Post-Copy Live Migration on Pass-through Devices

Kevin Tian (on behalf of Yan Zhao), Shaopeng He
Intel Corporation

Legal Disclaimer

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.

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

Live Migration

- Lively migrate the VM state between server nodes
  - CPU state
  - Memory state
  - Device state
  - VM configuration

- Cornerstone capability in data centers and clouds
  - Load balancing, SLA, infrastructure maintenance, etc.

Memory State: Pre-Copy vs. Post-Copy

Pre-Copy

1. Copy the entire guest memory to dest

2. Iteratively copy pages dirtied in previous round

Stop VM on src

3. Copy CPU/device state and remaining dirty pages to dest

Resume VM on dest

Mature and Reliable

Post-Copy

Stop VM on src

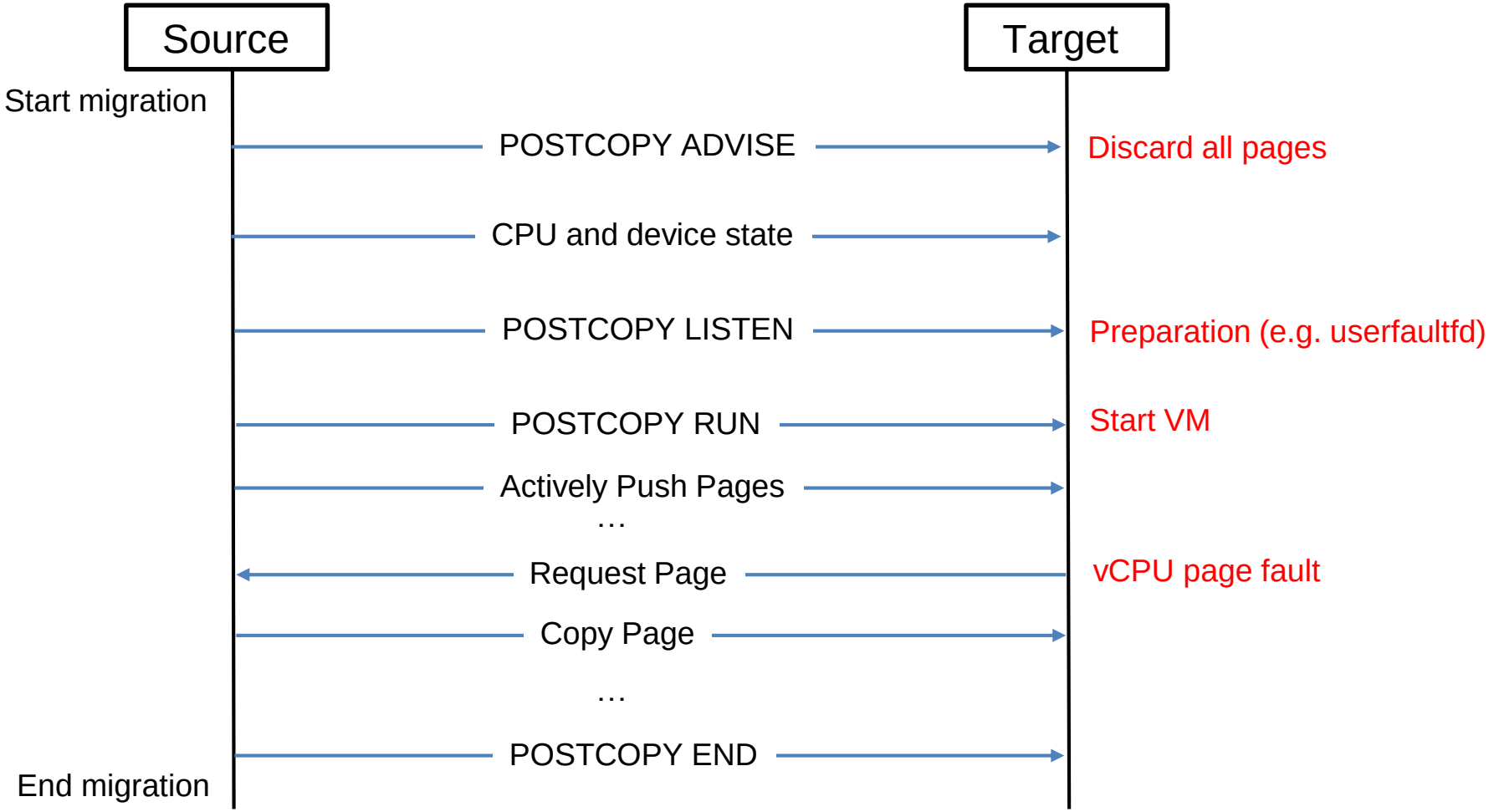
1. Copy CPU/device state to dest

Resume VM on dest

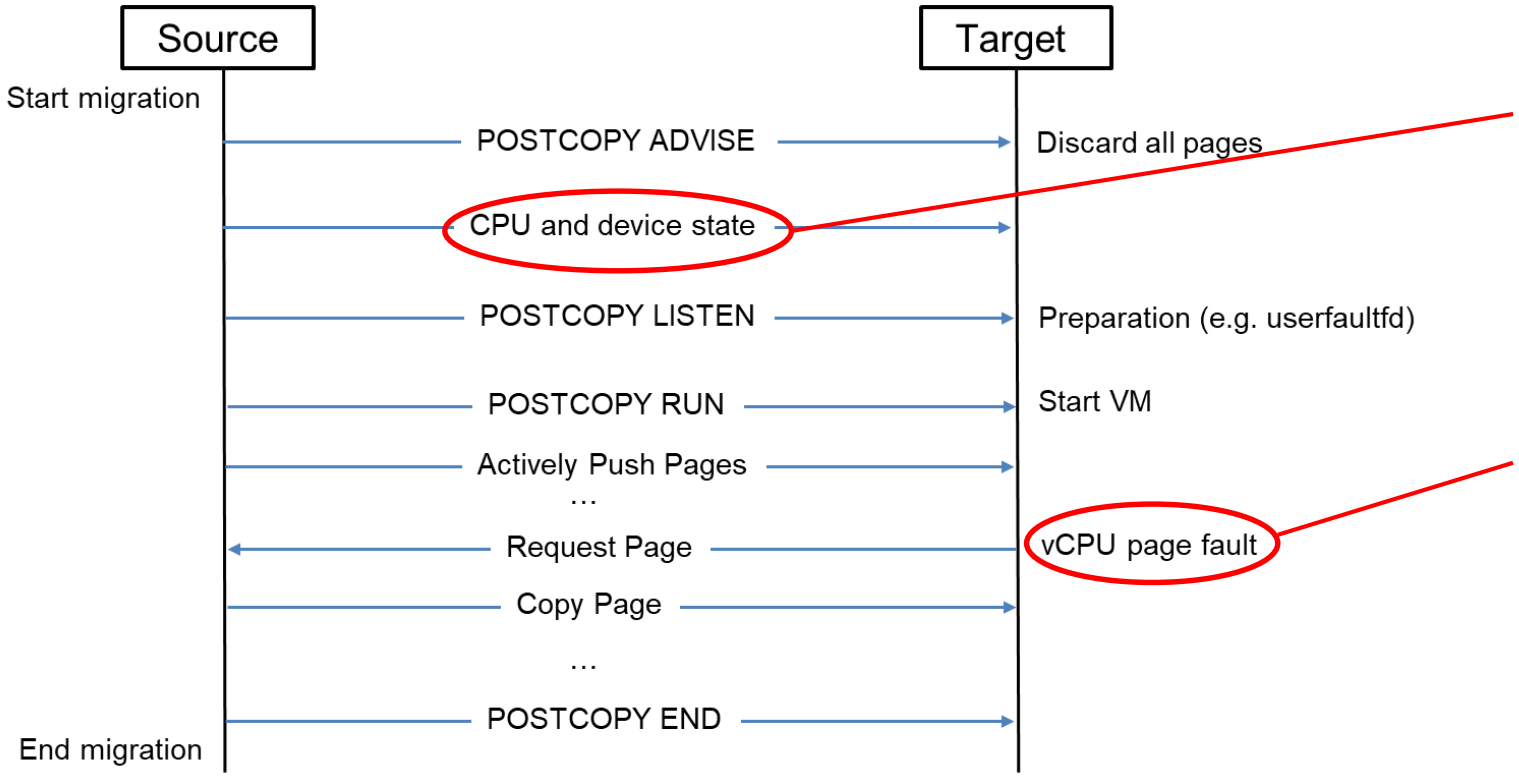
2. Copy the entire guest memory

Predictable and faster

A Detailed Flow of Post-Copy Live Migration



Challenges with Passthrough Devices



Host has no insight of the device state

Most devices don’t support DMA fault

Device State Migration

- Vendor specific definition of ‘device state’
  - Standard PCI resource (config space, MMIO BAR, MSI/MSI-X, etc.)
  - Vendor specific resource (e.g. hidden state, PF-maintained configuration, etc.)

- Use a wrapper driver for poking the device state
