AMD-vIOMMU:
A Hardware-Assisted IOMMU Virtualization Technology

Wei Huang

Suravee Suthikulpanit

Agenda

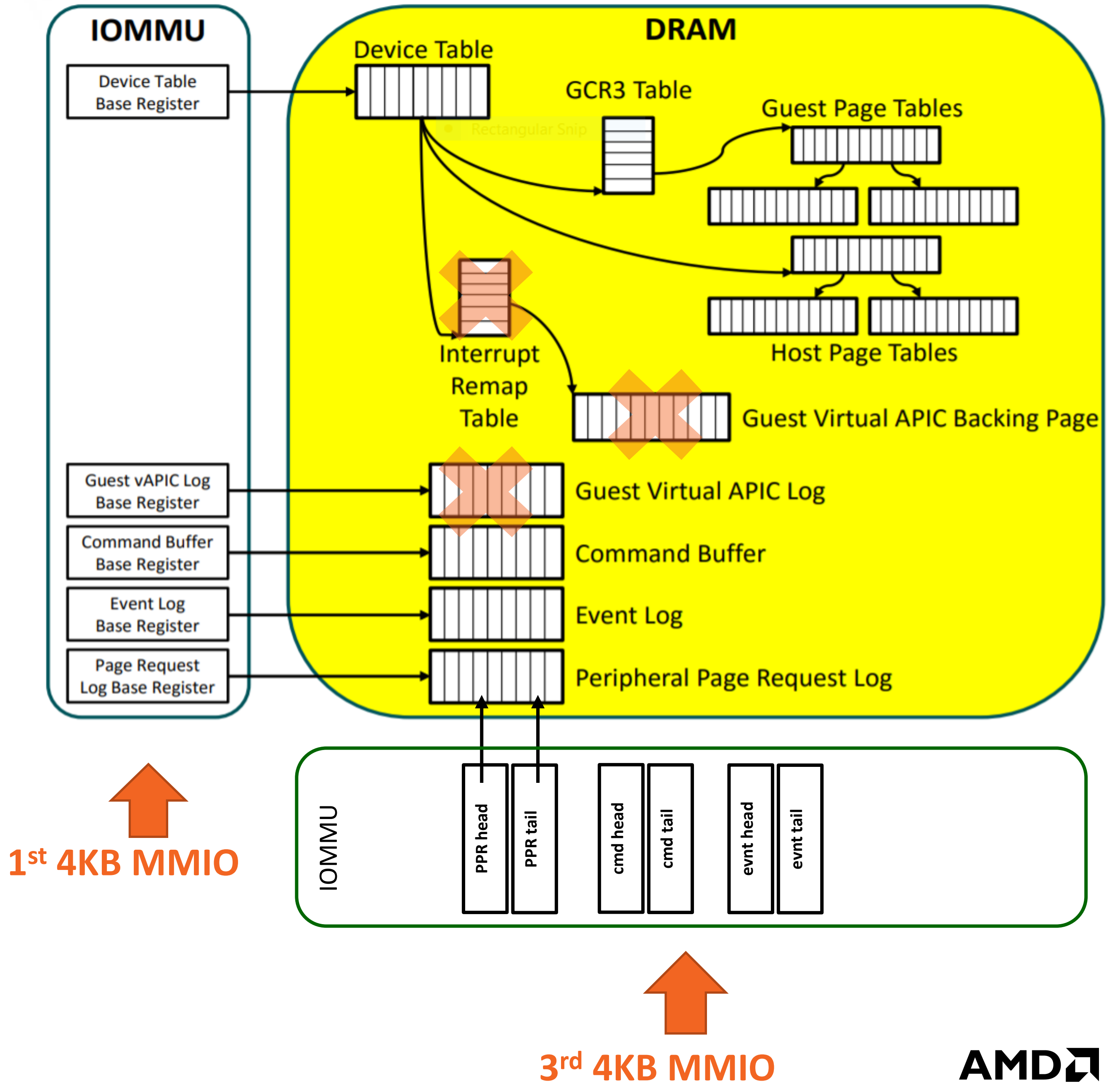
- AMD IOMMU and DMA-Remap Overview
- vIOMMU Support for Guest VM Overview
  - QEMU emulated vIOMMU (a.k.a SW-vIOMMU)
  - HW-assisted AMD-vIOMMU (a.k.a HW-vIOMMU)
- Hybrid (SW + HW) vIOMMU System Model
- Changes to support HW-vIOMMU
  - HW changes
  - Host IOMMU driver changes
  - QEMU changes
  - Guest IOMMU driver changes
- Summary & Discussion

AMD IOMMU DMA-Remap

- MMIO registers
  - 1st 4K region
    - Base and configure registers
  - 3rd 4K region
    - Head and tail pointers of command buffers
    - Head and tail pointers of event and PPR logs
    - Status registers

- Data structures
  - Device table
  - Command buffer
  - Event log
  - Peripheral-Paging-Request (PPR) log

- Two types of IOMMU page table
  - Host IO Page-table (v1)
    - IOMMU-specific (GPA or IOVA -> SPA)
    - Used by Linux DMA-API and VFIO
  - Guest IO Page-table (v2)
    - x86-compatible (GVA -> GPA)
    - Used by Linux KFD driver



IOMMU in the Guest VM

- Use cases
  - Guest IO protection
  - Guest Shared Virtual Memory
  - Nested pass-through devices

