Generic SR-IOV Device Support for Kernel Live Upgrade in Cloud Environment

jason.zeng@intel.com

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.

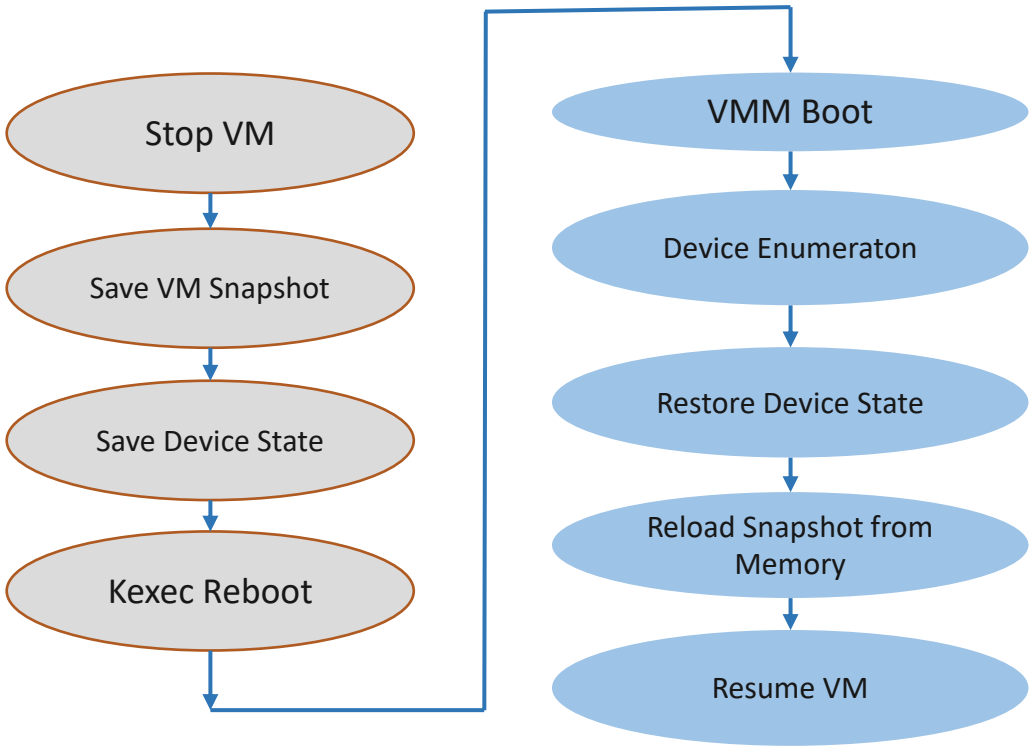
Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

