KVM FORUM

Device Keepalive State

For Local Live Migration and VMM Fast Restart

Jason Zeng jason.zeng@intel.com

Oct. 30 2020

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

- Recap of Problem
- Overview
- Keep Alive States
- VFIO Device Ownership
- Kexec-Reboot & PCI Enumeration
- Open, Status & Future Plan

Recap of Problem

- CSP’s painpoint of system update
  - CSP promises high uptime SLA to customers
  - Frequent urgent security updates, while system update usually takes long time
  - CSP sees more service downtime to customers

- Solution approaches
  - Move VM away
    - Live migration
  - Keep VM at local
    - Kernel Live Patching
    - Update components separately or in a whole (Qemu, KVM, Kernel)

Solution Spectrum & Where is VMM Fast Restart

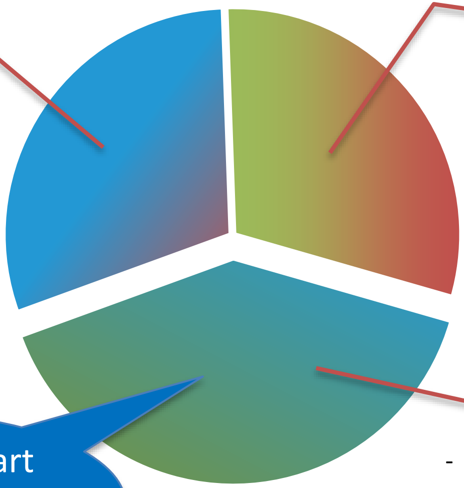
Allow pausing VM or not? Allow host reboot or not?

- Allow pausing VM

- Passthrough device can be suspended in VM
- host reboot naturally
- Optimization: put guest RAM in persistent memory to boost performance

- Don’t allow VM pausing
- Don’t allow host reboot

- Needs to pass device state to resumed VM
- Needs to pass guest memory mapping to resumed Qemu



VMM Fast Restart wants to solve this category

- Don’t allow pausing VM
- Allow host reboot

- Need to pass device state to new kernel
- Can leverage kexec-reboot
- A time window during which device has no owner
  - Device needs to be keepalive

Overview

- Device Keepalive State
- Two Incremental Stages
- Example Commands
- What to Preserve, What to Re-create

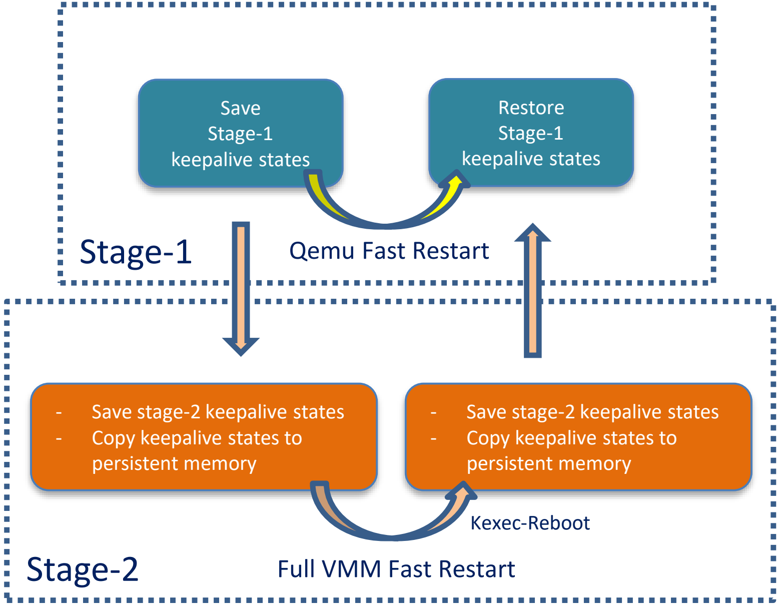
Device Keepalive State

- Device hardware is still alive (no owner)
  - May continue to perform DMA
  - May continue to issue interrupts to host
- Host software
  - Must not modify the underlying hardware state
  - Must not bind the device to any other drivers
- Device software state (created & managed by drivers)
  - Saved at one of two stages
    - Enters keepalive state, or host reboot time
  - When re-created, underlying hardware state is re-attached to this software state

```
struct device {
        struct kobject kobj;
        struct device *parent;
        ......
        bool keepalive:1;
};
```

Two Incremental Stages

- Stage-1 Keepalive States
  - Related to Qemu runtime operations
- Stage-2 Keepalive States
  - Related to Kernel configuration
- Stage-1 keepalive states alone can be used for implementing Qemu Live Update
  - An alternative solution to *file descriptor passing over exec(3)*
  - Also applicable for local live migration


Save Stage-1 keepalive states
Restore Stage-1 keepalive states
Stage-1
Qemu Fast Restart
- Save stage-2 keepalive states
- Copy keepalive states to persistent memory
- Save stage-2 keepalive states
- Copy keepalive states to persistent memory
Kexec-Reboot
Stage-2
Full VMM Fast Restart