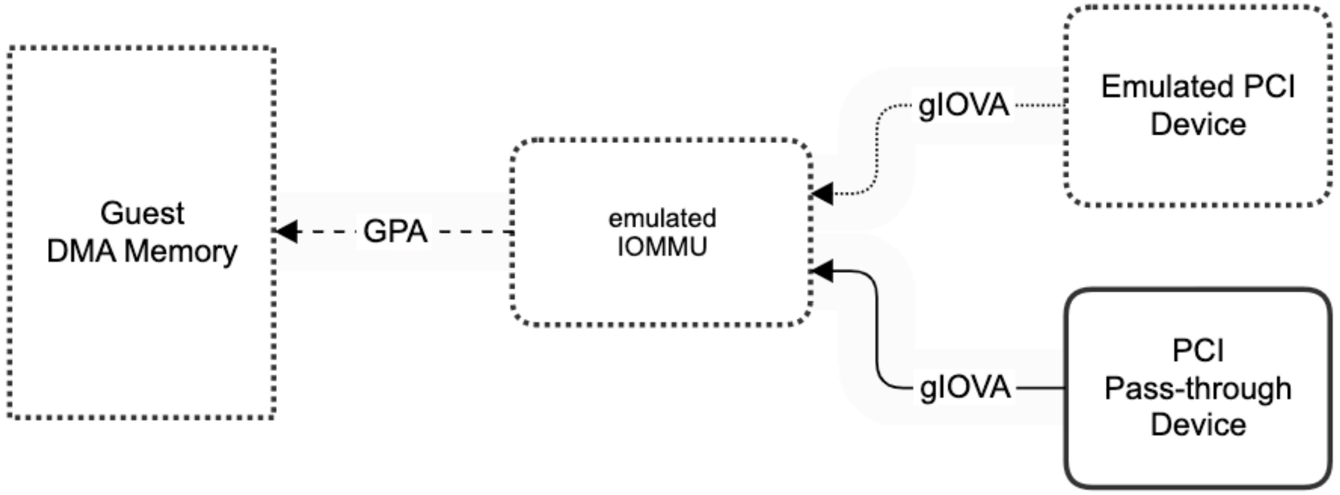
- QEMU SW-vIOMMU models
  - intel-iommu:
    - requires caching-on mode
  - arm-smmu:
    - tlbi-on-map or 2-stage page table support
  - amd-iommu:
    - not fully supported in upstream (work-in-progress)

```
qemu-system-x86_64 -enable-kvm -cpu host -smp 8 -machine q35,kernel-irqchip=split -m 4G \
     -device  ioh3420,bus=pcie.0,addr=1c.0,multifunction=on,port=1,chassis=1,id=root.1 \
     -device  amd-iommu,intremap=on \
     -device  vfio-pci,host=0000:41:00.0 \
```

QEMU Emulated amd-iommu (SW-vIOMMU)

- Two types of SW-IOMMU support in the guest VM

    1. For **Emulated PCI devices**
        - E.g. virtio-net-pci device
        - DMA-remap (emulated v1 page table)
        - INT-remap
    2. For **pass-through PCI devices**
        - E.g. Pass-through 10 GbE NIC
        - DMA-remap (shadow v1 page table)
            - Work-in-progress
        - INT-remap

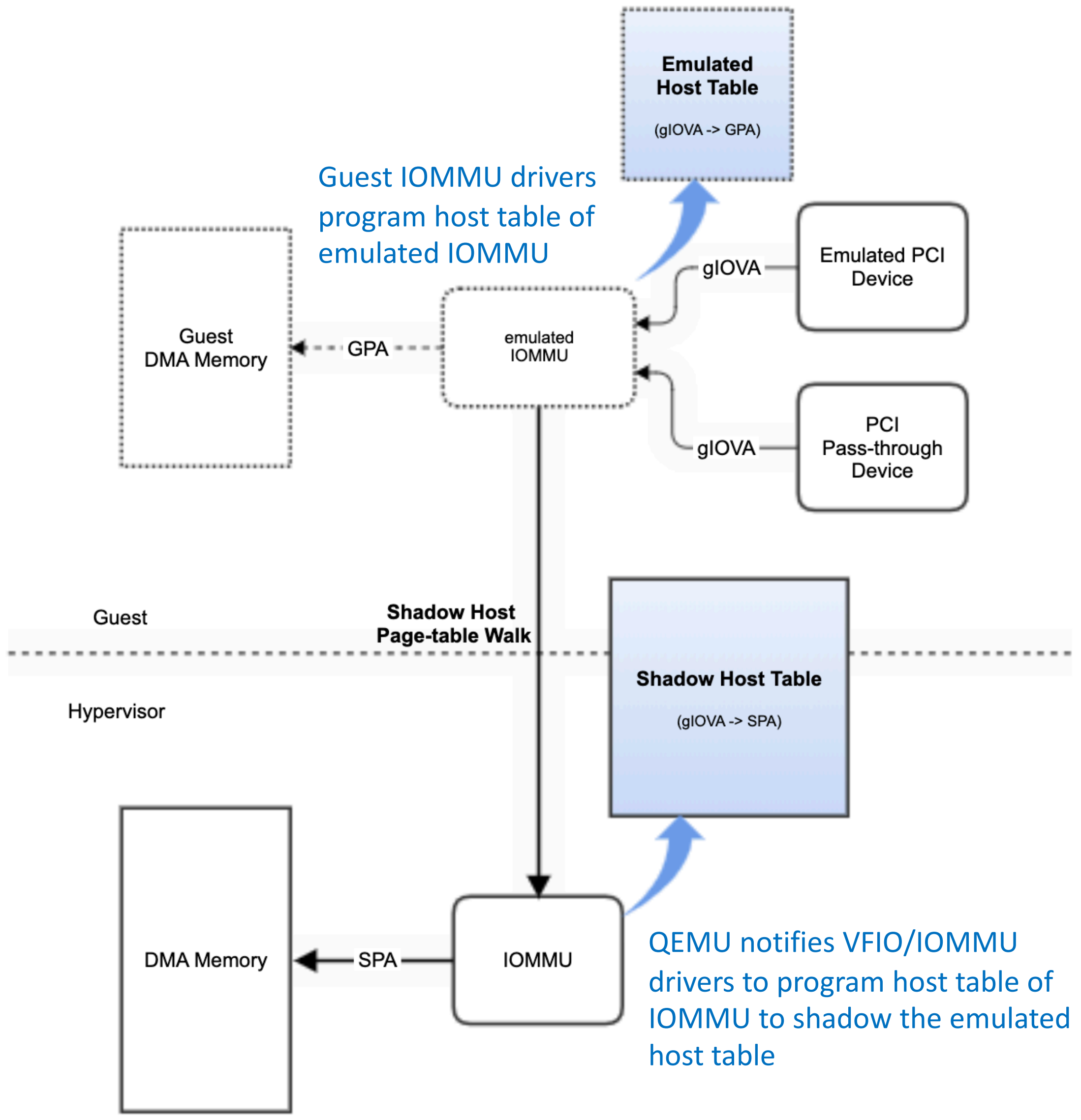


DMA-Remap Support for SW-vIOMMU

- QEMU vIOMMU + Emulated PCI Device
  - Emulated PCI Device uses **emulated host table** in the guest