  - Use vfio-mdev, by creating a single mdev on top of the device (example)
  - Use vfio-pci, by hooking to specific vfio device ops (under discussion)
  - Getting/setting device state through the migration region (v8)

DMA Page Prefault

- Bear the fact that most devices don’t support DMA fault
- Prefault through device specific mediation
  - Trap any guest operation which may cause DMA access
  - Decode the operation to find out pages that may be used for DMA
  - Prefault the pages to userspace for memory pulling
- Device specific mediation policies
  - E.g. trap guest MMIO accesses, scan workload descriptors, etc.

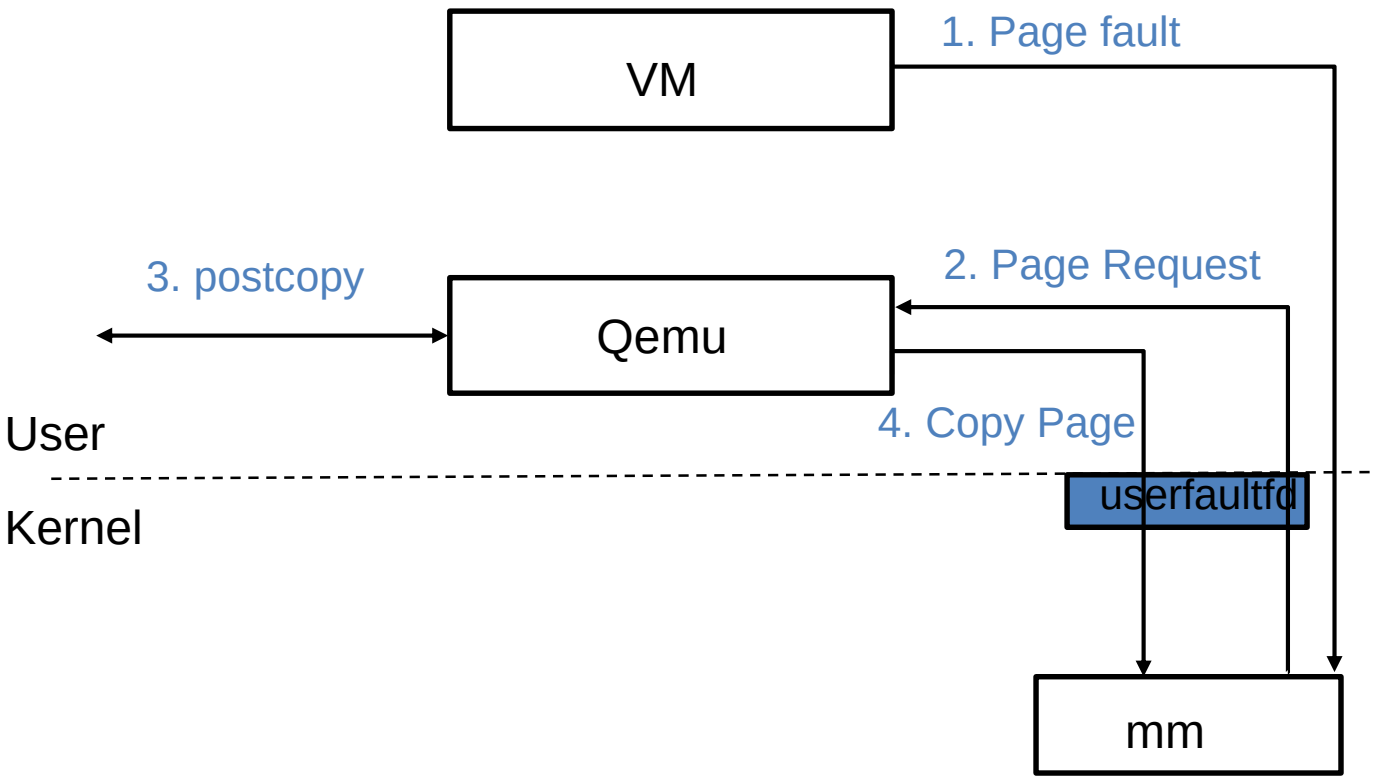
Need the capability of dynamic mediation!