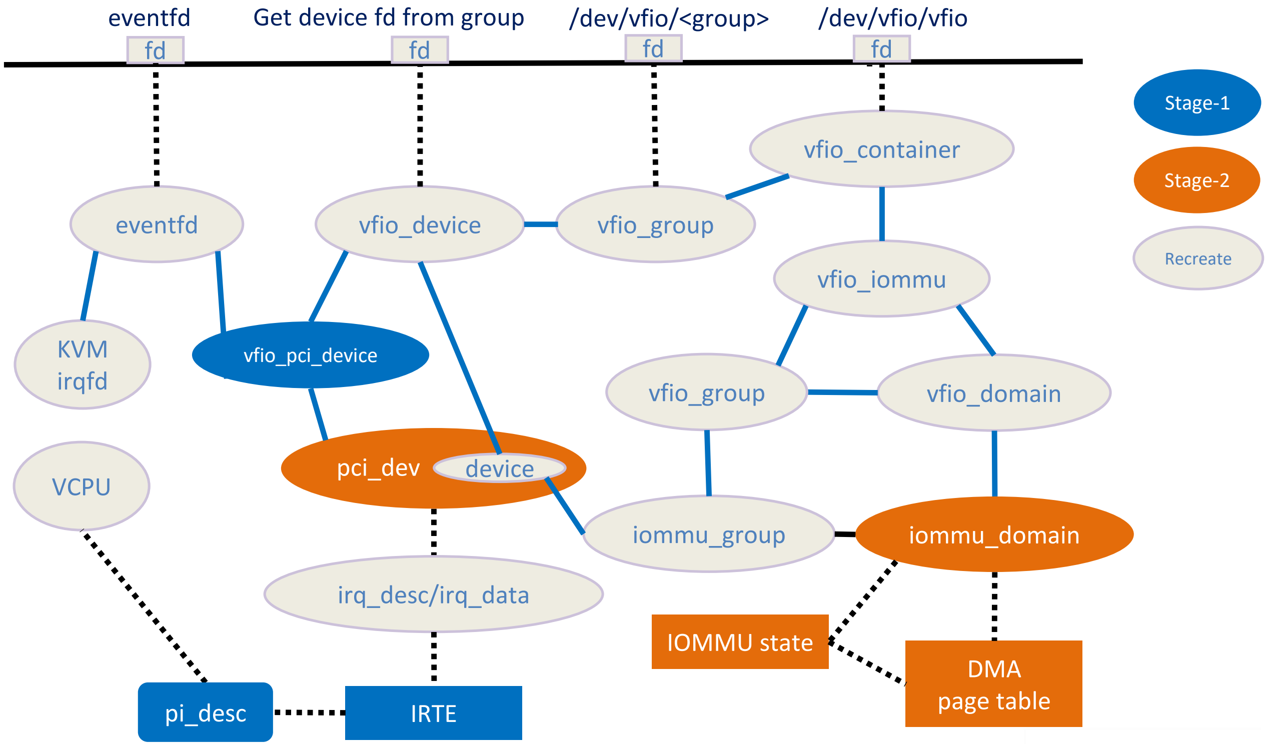
Example Commands for VMM Fast Restart

```
# qemu-system-x86_64 \
        --enable-kvm -M q35 -m 4G -smp 1 -hda ubuntu-1904.qcow2 -monitor stdio \
        -object memory-backend-file,id=dimm0,size=4g,mem-path=/dev/dax0.0,share=on,pmem=on,align=2M \
        -numa node,memdev=dimm0,cpus=0 \
        -device vfio-pci,host=81:00.0 \
QEMU 4.1.92 monitor - type 'help' for more information
(qemu) migrate_set_capability x-ignore-shared on
(qemu) stop
(qemu) set-keepalive on,token=619cdf24-226f-4418-9f5a-98346019860e
(qemu) savevm s0
(qemu) q


# qemu-system-x86_64 \
        --enable-kvm -M q35 -m 4G -smp 1 -hda ubuntu-1904.qcow2 -monitor stdio \
        -object memory-backend-file,id=dimm0,size=4g,mem-path=/dev/dax0.0,share=on,pmem=on,align=2M \
        -numa node,memdev=dimm0,cpus=0 \
        -device vfio-pci,host=81:00.0,keepalive_token=619cdf24-226f-4418-9f5a-98346019860e \
        -S
QEMU 4.1.92 monitor - type 'help' for more information
(qemu) migrate_set_capability x-ignore-shared on
(qemu) loadvm s0
(qemu) set-keepalive off
(qemu) c
```

What to Preserve, What to Re-create

- Rationale
  - Hardware dependent or not
  - Try to be less intrusive to other components
  - Stage-1 or Stage-2?
    - Qemu runtime operation related?
    - Kernel configuration related?

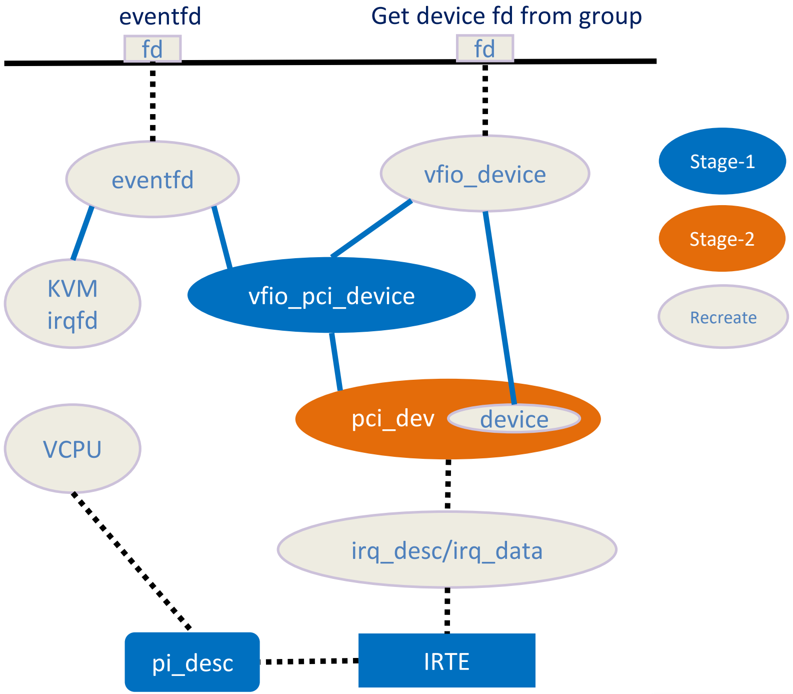


Keep Alive States

- Keep IRQ Alive
  - Preserve/Restore pi_desc
  - Preserve/Restore IRTE
- Keep DMA Alive

