redhat.

Virtual IOMMU Implementation using HW Nested Paging

Eric Auger

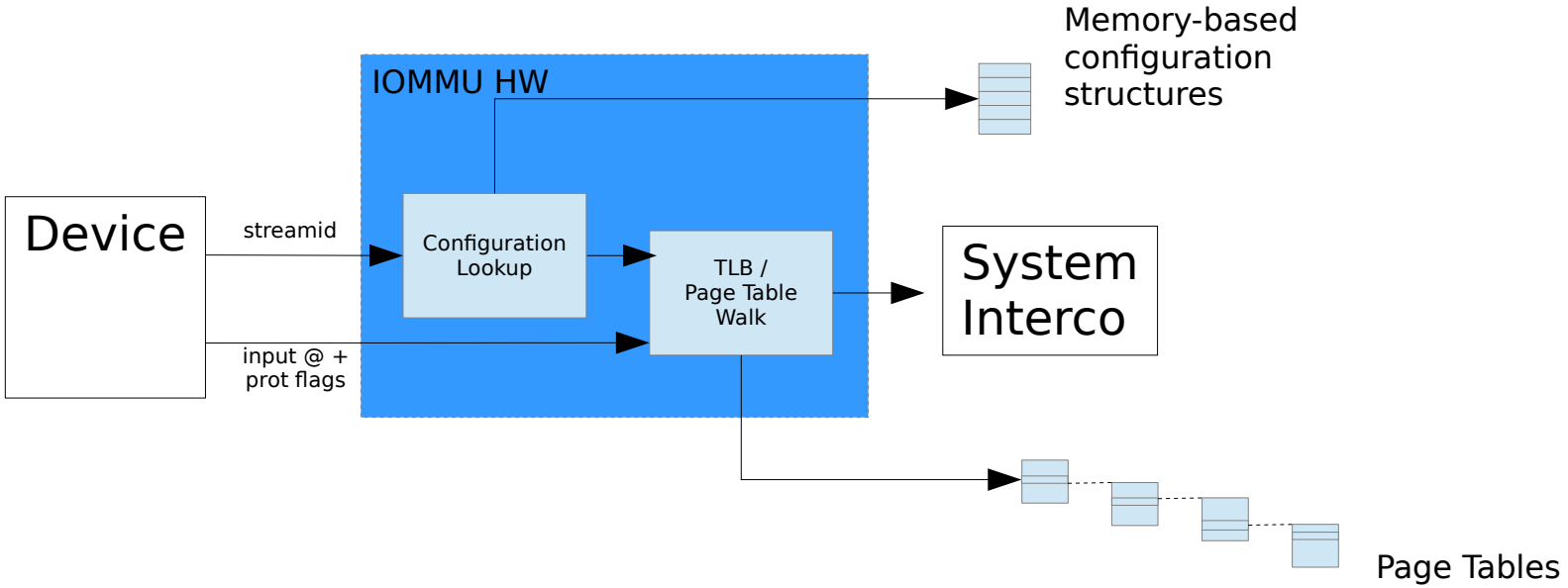
FOSDEM Feb 2019

Overview

- IOMMU Overview & Terminology
- Linux VFIO/QEMU Device Passthrough
  - Physical IOMMU Role
  - Virtual IOMMU benefits
- Virtual IOMMU/Device Assignment Implementation Choices
- SMMUv3 HW Nested Paging Enablement at Linux/Qemu
- Status & Challenges