Dynamic Mediation

- Dynamically turn on mediation when migration starts
  - For regions marked as VFIO_REGION_INFO_FLAG_MMAP
  - Re-enable pass-through when migration ends

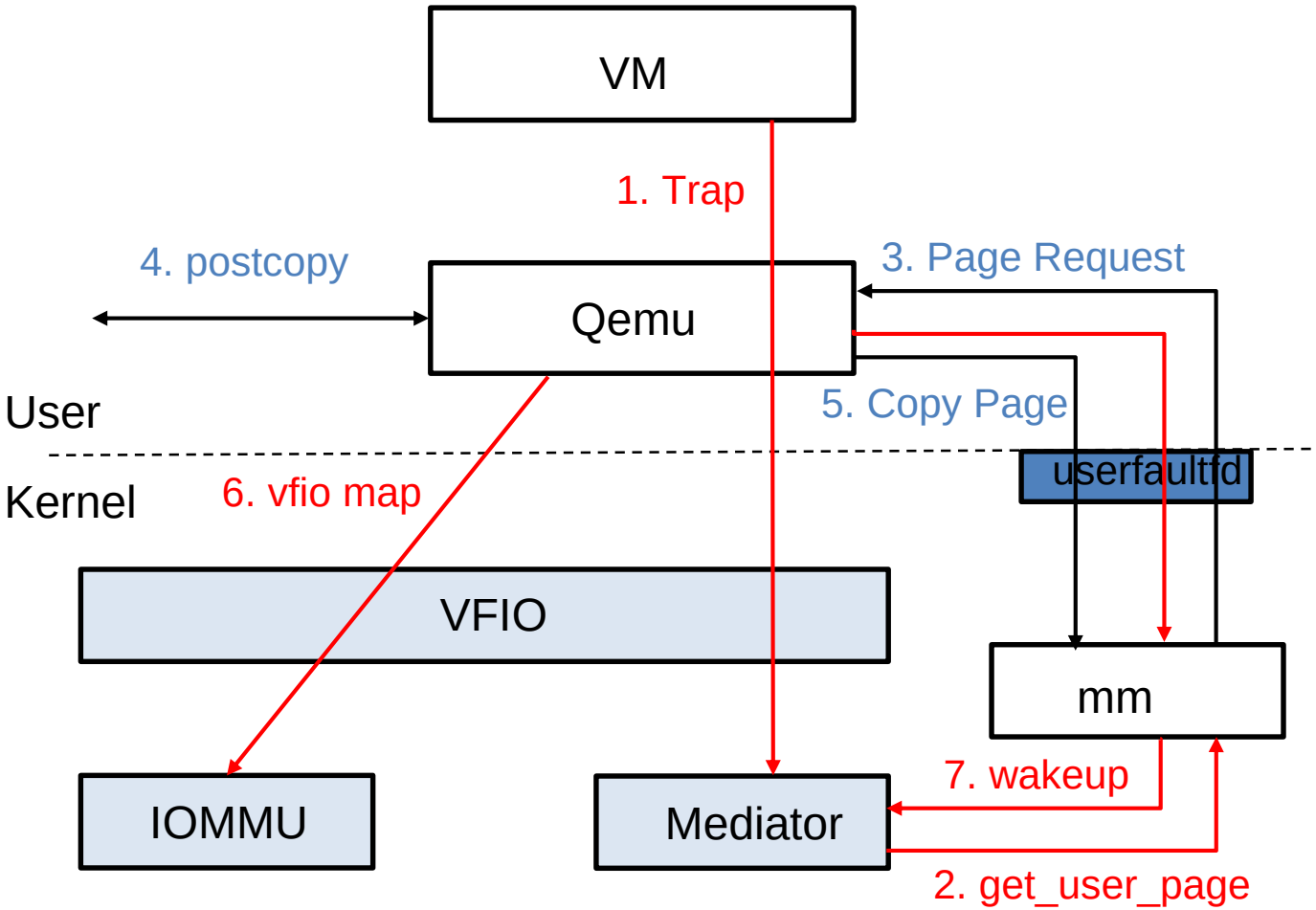
- Introduce a mediation bitmask in migration region
  - Indicate which region should be dynamically mediated when migration starts
  - Currently defined per MMIO region
  - Future
    - Implement a sparse structure for finer-grained mediation control
    - Event-based notification triggered by mediator

