Live Migration with Mdev Device

Yulei Zhang
yulei.zhang@intel.com

Intel® Open Source Technology Center

Background and Motivation

Live Migration Desgin of Mediated Device

vGPU Live Migration Implementation

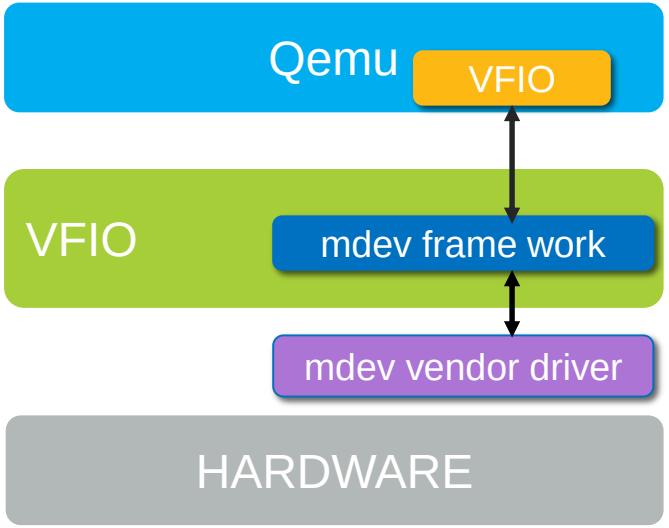
Current Status and Demo

Future Work

Mediated Device

1. Mediated device leverages the VFIO framework to build a lot of virtual devices.
   - GPU, network adapter, compute accelerators

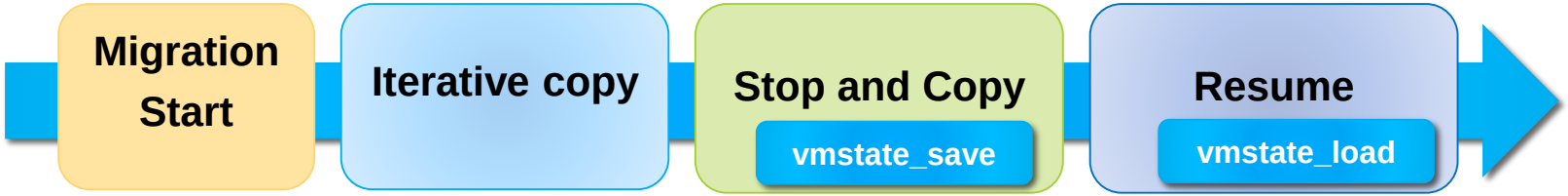
2. There is urgency to support live migration when use mediated devices in data center and cloud.



Mediated Device State to be Migrated

1. Emulated vmmio state
2. Hardware state on the device(e.g.Graphics memory)
3. Pending interrupt
4. Dirtied memory pages