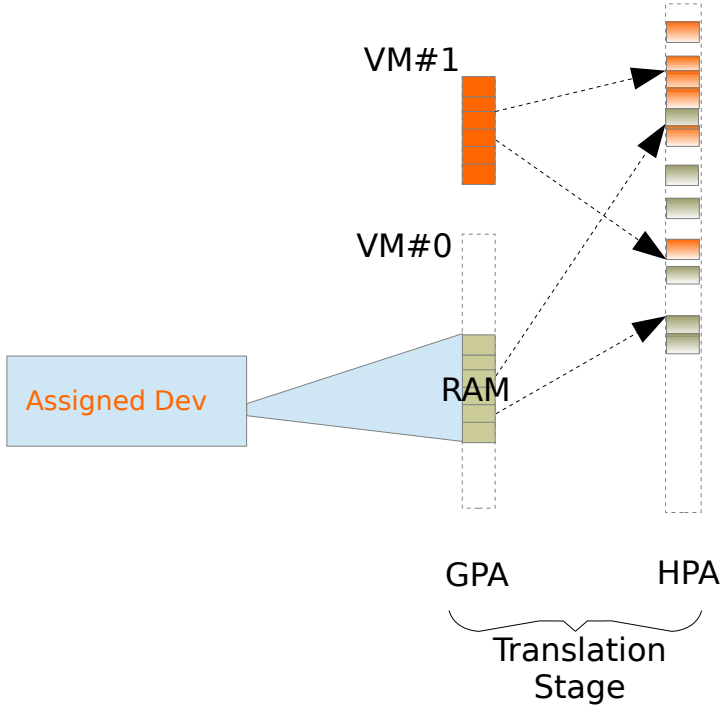
Terminology & Goals

IOMMU Overview



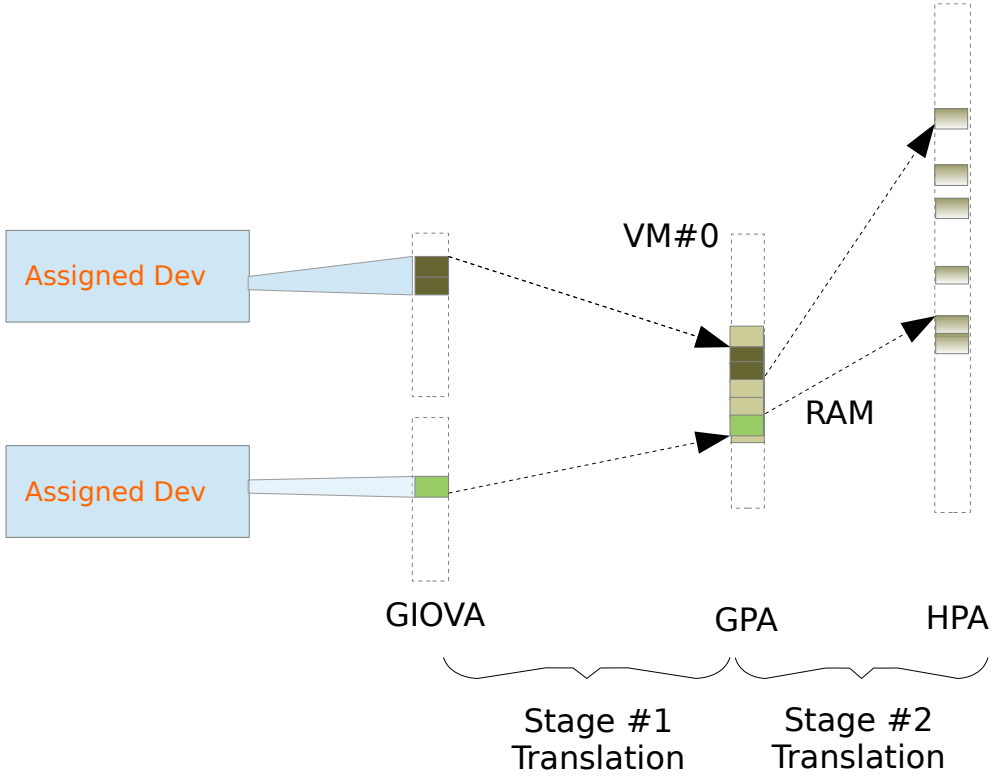
Device Assignment & Physical IOMMU

- Guest directly interacts with the physical device
  - DMA target address is programmed in GPA
- VMM transparently maps the whole guest RAM through the physical IOMMU for assigned devices
- Assigned devices cannot reach any other HPAs