- QEMU vIOMMU + VFIO PCI pass-through device
  - PCI Pass-through device uses **shadow page-table** in the host IOMMU v1 table

- *Performance hit due to cost of:*
  - *Maintaining **shadow host page table***
  - *Emulating **MMIO accesses***
  - *Emulating **IOMMU command** processing*
  - *Emulating **IOMMU Event and PPR logs** accesses*

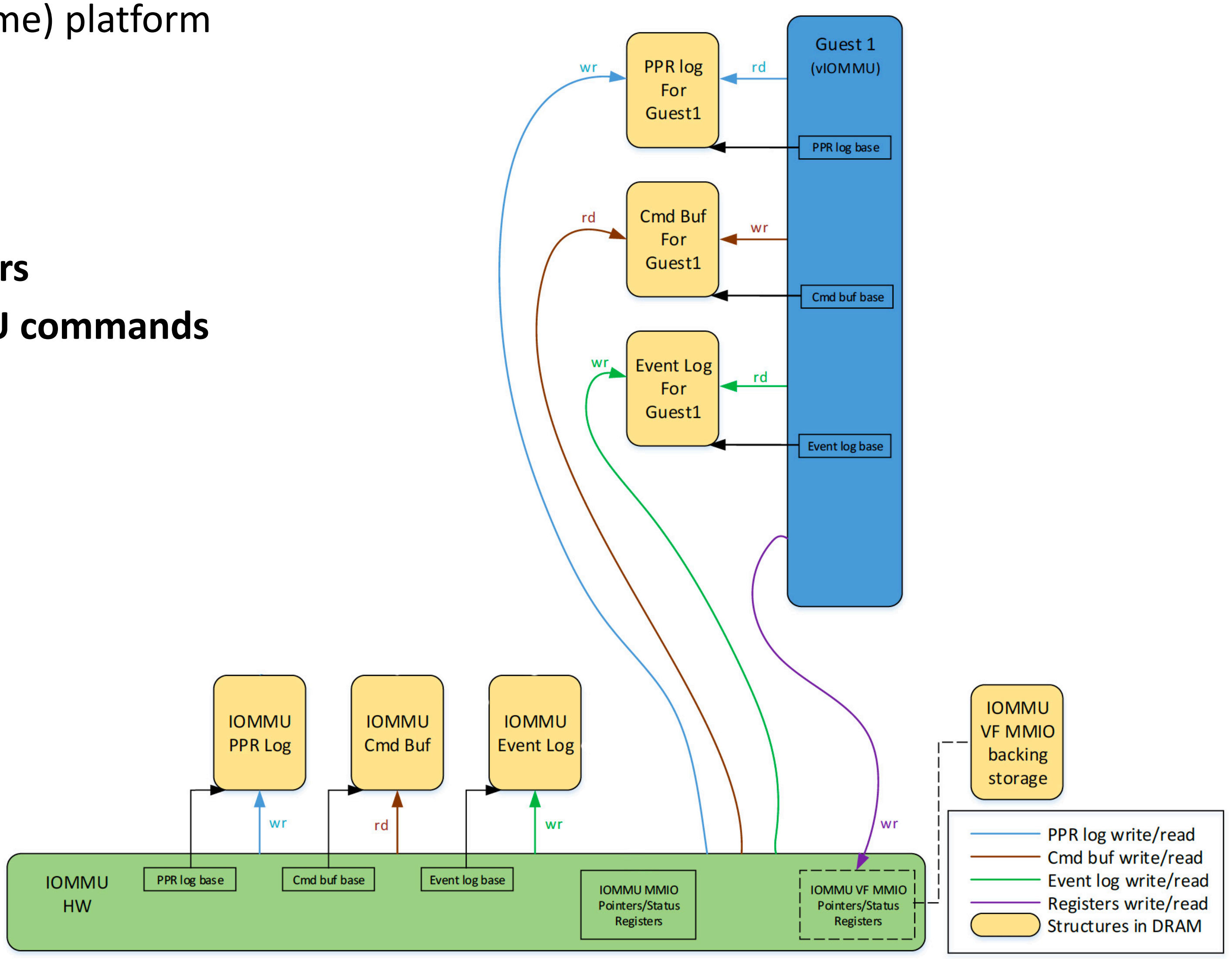


HW-Assisted vIOMMU (HW-vIOMMU)

- Available as of the 2nd generation AMD EPYC (Rome) platform

- Improve performance on guest IOMMU
  - Utilize **Nested IOMMU page-table**
  - Allow guest to directly access **MMIO registers**
  - HW accelerated processing of **guest IOMMU commands**
  - Guest direct access to **Event and PPR logs**

- On-going Development
  - Phase 1: **DMA-remap support**
    - AMD vIOMMU driver (host)
    - QEMU amd-viommu
    - AMD IOMMU driver (guest)
    - Ioctl interfaces
  - *Phase 2: INT-remap support*
  - *Phase 3: Virtual PPR Log support*