General Migration Flow



- Common virtual device put its states in the fields of VMState for migration

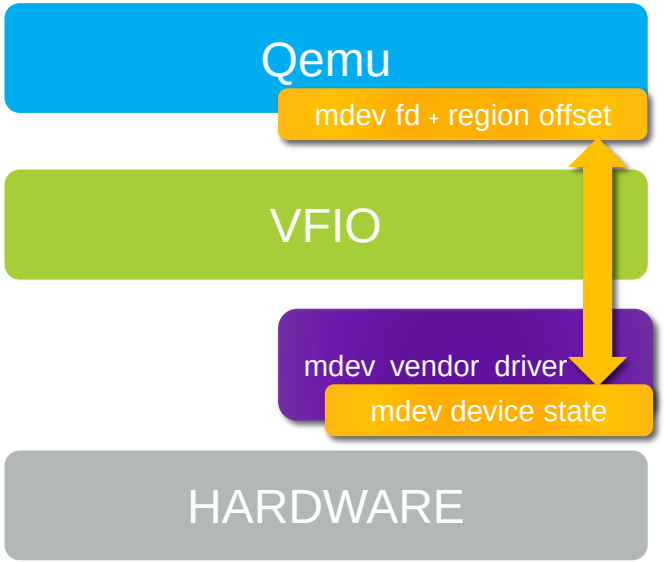
Mediated Device Migration Flow

1. Retrieve and restore the mdev context
2. Stop and resume the mdev
3. Dirtied memory pages copy

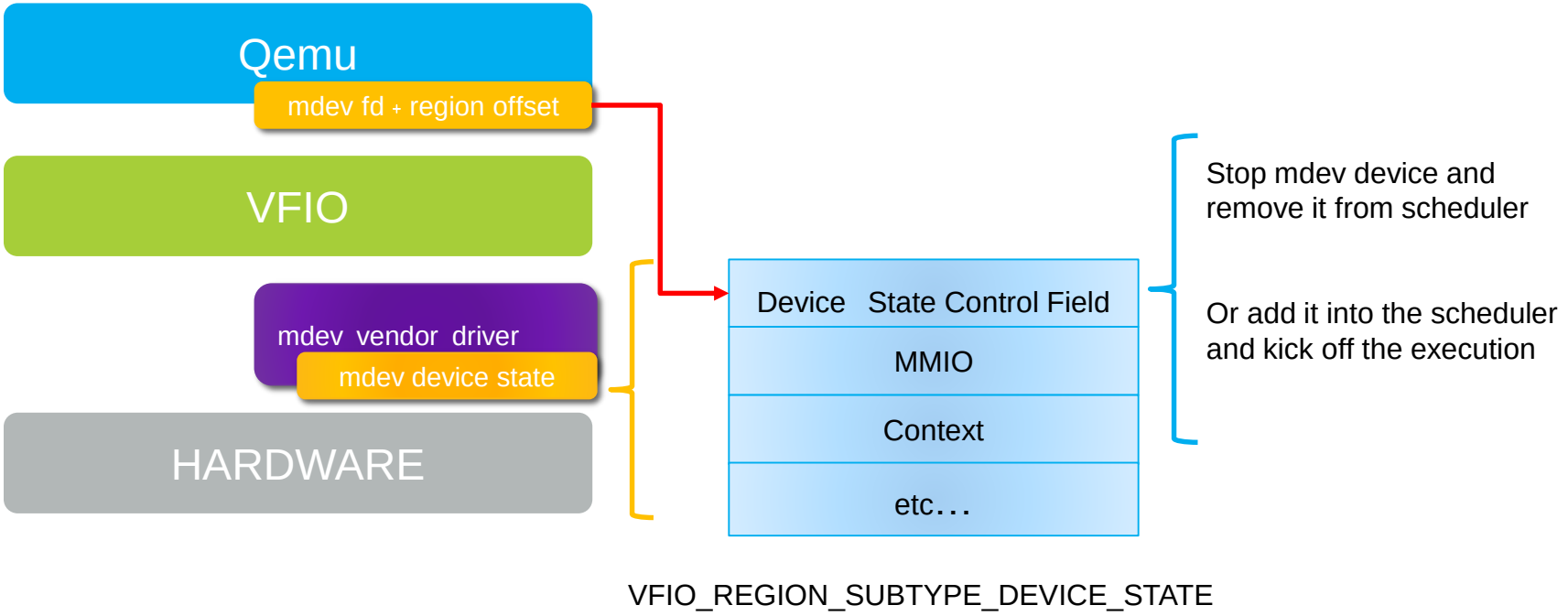
New VFIO Region for Mdev Device Context Transmit

1. VFIO region maps to device resource(MMIO, PCI configuration space)

2. Register a new vifo subregion for device context transmit, also can control the device running state
   - VFIO_REGION_SUBTYPE_DEVICE_STATE