Keep IRQ Alive

- Challenges
  - Hardware not available to response
    - CPU is rebooting
  - Software not available to handle irq
- 2 Options
  - Mask IRQ before restart
    - Some devices don't support MSI masking
  - Leverage Posted Interrupt (chosen here)
    - Not depend on MSI/MSIX
- At least three items need to be preserved
  - Posted Interrupt Descriptor (pi_desc, PID)
  - IRTE
  - Device interrupt vector usage
    - Managed by Qemu (not shown in picture)



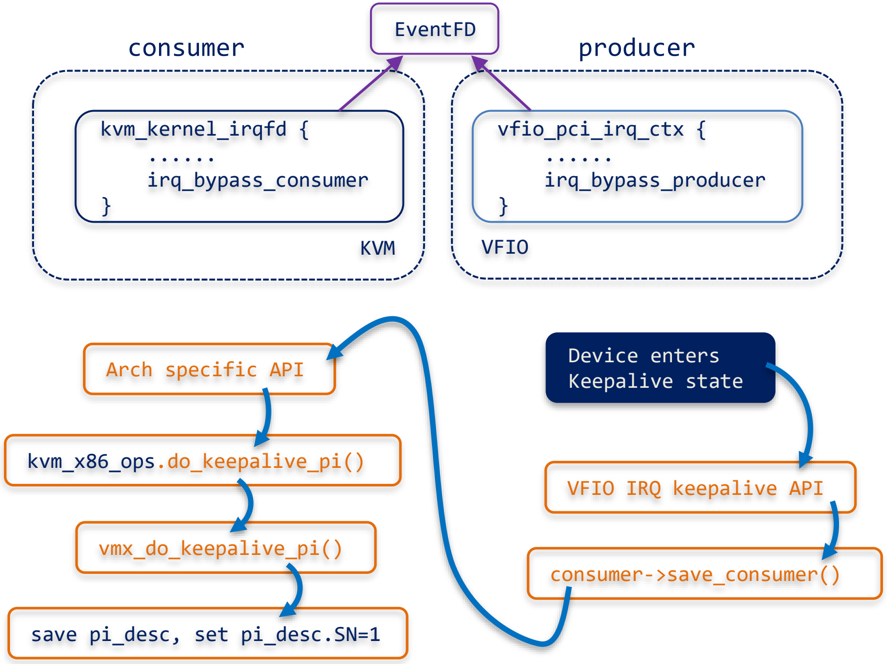
Preserve/Restore pi_desc

- 2 options
  - Introduce ioctl commands for KVM
    - UAPI change
  - Leverage irq_bypass mechanism (Chosen here)
    - Kernel internal API change

- Introduce irq_bypass_consumer{} callbacks:
  - save_consumer()
  - Restore_consumer()

- Introduce arch specific APIs:
  - kvm_arch_irq_bypass_save_consumer()
  - Kvm_arch_irq_bypass_restore_consumer()

- Introduce kvm_x86_ops{} callback
  - kvm_x86_ops.do_keepalive_pi()



EventFD

consumer

producer

```
kvm_kernel_irqfd {
    ......
    irq_bypass_consumer
}
```

KVM

```
vfio_pci_irq_ctx {
    ......
    irq_bypass_producer
}
```

VFIO

Device enters Keepalive state

VFIO IRQ keepalive API

consumer->save_consumer()

Arch specific API

kvm_x86_ops.do_keepalive_pi()

vmx_do_keepalive_pi()

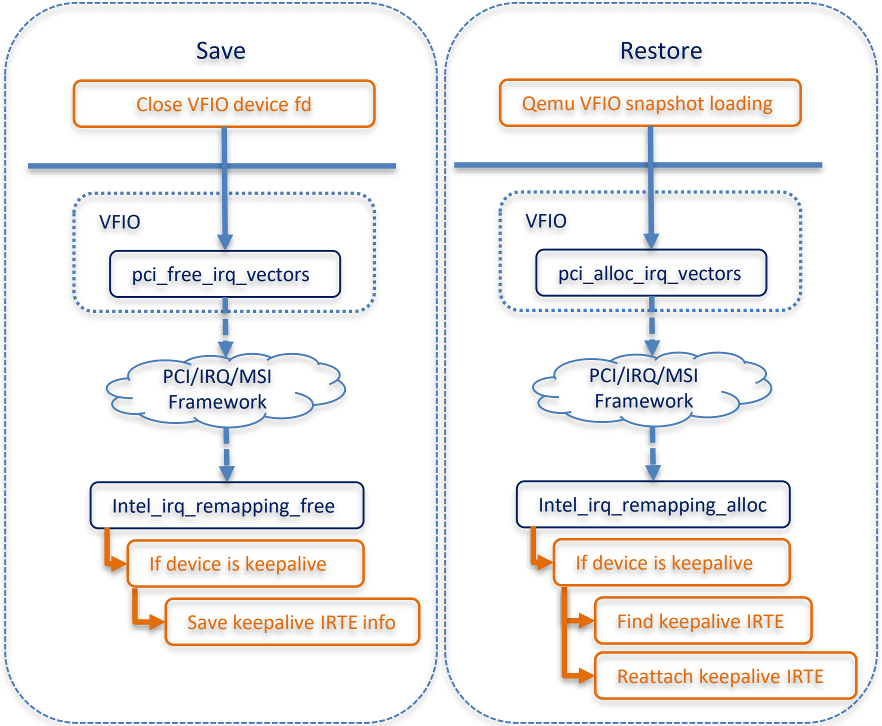
save pi_desc, set pi_desc.SN=1

Preserve/Restore IRTE

- 2 options
  - Awareness at irq_remapping driver (Chosen here)
    - Need to avoid HW clobber at PCI/MSI core
  - Awareness at PCI/IRQ/MSI core
    - Need introduce new API
    - Intrusive code change to PCI/IRQ core