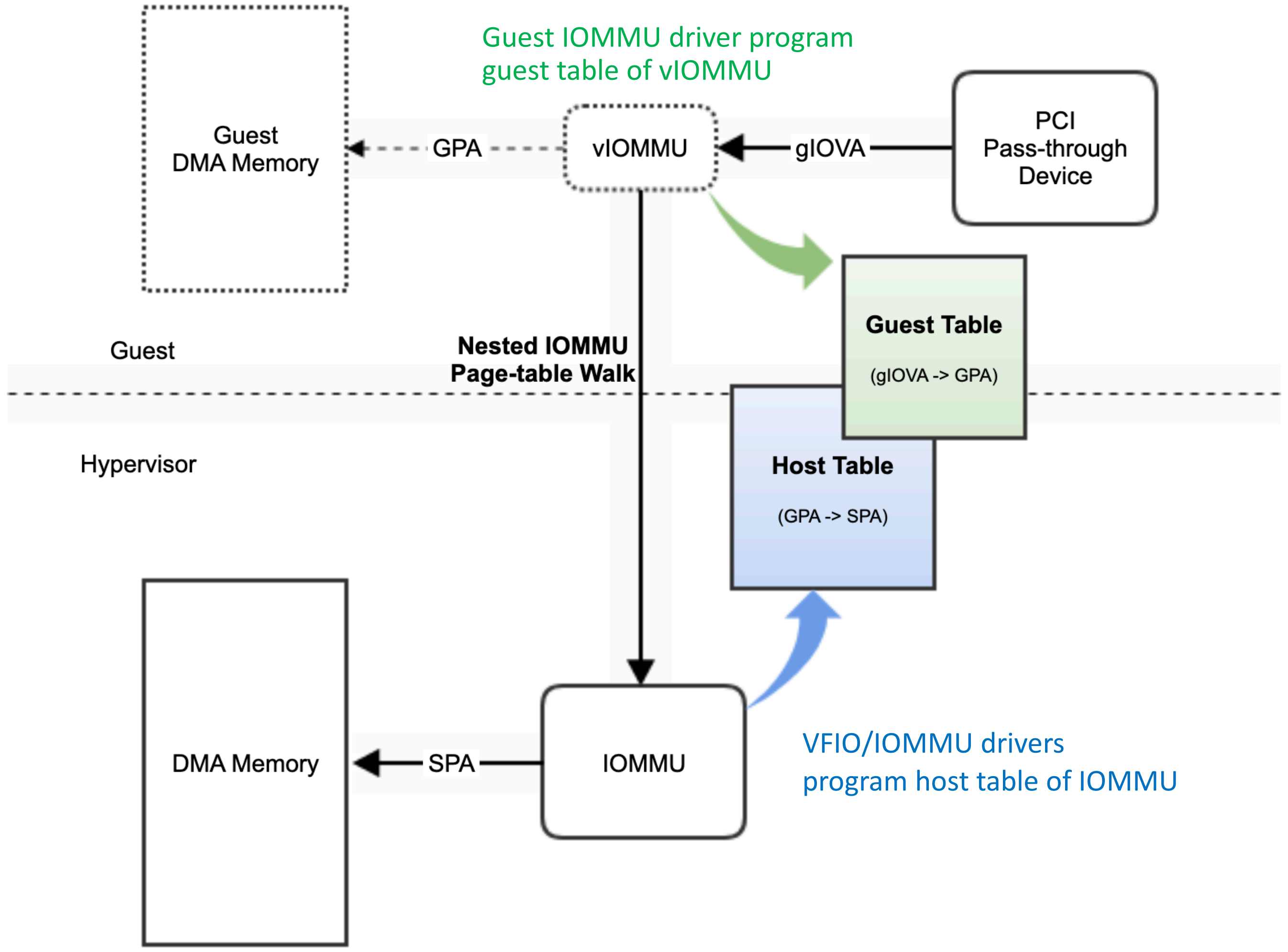


Nested Page-table Support for HW-vIOMMU

- Only support PCI pass-through devices

- No longer use **Shadow Host Table**

- Use **nested IO page-table**
  - Guest (v2) table for gIOVA -> GPA
    - Managed by guest IOMMU driver
  - Host (v1) table for GPA -> SPA
    - Managed by VFIO driver