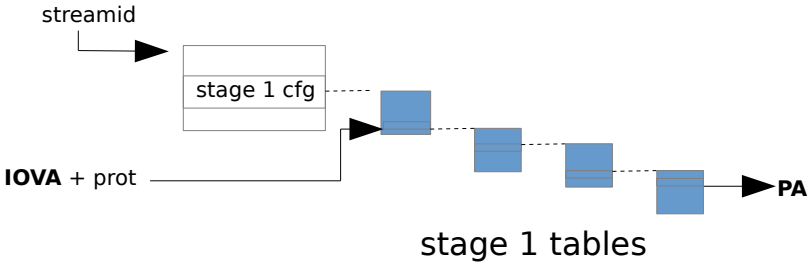


Adding the Virtual IOMMU

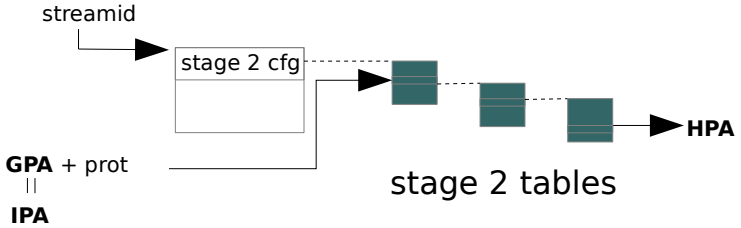
- DMA Isolation within the VM
- Guest drivers can program the device with IOVA (scatter-gather commodity)
- Linux VFIO can be used on guest
  - DPDK user-space drivers
  - Nested assignment to an L2 guest
- Nested logical stages
- Both logical stages must be mapped through the physical IOMMU



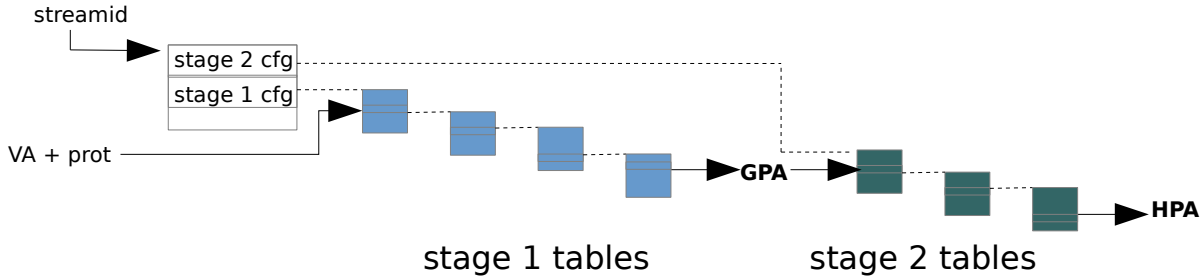
Translation Stages and HW Nested Paging



stage 1 Only (OS)



stage 2 Only (hyp)

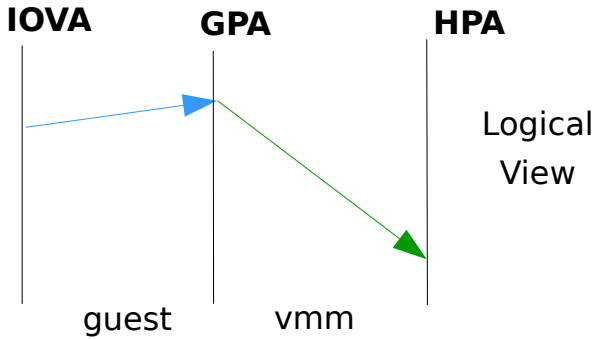


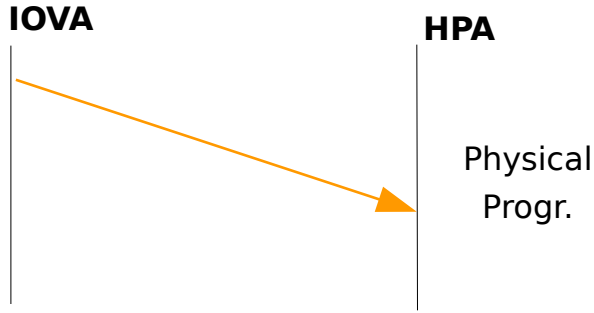
Nested Mode

Implementations of vIOMMU for assigned Device

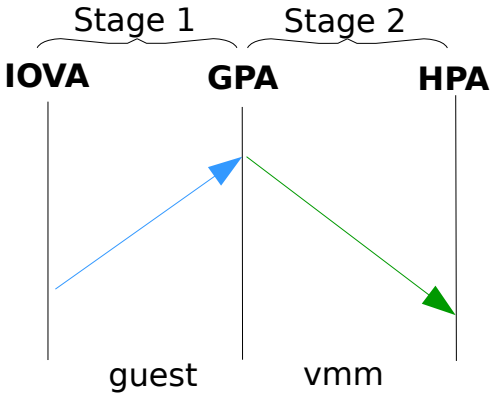
Single Physical stage Implementation

- VMM uses a single stage at physical level to implement both logical stages
- Trap on each guest config/translation structure updates
  - On each map notification, VMM SW translates the IOVA into GPA (vIOMMU emulation code)
  - Huge penalty if dynamic mappings
- On each cache invalidation, the IOVA->HPA mapping is invalidated
- This implementation is not possible with vSMMUv3!
  - we cannot trap on “map”