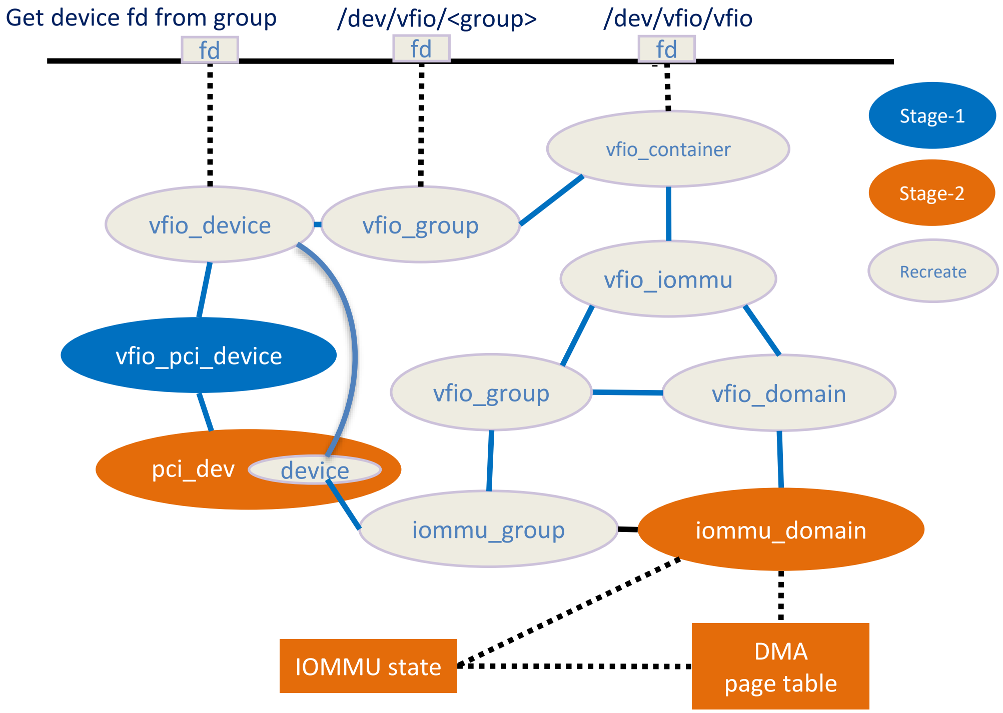
- IRQ remapping driver
  - Save/restore IRTE
  - Record mapping between IRTE and <bdf, vector index>

- PCI/MSI core
  - Don't write MSI/MSIX registers in PCI MSI/MSIX code path if device is keepalive



Keep DMA Alive

- What need to preserve?
  - DMA page table, VM Domain id, Pasid (not covered), IOMMU configuration, etc.
- Preserve iommu_domain or not?
  - Preserve: most code change in VFIO (chosen here)
  - Not Preserve: much code change in IOMMU
- Open: vDPA support?



VFIO Device Ownership

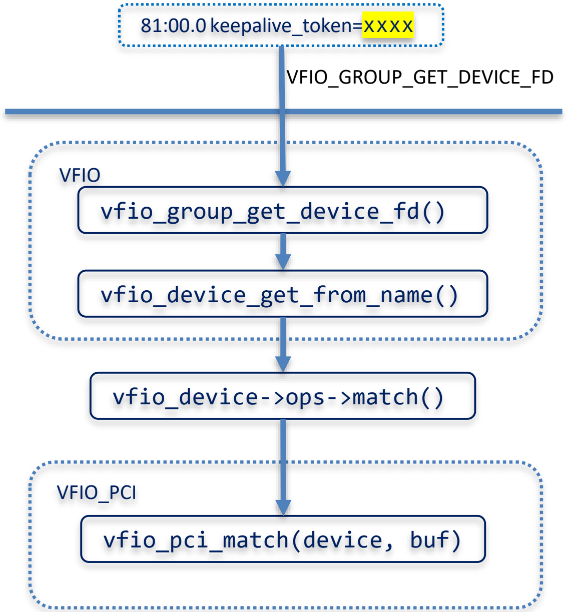
- No owner when device is in keepalive state
- /sys interface to allow admin to grant cap to user/process
- Need an authentication mechanism to verify ownership
  - Only previous owner has the permission to inherit the device
- Introduce a uuid token
  - Set the token to the VFIO device when keepalive operation starts
  - Validate when VM resumes and opens the vfio device

```
......

(qemu) set-keepalive on,token=619cdf24-226f-4418-9f5a-98346019860e

......

# qemu-system-x86_64 \
   ......
   -device vfio-pci,host=81:00.0,keepalive_token=619cdf24-226f-4418-9f5a-98346019860e \
   ......
```



81:00.0 keepalive_token=XXXX

VFIO_GROUP_GET_DEVICE_FD

VFIO

vfio_group_get_device_fd()

vfio_device_get_from_name()

vfio_device->ops->match()

VFIO_PCI

vfio_pci_match(device, buf)

Kexec-Reboot

- Introduce keepalive callback notifier for kexec-reboot
  - Save Stage-2 keepalive states
    - PCI core, IOMMU etc. keepalive states
  - Save passthrough device list
  - Copy keepalive states to persistent memory

- Restore keepalive state after kexec-Reboot
  - Restore IOMMU state early
  - Customize PCI enumeration process to restore PCI device state

- Need a memory handover mechanism cross kexec-Reboot
  - https://lore.kernel.org/lkml/1588812129-8596-1-git-send-email-anthony.yznaga@oracle.com/

PCI Enumeration

- Special handling at scan phase if device is passthrough
  - Don’t write HW registers
  - Restore state from data passed from old kernel
  - Skip firmware loading

- Resource Assignment
  - Restore BAR resource assignment from HW
    - Allocated by old kernel, should have no conflict (fail if conflict)

Opens

- We avoid writing MSI/MSIX registers of keepalive device on IRQ teardown and setup code path
  - Do we need to check keepalive flag in all other PCI code path? How intrusive to PCI core?