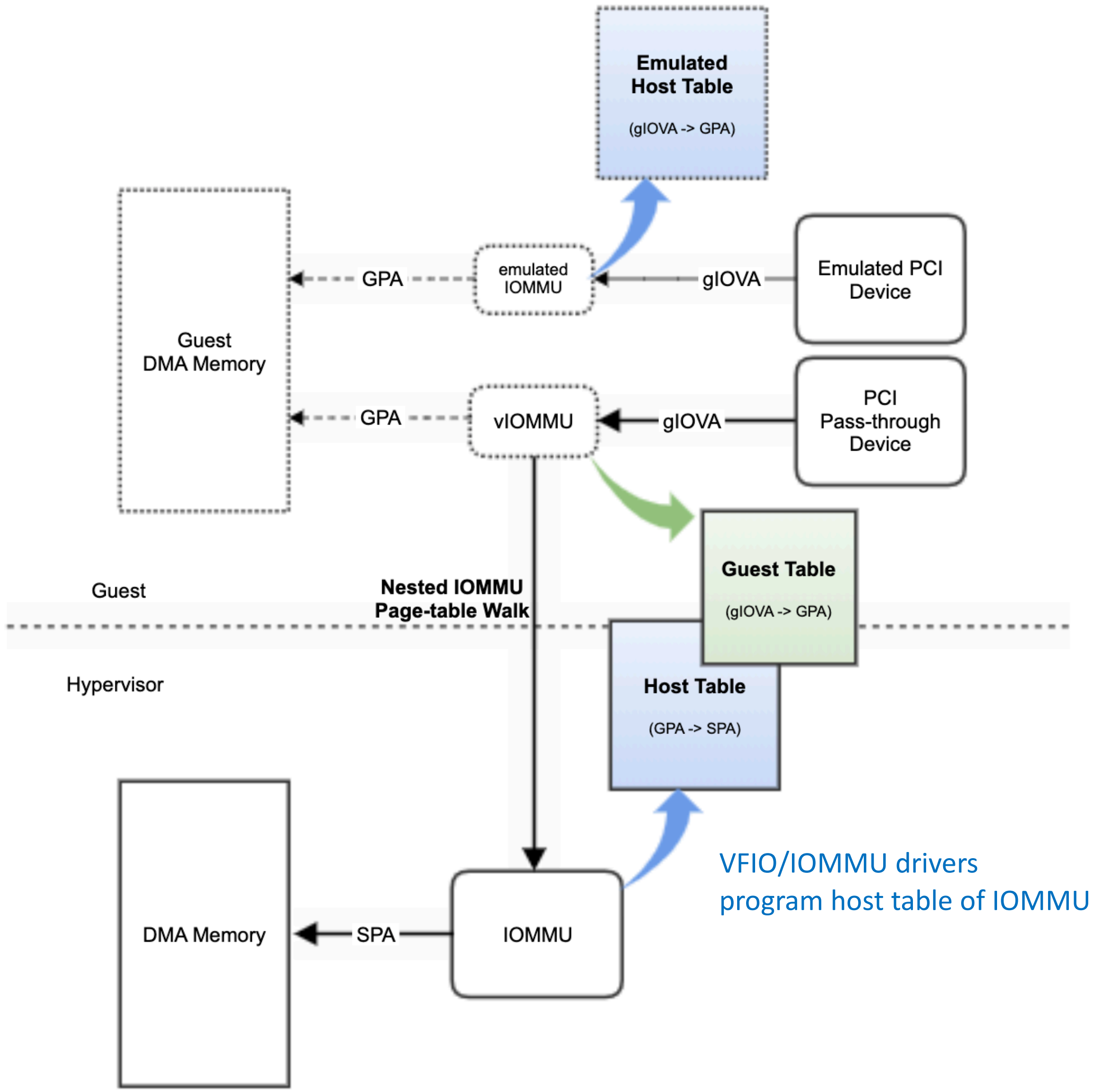


Hybrid System Model : SW+HW vIOMMU

- When support both emulated and pass-through PCI devices in a single VM:
  - amd-iommu (SW-vIOMMU) for emulated devices
  - amd-viommu (HW-vIOMMU) for pass-through devices

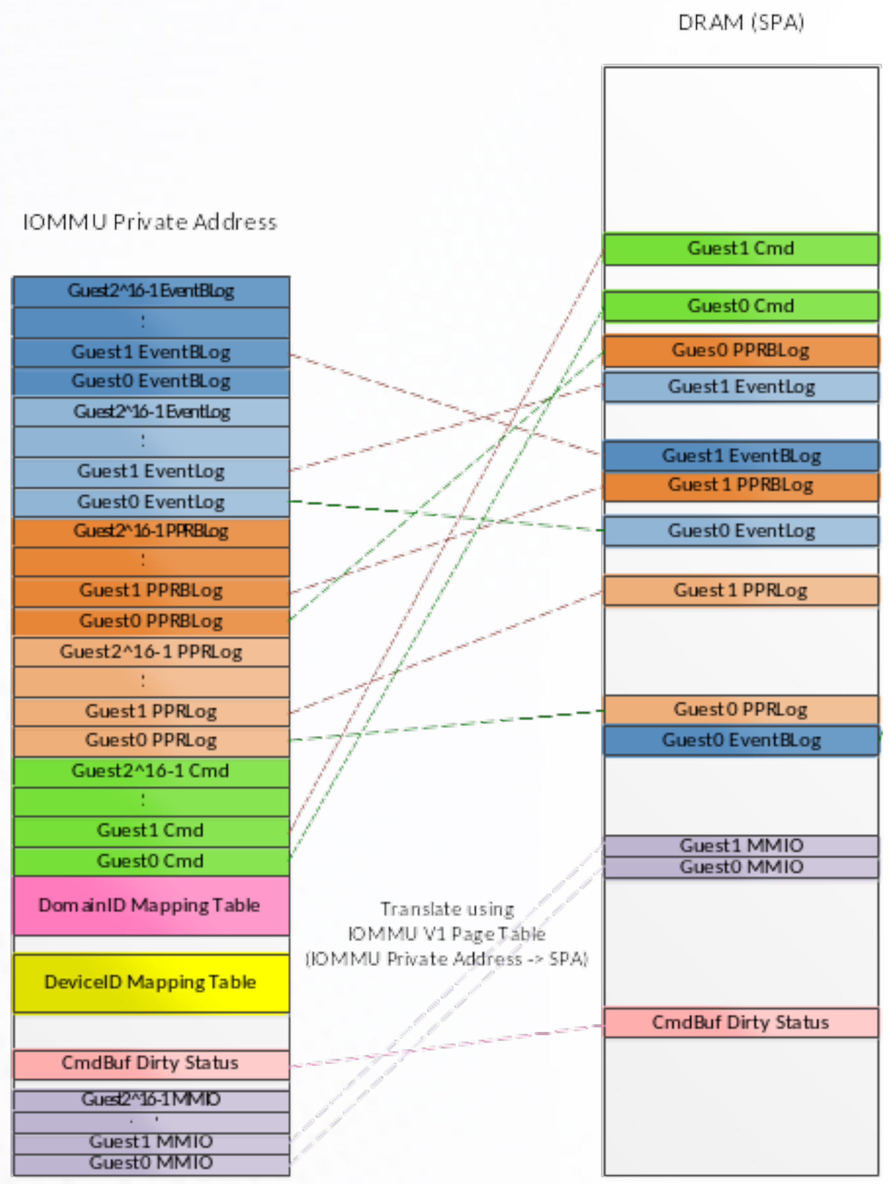
- QEMU walks emulated host table for emulated device