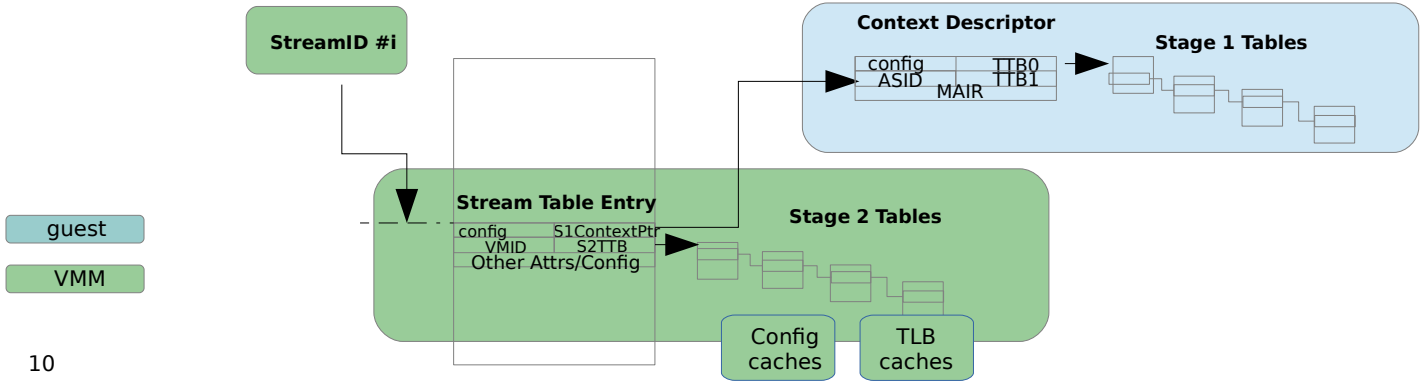




HW Nested Paging Implementation



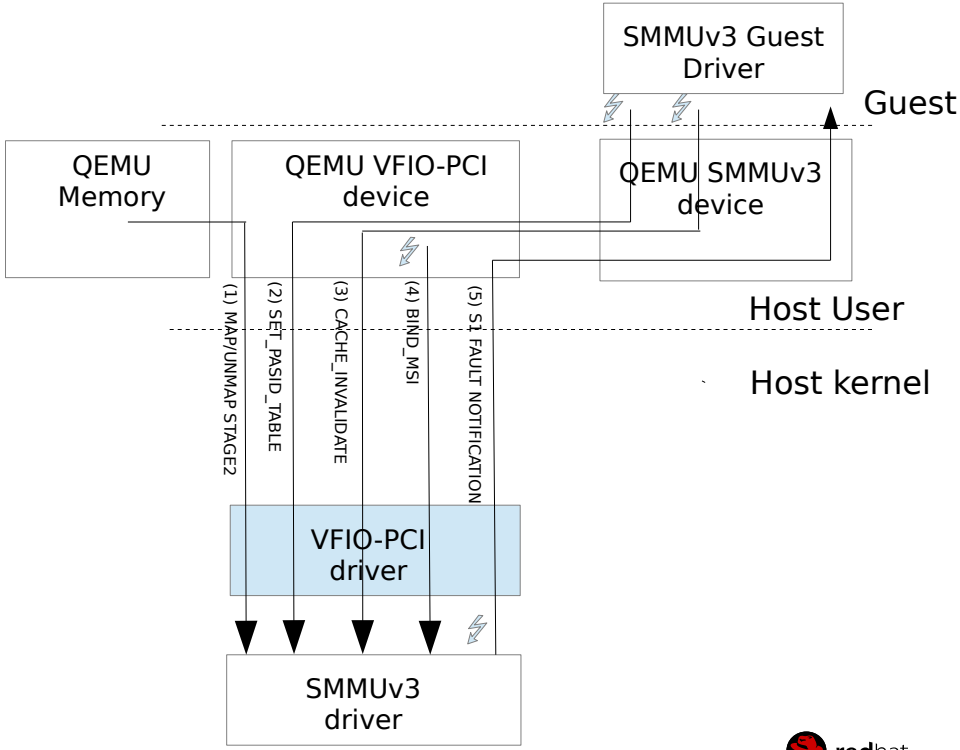
- VMM owns the main configuration structure for the streamid, including the stage2 config structure and stage2 tables
- The guest owns the stage1 config structure and stage1 tables
- No need to trap on map (\o/)
- Need to trap on guest config/IOTLB invalidations
- HW performs both translations (Need for HW fault escalation)
- Guest IOMMU driver works without any specific mode