Fault-and-Pull



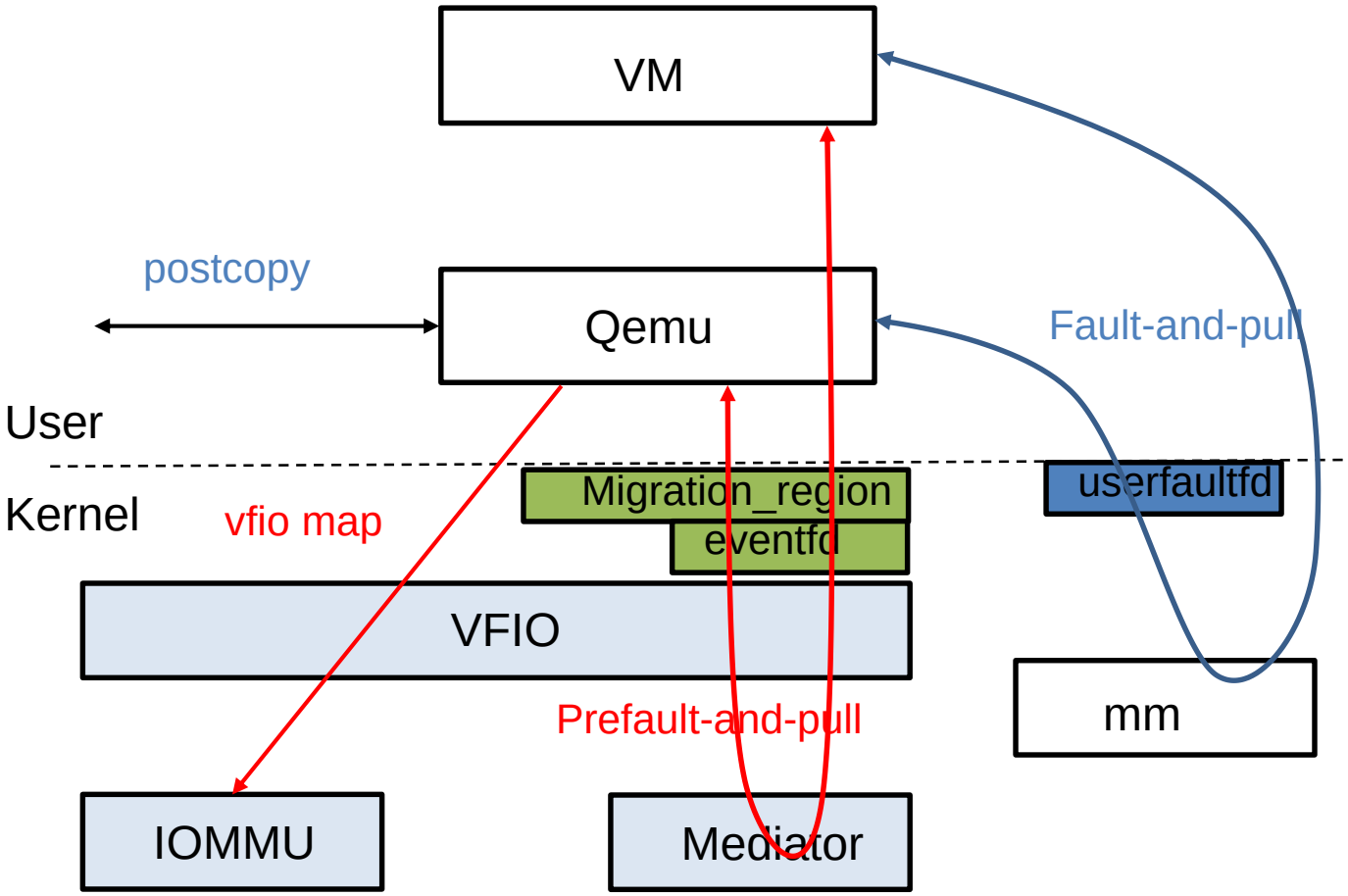
- Based on userfaultfd
  - Introduced in 2015
- For handling vCPU page fault in userspace
- UFFDIO_COPY (step 4)
  - Allocate/copy page and wake up vCPU