- HW-vIOMMU walks nested IO page-table for PCI pass-through device

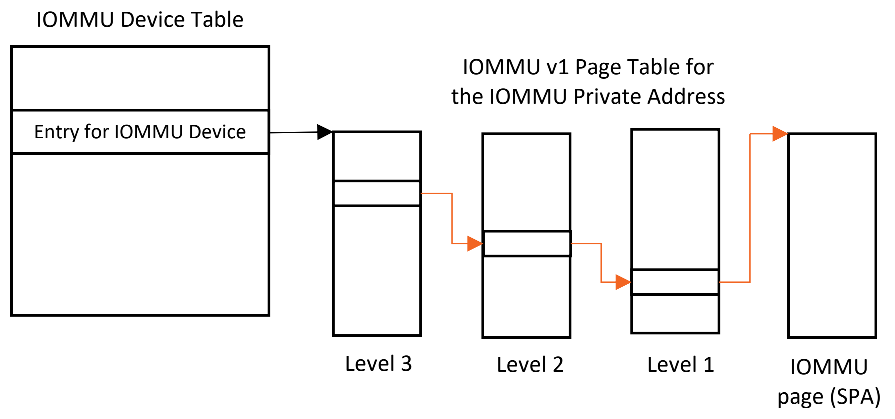


Changes to Support HW-Assisted vIOMMU

HW Changes : IOMMU Private Address Space



- Used by IOMMU hardware
- Per-guest IOMMU data structures
  - Event / PPR logs
  - Command buffer
  - Guest MMIO registers
- vIOMMU-specific data structures
  - Domain ID mapping table
  - Device ID mapping table (pass-through devices)
  - CmdBuf dirty status table
- Mapped by IOMMU v1 page table
  - IOMMU private address -> SPA
- Support up-to 64K VMs



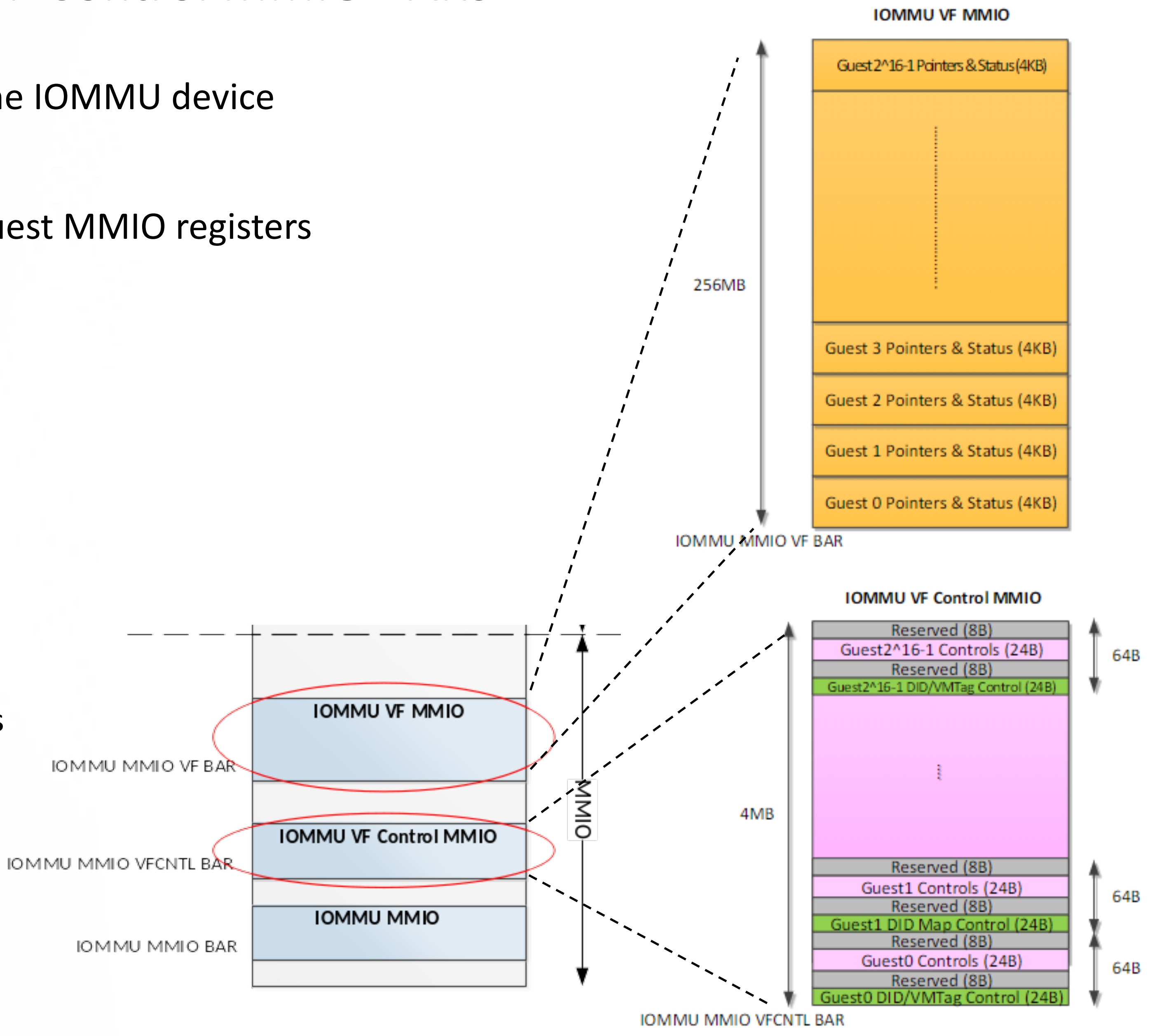
HW Changes : IOMMU VF and VF Control MMIO BARs

- Specified in PCI capability header of the IOMMU device

- Used by hypervisor to access to per-guest MMIO registers (up-to 64K of VMs)

- **VF Control MMIO BAR**
  - Control bits
  - 1st 4KB of guest MMIO registers

- **VF MMIO BAR**
  - Command buffer head/tail pointers
  - Event and PPR Logs head/tail pointers
  - Status registers
  - 3rd 4KB of guest MMIO registers



HW Changes : New IOMMU Commands and Events

- Additional bit fields for existing IOMMU events
  - VCmd, VPpr, VEvent, VNR bits
    - ILLEGAL_DEV_TABLE_ENTRY
    - IO_PAGE_FAULT
    - DEV_TAB_HARDWARE_ERROR
    - PAG_TAB_HARDWARE_ERROR