- What about PCI enumeration failure after kexec-reboot?
  - How to notify Qemu about PCI resource conflict?

Opens – Cont'd

- Port service drivers of switch/root port
  - Propagate keepalive flag to switch/root port?
  - AER, BW_notification, DPC, PCIeHP, etc. capabilities
  - Most of them register IRQs, how to handle these IRQs?
    - Disable & read back status registers after kexec-reboot?

- SRIOV/SIOV support
  - PF device state also need to be preserved
    - Propagate keepalive flag to PF?
    - PF vendor driver also need to change?
  - PCI enumeration handling
    - Avoid destroy VF configuration

Status & Future Plan

- POC of Qemu Fast Restart & full VMM Fast Restart - done
  - Haswell/Broadwell platform, Intel x540 NIC
  - Workloads can be restored after full VMM fast restart
    - Youtube video streaming workloads
    - SCP workloads
- Github repo:
  - https://github.com/intel/vmm-fast-restart-linux
  - https://github.com/intel/vmm-fast-restart-qemu
- Future plan: Upstreaming
  - POC proved feasible, but challenging for upstreaming
- Welcome comments, suggestions, and cooperation !

Question?

KVM FORUM

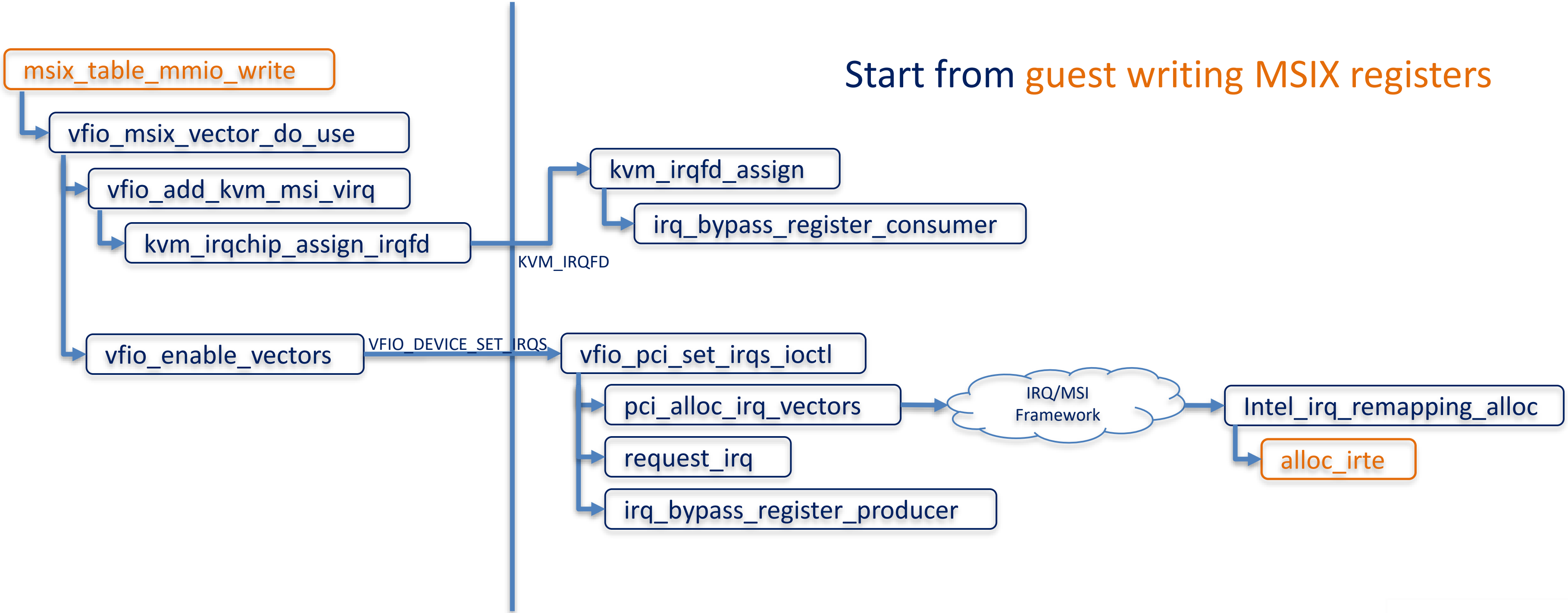
intel

Backup

Keep IRQ Alive

Normal VFIO Flow for IRQ Configuration

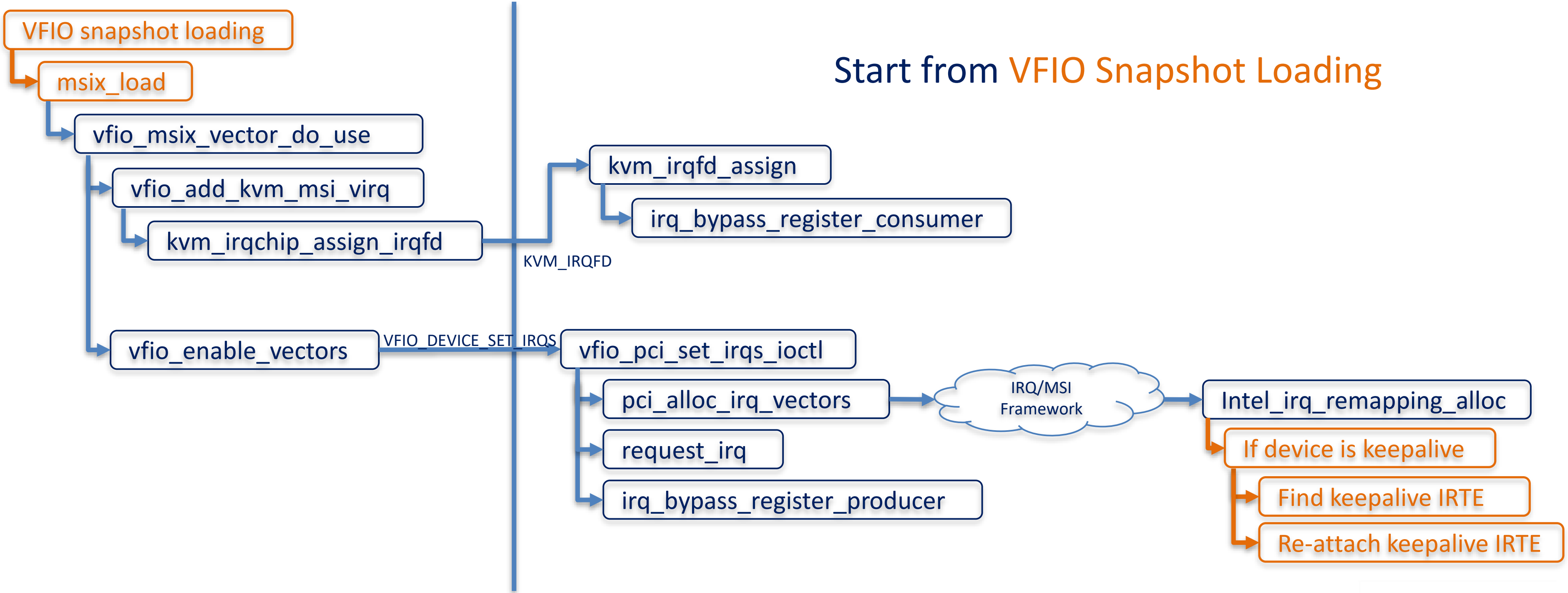
Start from guest writing MSIX registers



Keep IRQ Alive

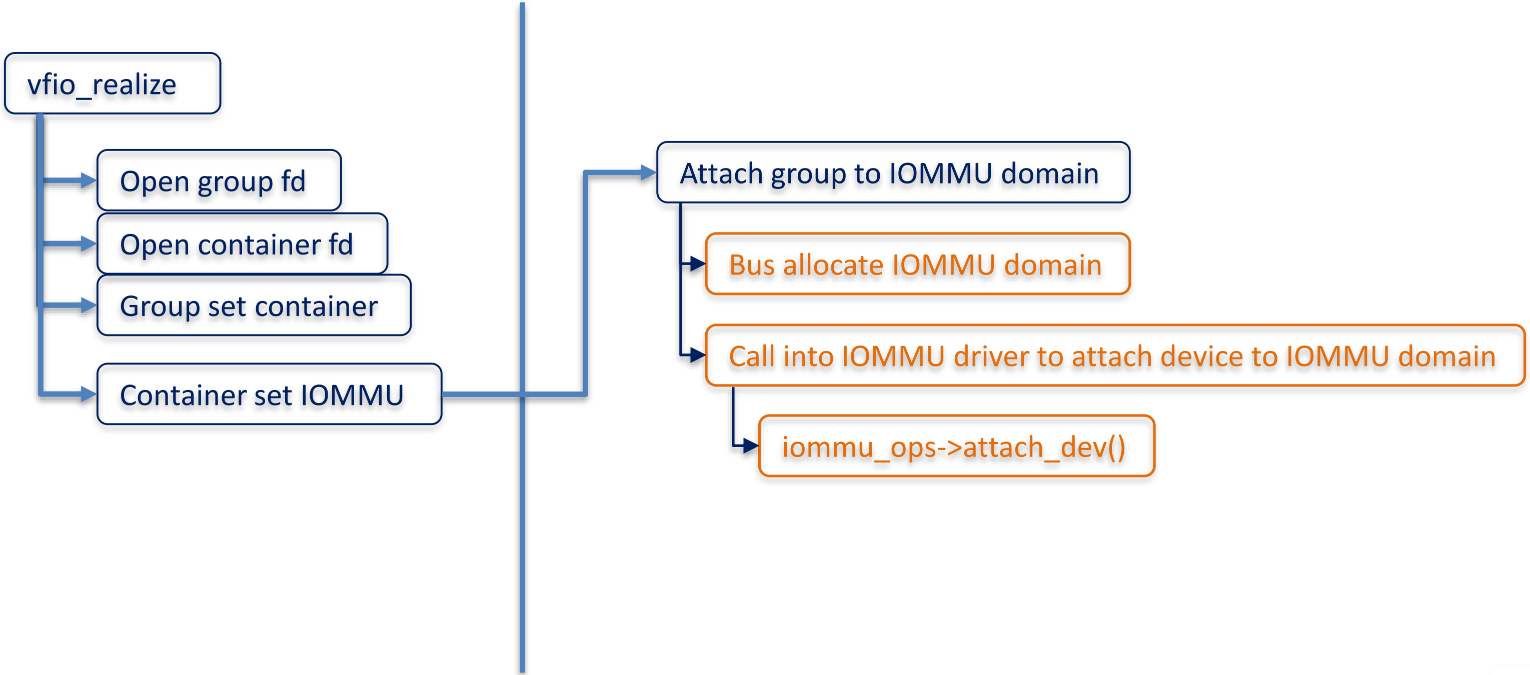
Keepalive VFIO Flow for IRQ Configuration and IRTE Restoration