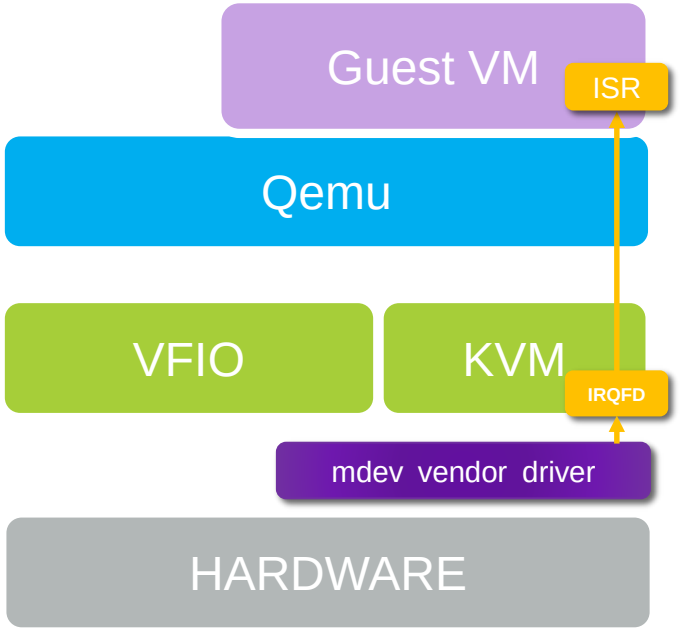


Stop and Resume the Mdev Device



Resume Mdev: PCI Configuration Region Reconstruction

1. In VFIO framework PCI configuration Region setup is trapped and emulated by Qemu
2. On the target VM side, Qemu need go through the same PCI configuration region to construct the device resource map and virtual interrupt injection patch.



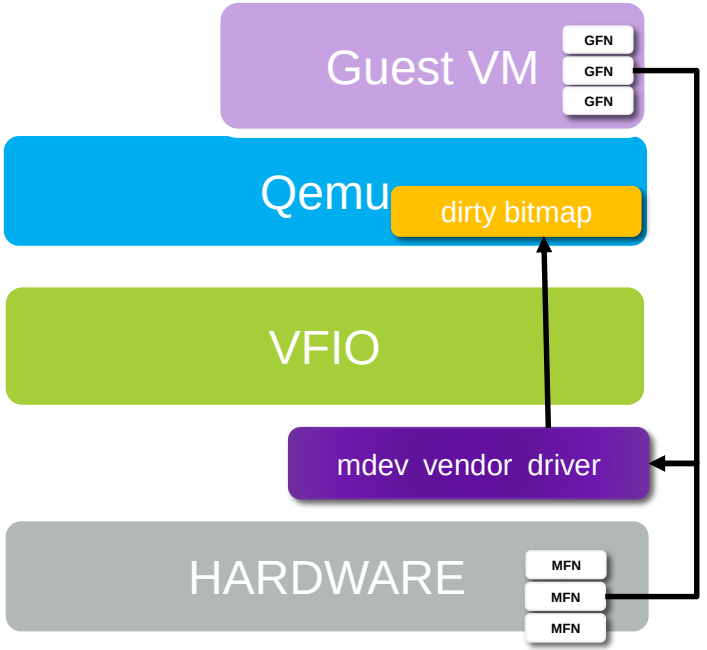
Dirtied Memory Pages

1. Vendor mdev driver report the dirty bitmap.
2. Query the memory mapping from vfio iommu driver and build up the dirty bitmap.

Vender Driver Report the Dirty Bitmap

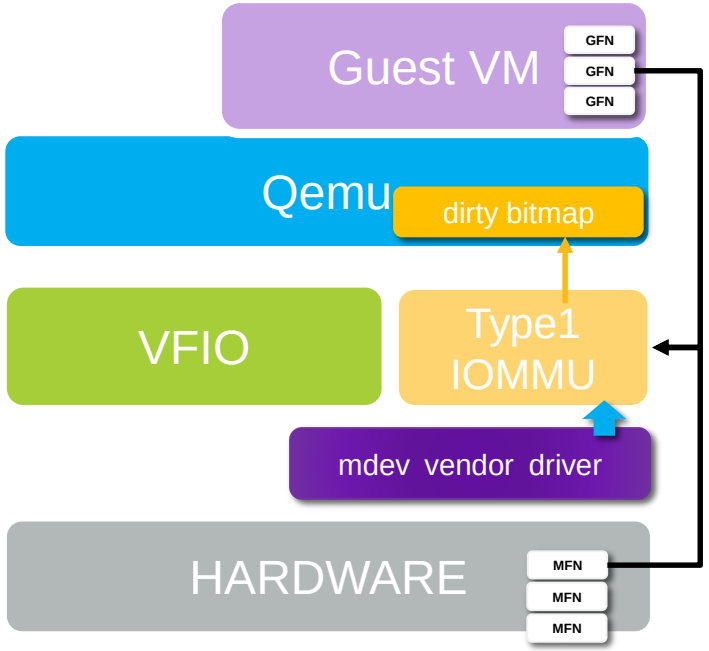
1. Vender driver create shadowing page table for DMA operation.