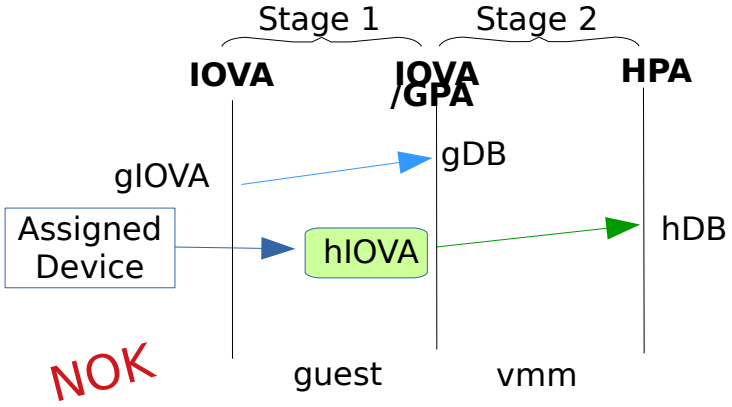
Linux VFIO/QEMU SMMUv3 HW Nested Paging Enablement

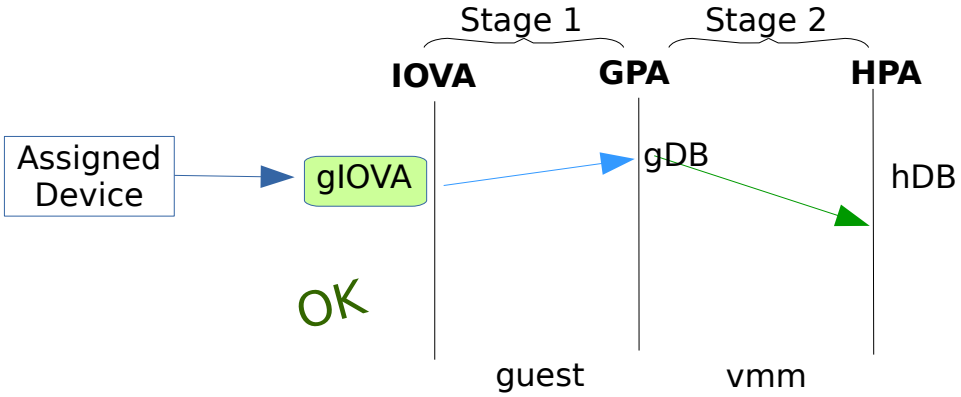
Nested Stage Control Flow

1	A Guest RAM region is added	Build stage2 mapping. Force HW stage2 to be used.
2	Guest config invalidation commands	Propagate stage 1 guest config to the host
3	Guest sends TLB/PASID cache invalidation commans	Propagate invalidations to Host
4	MSI Enable	Propagate stage1 MSI binding from guest to host
5	Stage 1 related fault	Propagate stage 1 faults from host to guest



MSI/ARM Brain Teaser





Status & Remaining Work

- Tested on Qualcomm Centriq QDF2400 and Cavium ThunderXv2
- RFC at both kernel/QEMU levels
  - Convergence on Linux IOMMU and VFIO kernel API
- Sharing/extending the integration for virtio-iommu, vt-d 3.0, SVM
- Performance needs to be assessed compared to Caching Mode technique
  - A lot of stage 2 page table walks can be induced
  - Performance may really depend on cache structures

References / Credits

- Device Assignment with Nested Guest and DPDK, Peter Xu, KVM Forum 2018
- Shared Virtual Address in KVM, Yi Liu, Jacob Pan, KVM forum 2018

- Kernel Series: [RFC v3 00/21] SMMUv3 Nested Stage Setup
  - https://github.com/eauger/linux/tree/v5.0-rc1-2stage-rfc-v3
- QEMU Series: [RFC v2 00/28] vSMMUv3/pSMMUv3 2 stage VFIO integration
  - v2++: https://github.com/eauger/qemu/commits/v3.1.0-rc5-2stage-v3-for-rfc3-test-only

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHat

youtube.com/user/RedHatVideos