- Case #1: Inject IOMMU Events into Guest
  - **INSERT_GUEST_EVENT** command
    - IOMMU logs the event into the host event log
    - Hypervisor can use this command to insert the event into a guest event log.
  - **GUEST_EVENT_FAULT** event
    - Invalid INSERT_GUEST_EVENT (e.g. reserved bit not zero)

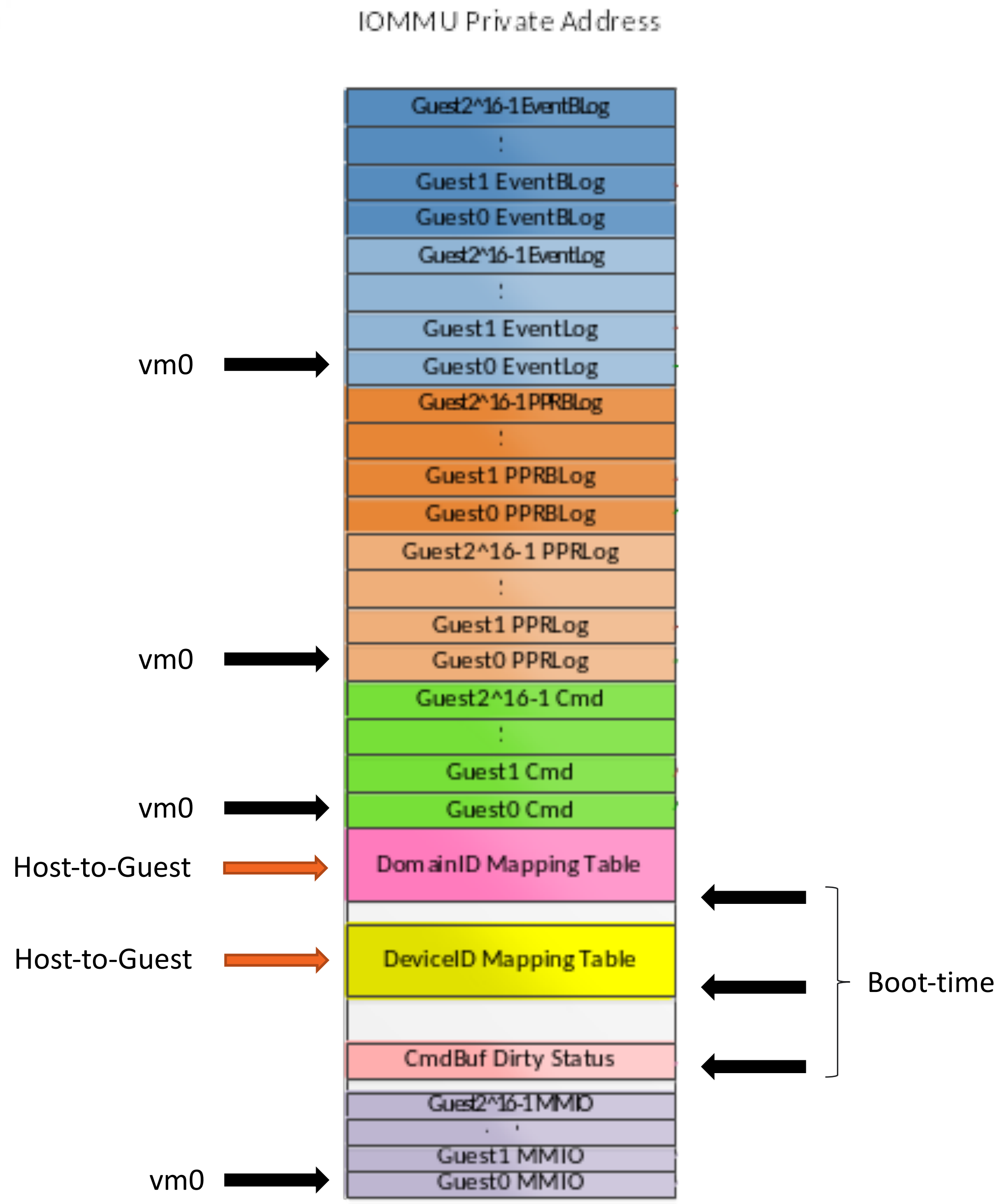
- Case #2: Handle vIOMMU Error
  - **VIOMMU_HARDWARE_ERROR** Event
    - Such as page fault while accessing IOMMU Private Address Space memory
  - **RESET_VMMIO** command
    - Reinitialize a particular guest MMIO registers to recover from error events

Host IOMMU Driver Changes

- Boot-time initialization
  - Logic for detect and enable vIOMMU feature
  - Set up IOMMU v1 page table
    - For IOMMU private address mapping
  - Allocate and map:
    - Domain ID table
    - Device ID table
    - CmdBuf dirty status table

- Per-VM initialization
  - Create Private-Address → SPA mappings
    - Evt/PPR logs
    - Cmd buffers
    - Guest MMIO registers
  - Host-to-Guest DevID and DomID mappings
  - Trap guest access to 1st 4K of MMIO region (control)