Prefault-and-Pull: Ideal Approach



- Leverage userfaultfd interface
  - Triggered by mediator
- Separate page copy and vCPU wakeup
  - UFFDIO_COPY (DONTWAKE) + UFFDIO_WAKE
  - Allow VFIO map in the middle
- Limitations
  - Challenges on finding IOVA->HVA for GUP
    - VFIO mappings have been discarded
    - KVM may provide GPA->HVA (w/o vIOMMU), but what about w/ vIOMMU?
  - No support of device local memory

Prefault-and-Pull: VFIO Approach

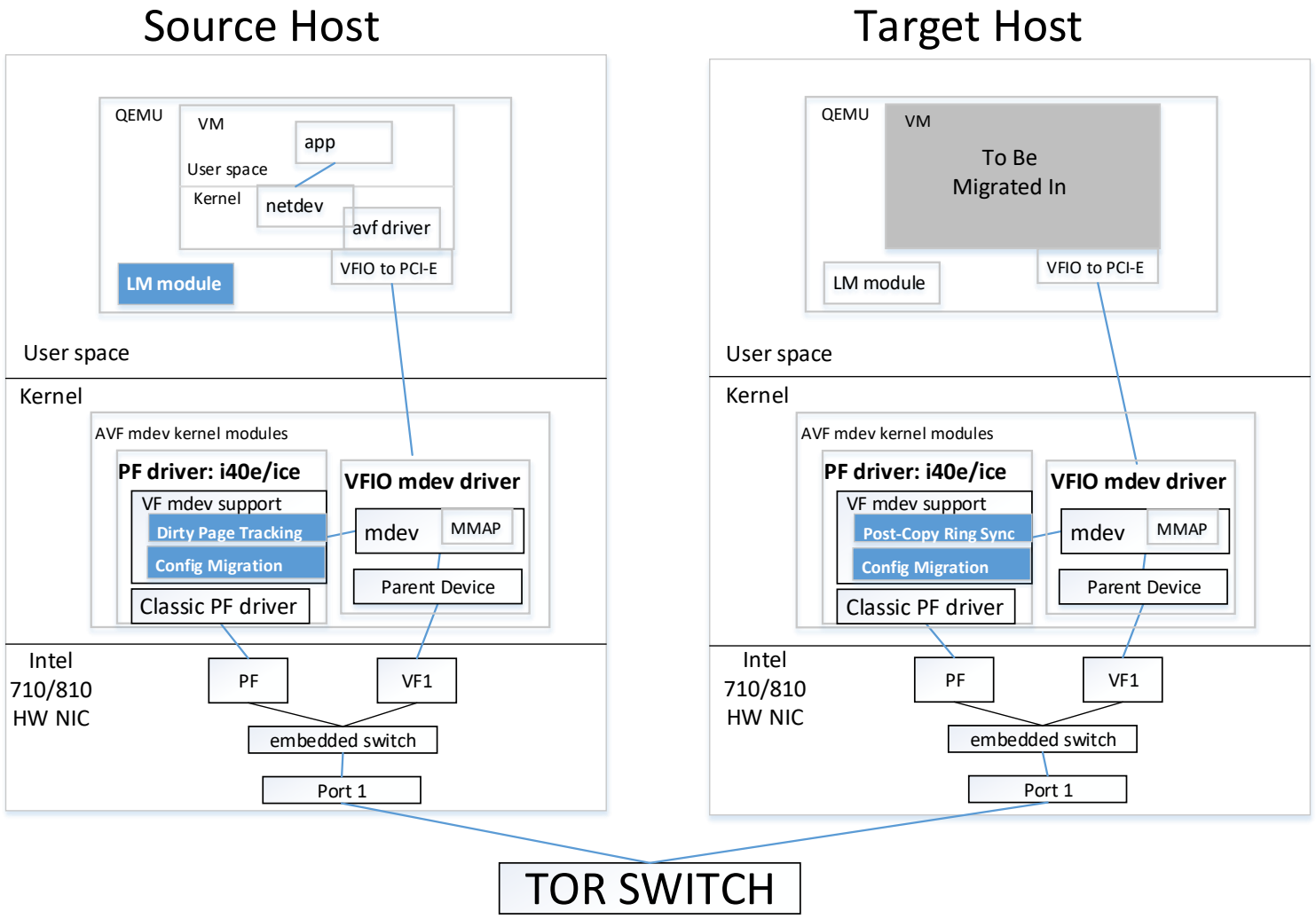


- Built on VFIO migration_region interface
  - A new eventfd for notifying userspace
  - Reuse data_region to carry prefault info
  - Qemu translates from IOVA into GPA
- CPU page fault continues on userfaultfd
- Pros
  - Specifically designed for IOVA-based prefault
  - Easily extended to cover device local memory

Case Study: NIC Passthrough

- Intel xxv710 NIC VF
  - New 25G version of widely used Intel 10G NIC, with RDMA supported
  - VF interface compatible with 10G NIC and later Intel E800 NIC
- Current prototype puts the VF in reset state before migration
  - A quick workaround to avoid copying device state
  - Long-term will follow aforementioned direction by doing state save/restore
- Track DMA pages by scanning ring descriptors in target machine
  - By mediating guest writes to the ring tail register
  - Pull the DMA page from source machine before sending to NIC for DMA R/W