Start from VFIO Snapshot Loading



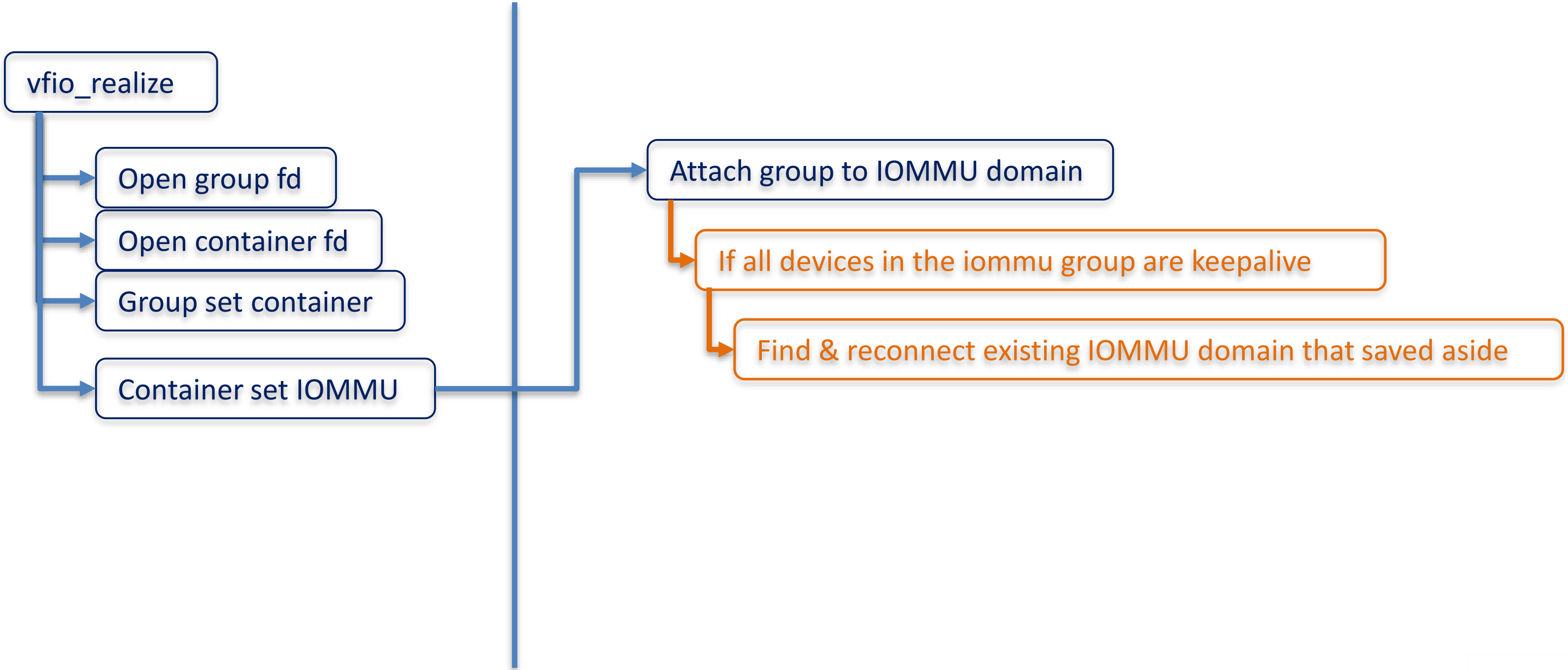
Keep DMA Alive

Normal VFIO Flow for IOMMU Domain Configuration


vfio_realize
Open group fd
Open container fd
Group set container
Container set IOMMU
Attach group to IOMMU domain
Bus allocate IOMMU domain
Call into IOMMU driver to attach device to IOMMU domain
iommu_ops->attach_dev()

Keep DMA Alive

Keepalive VFIO Flow for IOMMU Domain Configuration



vfio_realize

Open group fd

Open container fd

Group set container

Container set IOMMU

Attach group to IOMMU domain

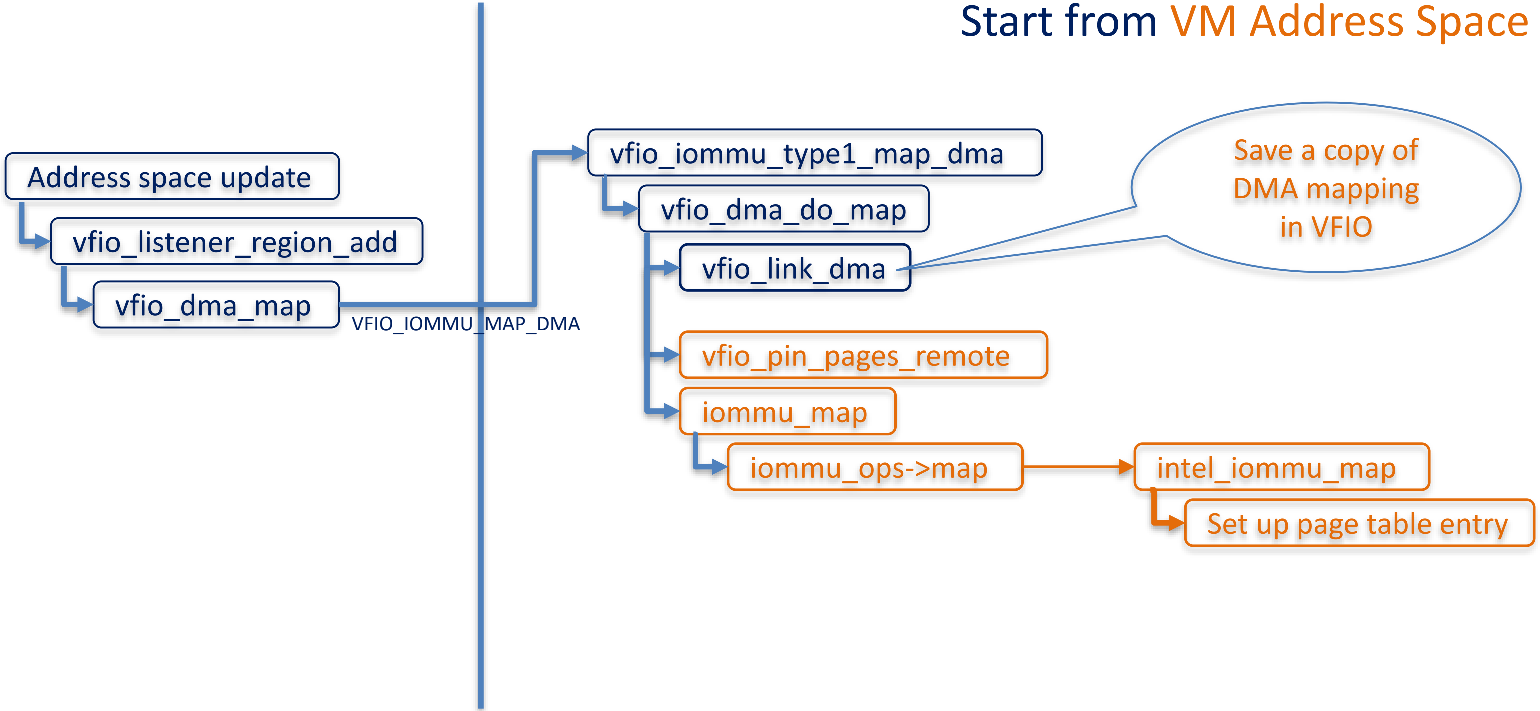
If all devices in the iommu group are keepalive