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

Agenda

- Kernel Live Upgrade Recap
- SR-IOV Device Support
- Implementation

Kernel Live Upgrade Recap

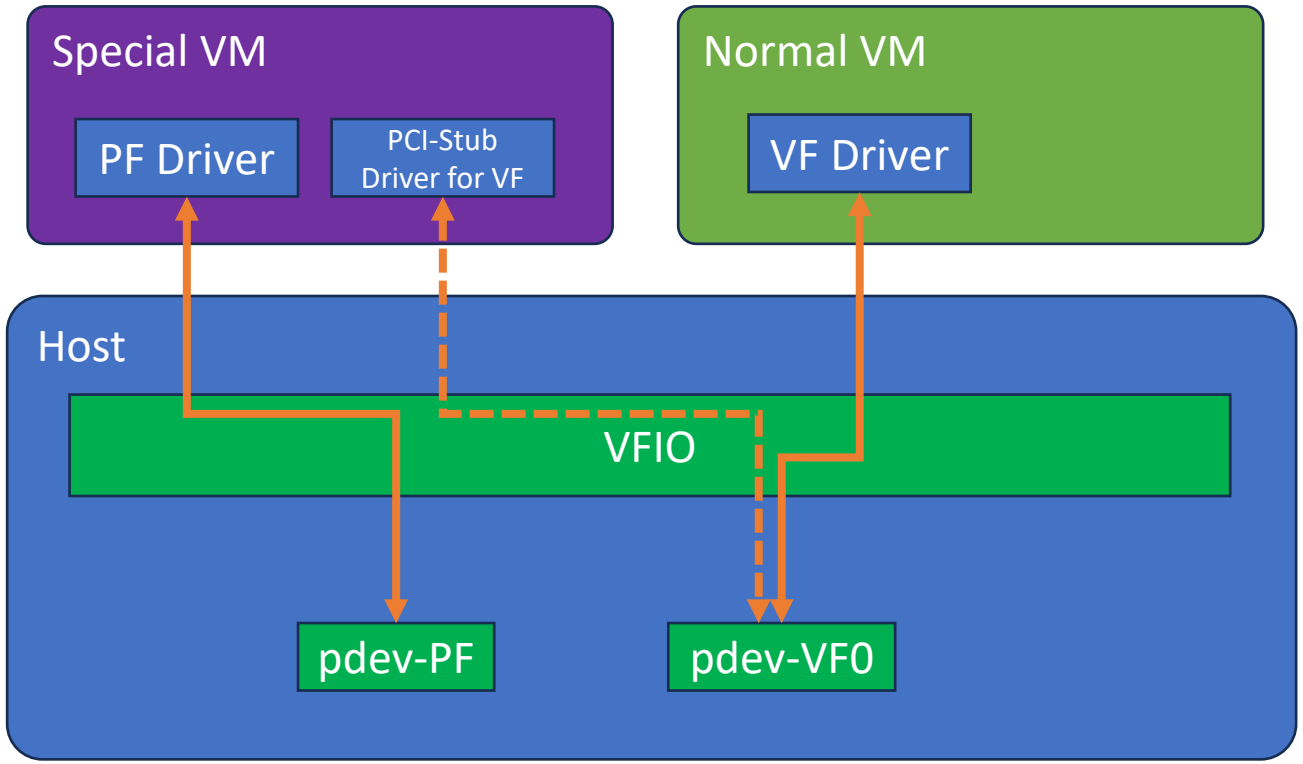


- Passthrough Device Support
  - Keep upstream devices alive
    - IOMMU
    - PCIe switch/root-port
  - Keep Device HW alive
    - DMA/Interrupt
    - Other HW states

SR-IOV Device Support – The problem

- SR-IOV PF usually managed by host
- PF HW state need to be preserved across kernel upgrade
- There are many PF vendors
  - Huge effort of vendor specific PF driver code change to preserve PF HW state

The Generic Way – Put PF in a Special VM

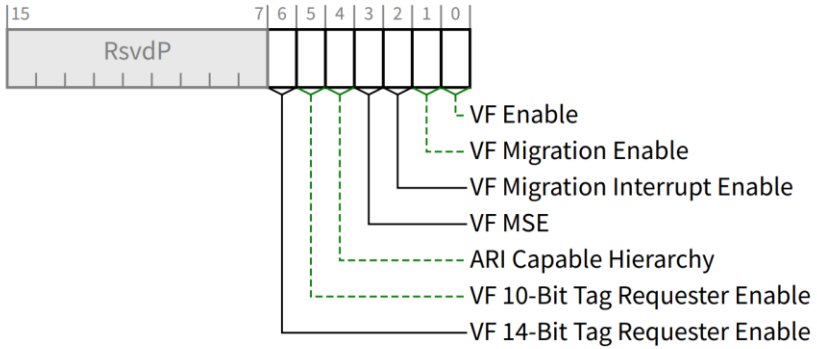


- Passthrough PF to Special VM
- When kernel live upgrade happens
  - PF HW states preserved together with the Special VM

Steps:

1. From Host
   - Passthrough PF to Special VM
2. From Special VM
   - Create VFs
   - Bind VF driver to pci-stub driver
3. From Host
   - Assign VF to Normal VM
4. From Normal VM
   - Bind VF driver to vendor's VF driver