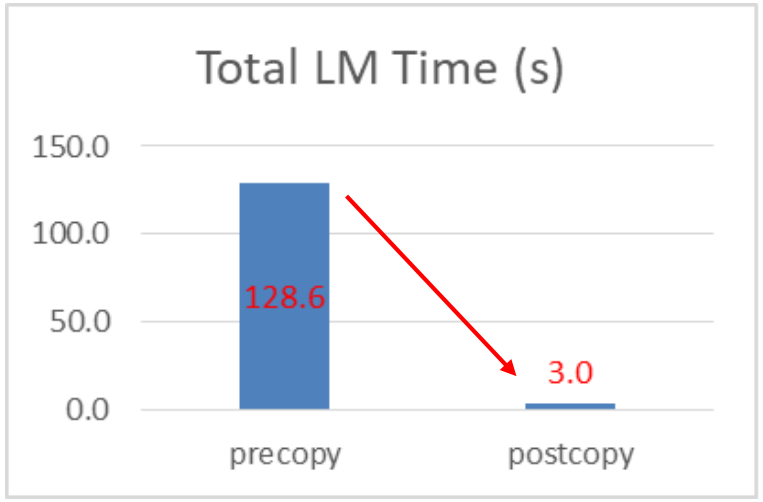
Block Diagram

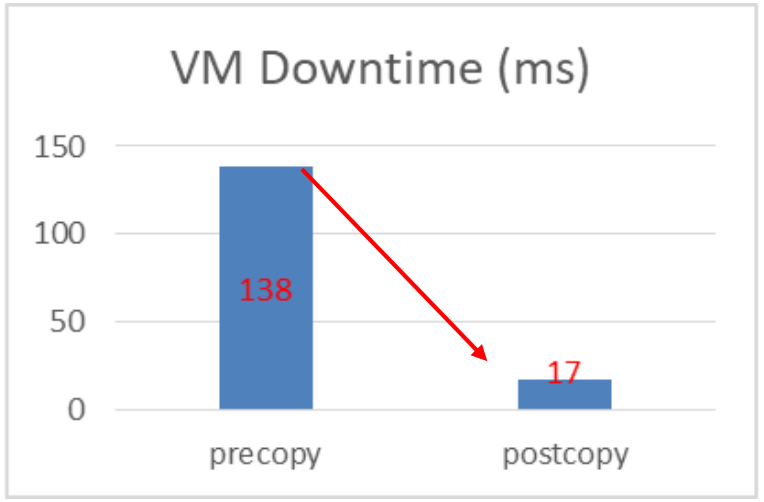


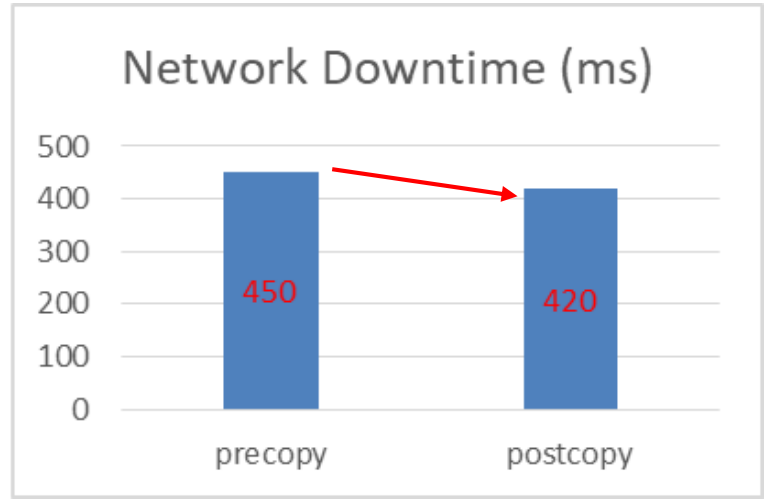
- Same framework for both pre-copy and post-copy
  - Pre-copy: track DMA writes on src
  - Post-copy: track DMA accesses on dest
- Mdev device model in PF driver
  - Manages ring mediation
  - GET/SET VF configuration data
- Easily extended to support migration cross different generations
  - Based on unified interface (AVF)

Performance

- Memory-intensive workload with 2GB guest memory
- 10Gbps migration bandwidth
- Network downtime measured by ping command







- Both include the time of device reset and switch configuration (~300ms)
- Postcopy data is based on optimized VFIO DMA mapping policy

Status and Plan

- Initial prototype work completed
- Future explorations
  - Remove reset
  - Extend userfaultfd or create VFIO-based interface?
  - IOMMU mapping efficiency
  - Huge page support
  - Device local memory
  - Extend to other NICs and device types (e.g. storage)
- Send out precopy RFC first, and then postcopy RFC

Q/A