Find & reconnect existing IOMMU domain that saved aside

Keep DMA Alive

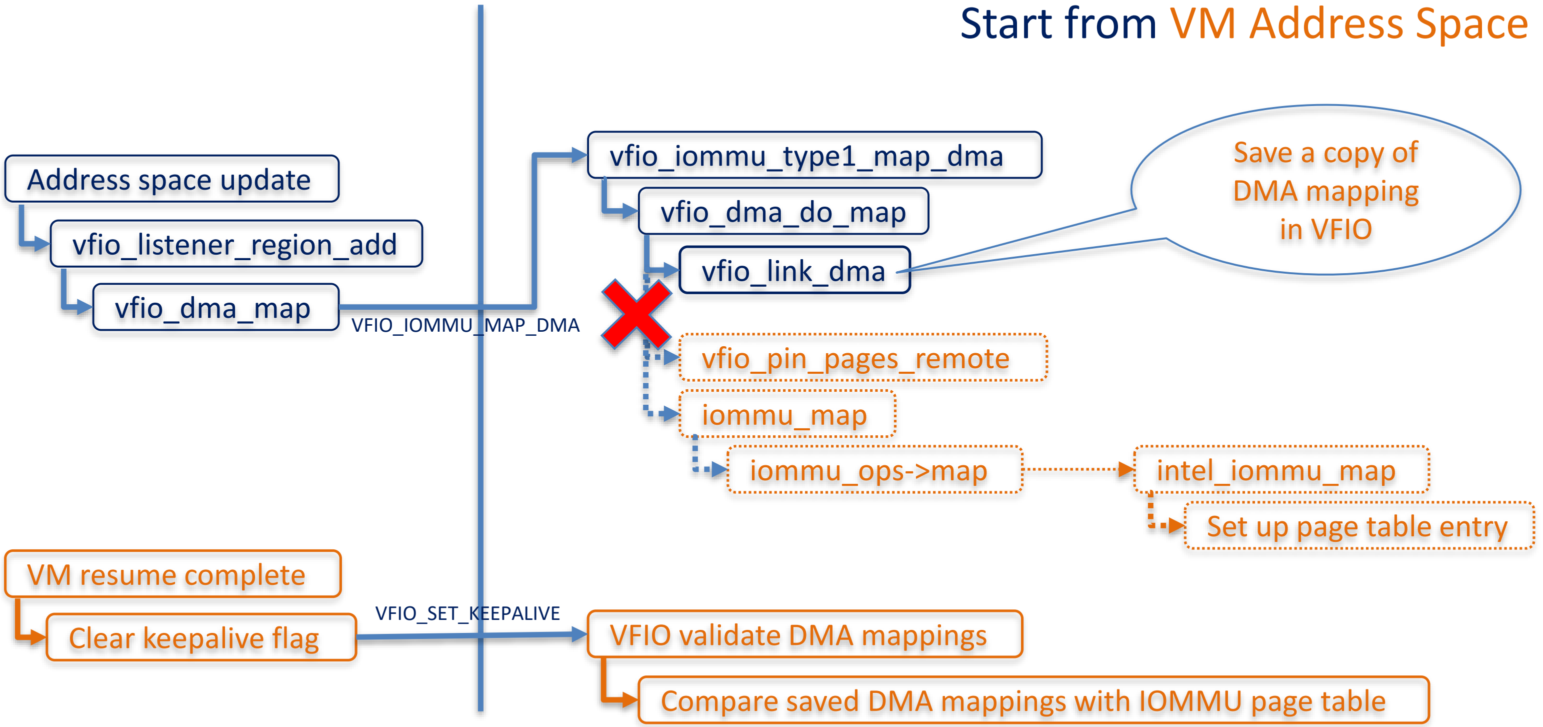
Normal VFIO Flow for DMA Mapping

Start from VM Address Space Update



Keep DMA Alive

Keepalive VFIO Flow for DMA Mapping


Start from VM Address Space Update
Address space update
vfio_listener_region_add
vfio_dma_map
VFIO_IOMMU_MAP_DMA
vfio_iommu_type1_map_dma
vfio_dma_do_map
vfio_link_dma
Save a copy of DMA mapping in VFIO
vfio_pin_pages_remote
iommu_map
iommu_ops->map
intel_iommu_map
Set up page table entry
VM resume complete
Clear keepalive flag
VFIO_SET_KEEPALIVE
VFIO validate DMA mappings
Compare saved DMA mappings with IOMMU page table