2. Vender driver can track the guest page used for DMA and build up dirty bitmap for migration.

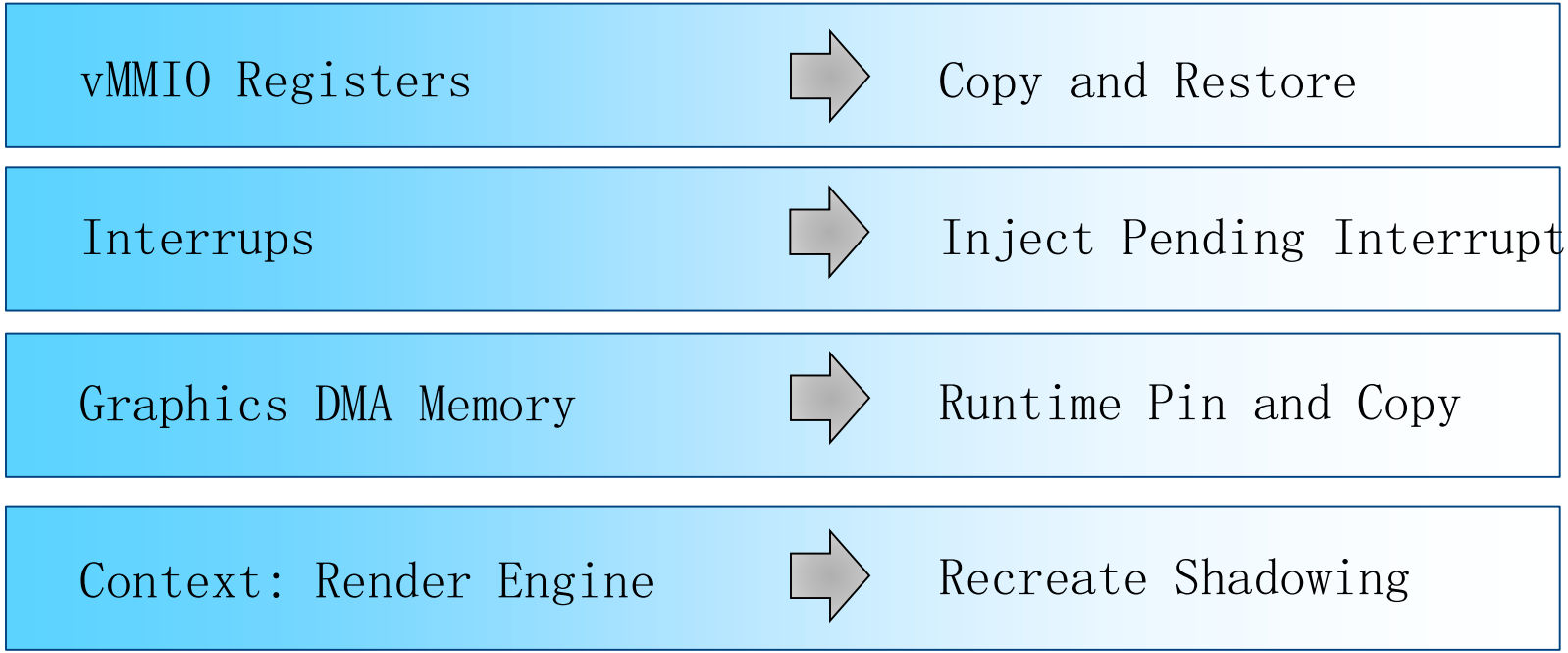


Query the Memory Mapping from VFIO IOMMU Driver

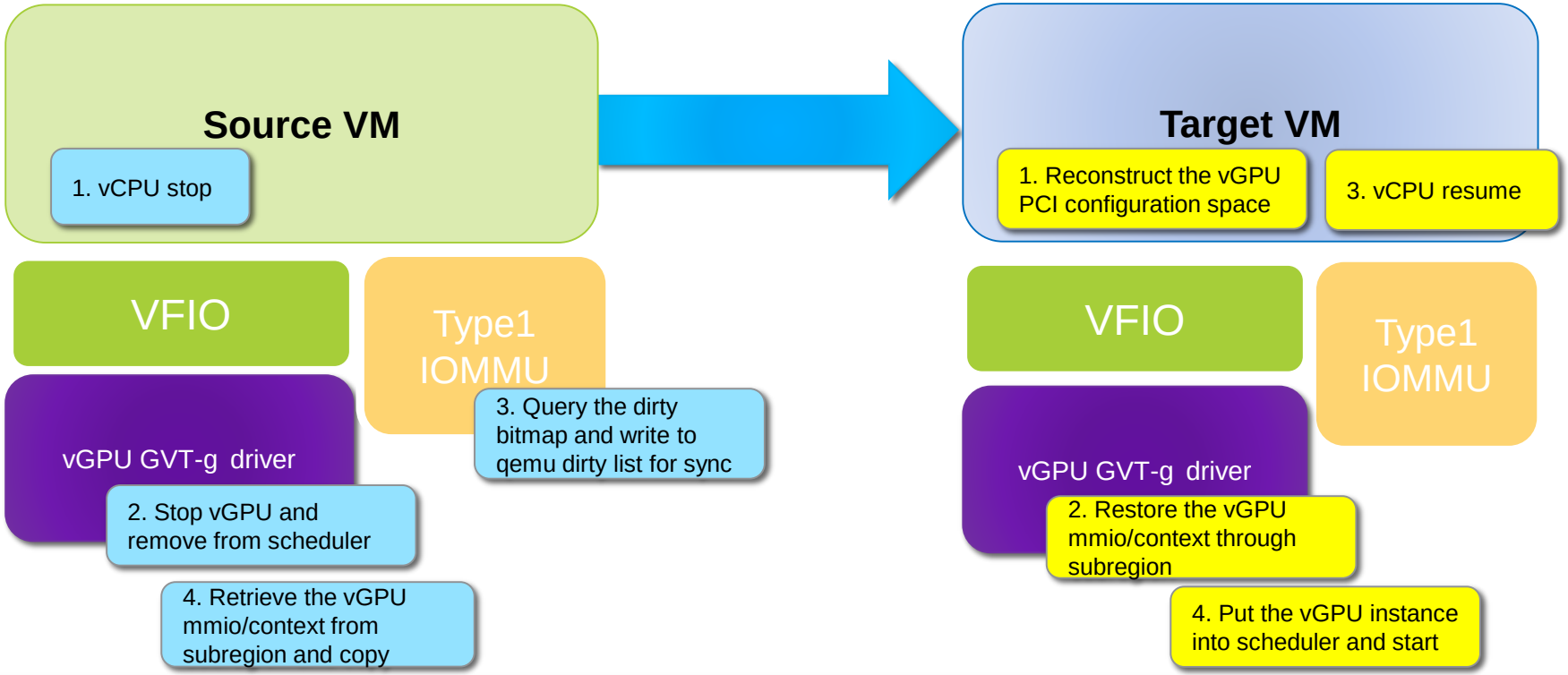
1. Vendor driver will pin the memory in runtime for DMA usage. Those mapping is tracked by vfio iommu driver.
2. During migration qemu could query it from vfio iommu driver and build up dirty bitmap to transfer the DMA memory used by mdev.



Migration Policies for vGPU Resources



vGPU Live Migration Flow



Current Status

1. Experimental vGPU live migration support for KVM and XEN
2. Platforms: Intel® 6th /7th Generation Intel® Core™ Processors
   - Benchmarks covered:
     - Heaven, 3Dmark06, Trophic, Media encoding/decoding
     - lightsmark, 2D, Media workload

Demo:vGPU live migration with 3D Workload

1. Platforms: Intel® 6th Generation Intel® Core™ Processors
2. Guest VM has 2 vcpus, 2G RAM and 512MB graphics memory
3. Physical machines were connected through 10Gbps network adapter
4. Services downtime less than 500ms, total migration time is about 2.5~3s

Demo:vGPU live migration with 3D Workload

Future Work

1. RFC has been sent out, try to upstream the current implementation for vGPU live migration
   a. https://lists.gnu.org/archive/html/qemu-devel/2017-07/msg09242.html
2. Leverage the current work to support passthrough/SRIOV device migration

Project webpage and release: https://01.org/igvt-g

Q & A