translation
- PASID virtualization



Guest SVA Support on Intel® VT-d

Native SVA

- Intel iommu driver has SVA support since kernel 5.0

- Generic SVA framework is developed in community
  - Generic kernel API is introduced (iommu_sva_bind/unbind_device())
  - IOASID is introduced to manage PASID (merged in 5.5)
  - Uacce is introduced to support userspace usage (merged in 5.7)

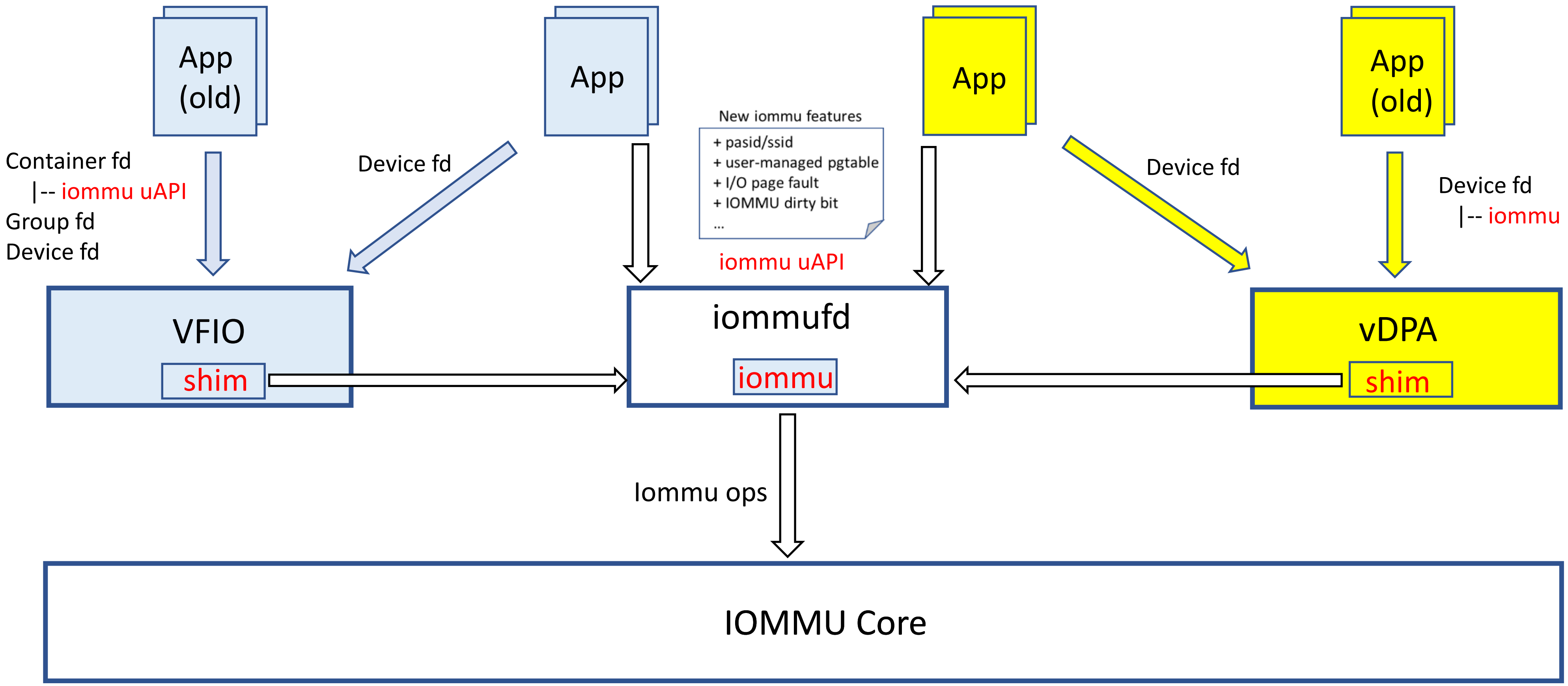
Intel VT-d Enhancement

- Scalable mode DMA remapping
  - PASID granule 1st-level, 2nd-level, nested and pass-thru
  - PASID table now two-level structure
  - Cover both Scalable IOV and SVA usages
    - Extended Context (ECS) is deprecated

- Access/Dirty (A/D) bits in 2nd-level
  - Assist dirty memory tracking in live migration

Upstream Refactoring

Refactoring for a Unified Framework (iommufd)



https://github.com/jgunthorpe/linux/commits/iommufd