- Support new IOMMU commands and events

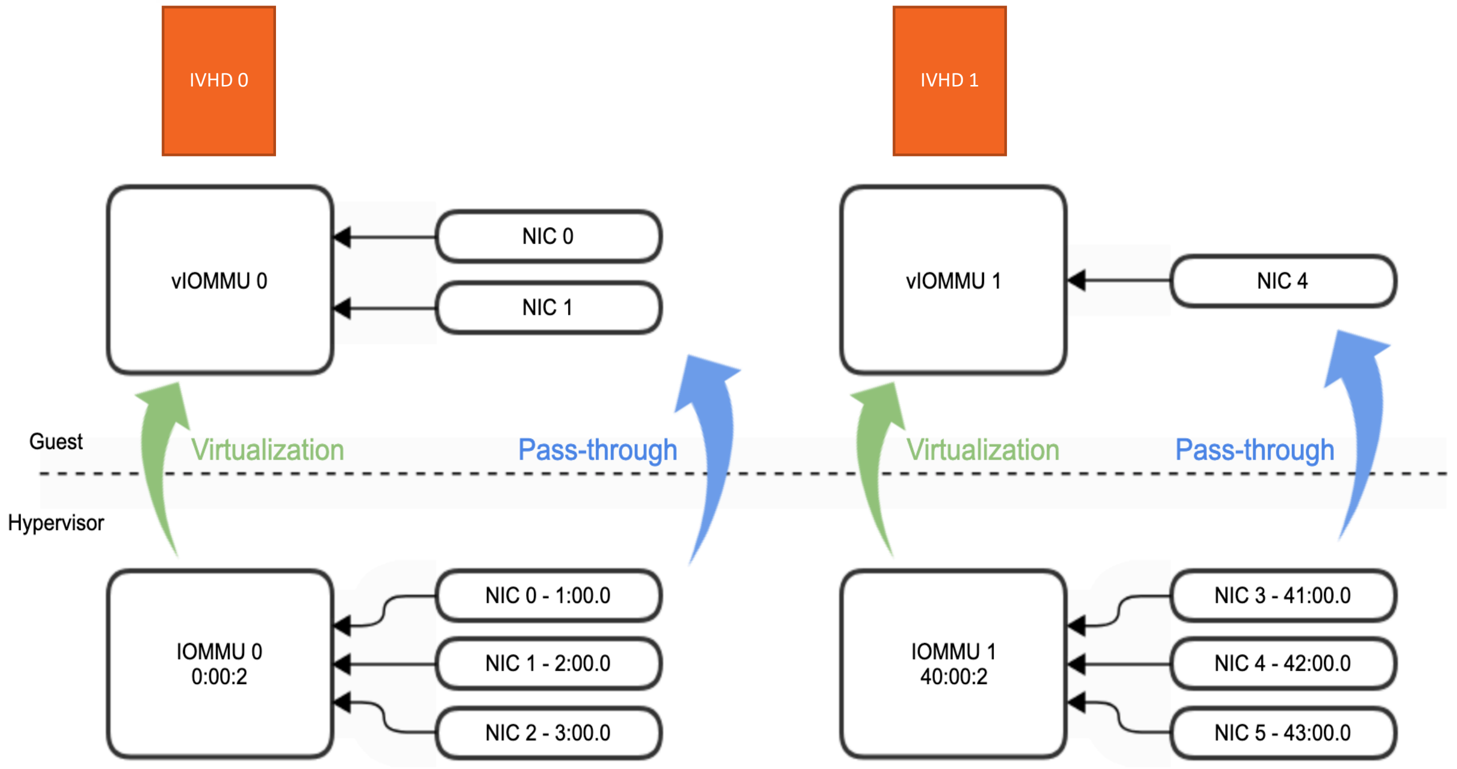


QEMU Changes : New HW-vIOMMU Device Model

- Different model for SW vs HW-vIOMMU

- Need to specify PCI topology for the VM
  - Need to specify multiple amd-viommu instances
  - Each PCI pass-through device must be associated with the corresponding amd-viommu instance

- New ACPI IVRS table to support additional IVHD blocks for the new amd-viommu device model



```
qemu-system-x86_64 -enable-kvm -cpu host -smp 8 -machine q35,kernel-irqchip=split -m 4G \
     -device  ioh3420,bus=pcie.0,addr=1c.0,multifunction=on,port=1,chassis=1,id=root.1 \
     -device  amd-viommu,host=0000:00:00.2,id=viommu0 \
     -device    vfio-pci,host=0000:01:00.0,iommu=viommu0 \
     -device    vfio-pci,host=0000:02:00.0,iommu=viommu0 \
     -device  amd-viommu,host=0000:40:00.2,id=viommu1 \
     -device    vfio-pci,host=0000:42:00.0,iommu=viommu1
```

QEMU / Host AMD IOMMU Driver Changes : New Dev-FS and IOCTL

- Introduce new device FS
  - /dev/amd-viommu

- Introduce vIOMMU-specific IOCTL interfaces
  - VIOMMU_VM_[INIT|DESTROY]
  - VIOMMU_DEVICE_[ATTACH|DETTACH]
  - VIOMMU_DOMAIN_[ATTACH|DETACH]
  - VIOMMU_MMIO_ACCESS
    - For the trapped guest MMIO access
  - VIOMMU_CMDBUF_MAP_UPDATE

Guest AMD IOMMU Driver Changes

- HW-assisted vIOMMU implementation does not virtualize IOMMU page translation.
  - Instead, it uses nested IOMMU page table (i.e. gIOVA -> GPA -> SPA)
    - The v1 table is already used by VFIO for pass-through device (i.e. GPA -> SPA)
    - The v2 table is managed by the guest IOMMU driver (i.e. gIOVA -> GPA)
  - Current AMD IOMMU driver only support the use of v1 table for DMA-API
    - E.g. to implement struct iommu_ops

- Part1: Leverage the existing generic IO page table framework
  - Introduced by ARM SMMU drivers
  - Re-factor current AMD IOMMU driver (for IOMMU v1 page table)
    - https://lore.kernel.org/linux-iommu/20201004014549.16065-1-suravee.suthikulpanit@amd.com/T/#t

- Part2: New v2 page table support for DMA-API
  - Also use the generic IO page table framework
  - Work-in-progress

Summary & Discussion

- Summary of SW-vIOMMU vs. HW-vIOMMU
  - Shadow host page table → Nested (v1/v2) I/O page table
  - Emulating guest MMIO accesses → Guest directly access MMIO
  - Emulating guest IOMMU cmd/evt/ppr buffers → HW-virtualized cmd/event/ppr buffers

- Topics for Discussion
  1. The new HW-vIOMMU device model (amd-viommu)
  2. Hybrid vIOMMU system model (i.e SW + HW vIOMMUs)
  3. The IOCTL interface design
     - Leverage existing VFIO IOCTL interface or define a new one?
  4. Additional usage models based on the proposed design

Thank you

DISCLAIMER AND ATTRIBUTIONS