Implementation - Passthrough SR-IOV PF

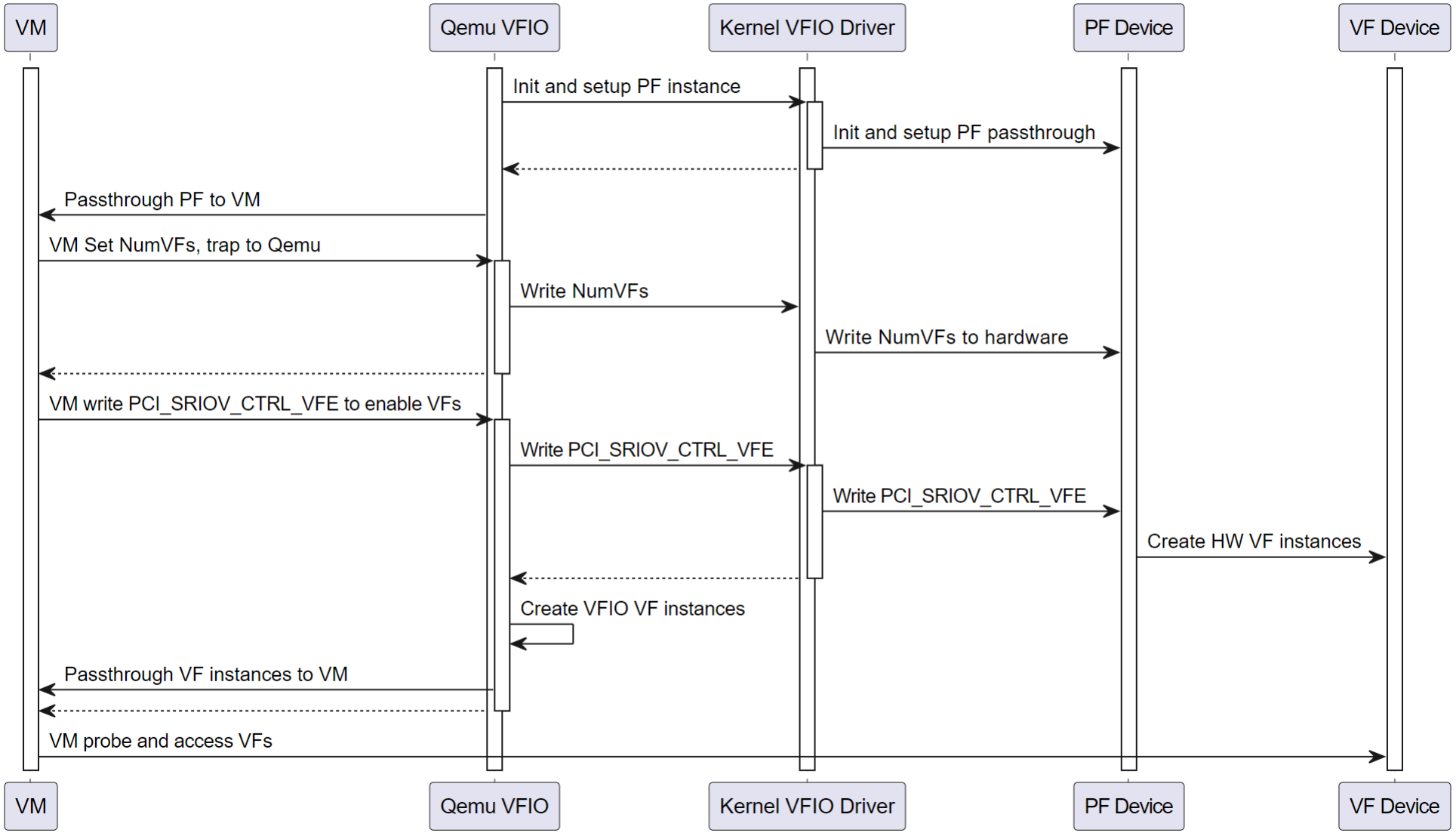


§ *Figure 9-13 SR-IOV Control Register*

- Kernel VFIO driver
  - Intercept NumVFs
  - Intercept VF_Enable & VF_MSE in Control Register
  - SR-IOV VF BAR support
    - VFIO device region
- Qemu
  - Enable SR-IOV capability handling in VFIO driver
  - SR-IOV VF BAR support
  - VF creation/destroy upon VM SR-IOV cap read/write

Next Capability Offset	Capabilities Version	PCI Express Extended Capability ID
SR-IOV Capabilities		
SR-IOV Status		SR-IOV Control
TotalVFs(RO)		InitialVFs(RO)
RsvdP	Function Dependency Link (RO)	NumVFs(RW)
VF Stride(RO)		First VF Offset(RO)
VF Device ID		RsvdP
Supported Page Size(RO)		
System Page Size(RW)		
VF BAR0 (RW)		
VF BAR1 (RW)		
VF BAR2 (RW)		
VF BAR3 (RW)		
VF BAR4 (RW)		
VF BAR5 (RW)		
VF Migration State Array Offset (RO)		

Flow: Create VFs from VM



Status & Plan

- PF Passthrough POC
  - https://gitee.com/x56Jason/vmmfr-linux/tree/pfpt/
  - https://gitee.com/x56Jason/vmmfr-qemu/tree/pfpt/

- Next
  - ARI capability support?
  - Target openEuler-22.03-LTS-SP3?