CLK 2021

Device-passthrough Framework Refactoring – Modernized DMA Isolation Framework for Passthrough Devices

Yi Liu <yi.l.liu@intel.com>

Oct. 24th, 2021

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.

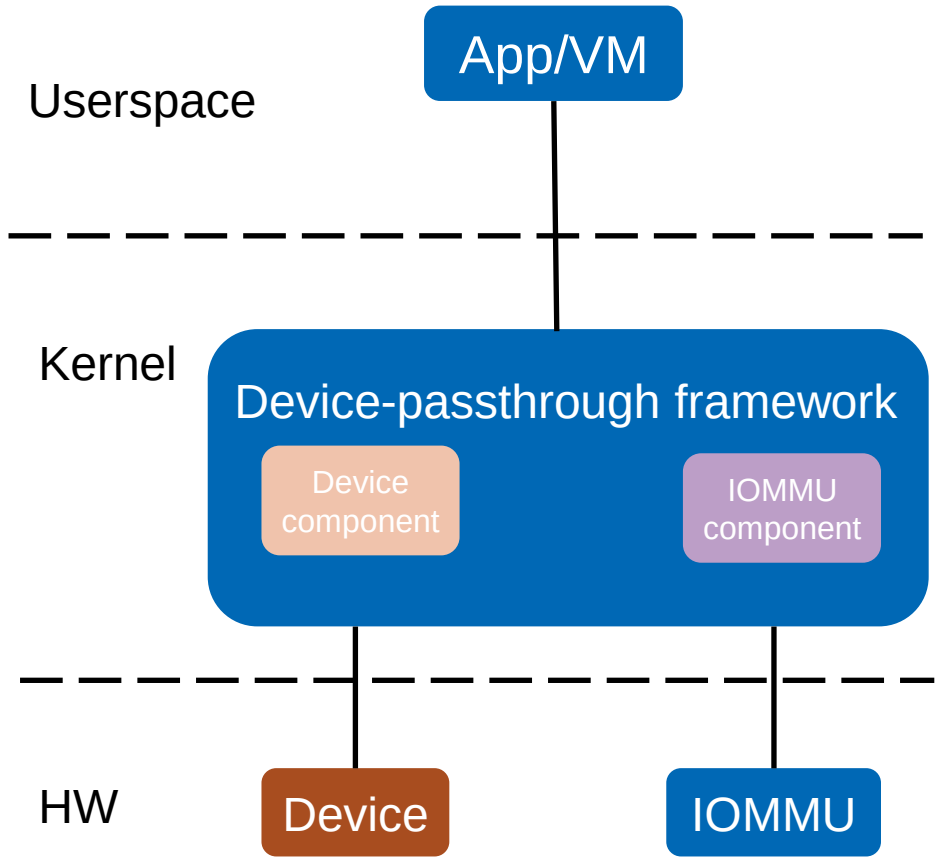
*Other names and brands may be claimed as the property of others

Agenda

- Backgrounds
  - Device Passthrough Recap
  - VFIO Recap
  - Challenges today
- IOMMU FD Proposal
  - Basic concepts for IOMMU FD proposal
  - Basic flow of IOMMU FD model
- Efforts for Adapting passthrough frameworks to IOMMU FD
- Q&A

Device Passthrough Recap

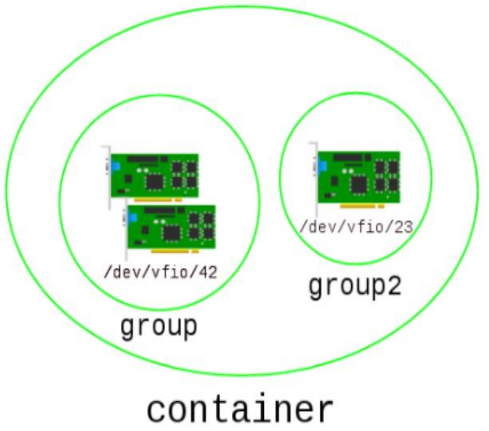
- Device passthrough
  - A way to gain BareMetal I/O performance in virtualization
  - A multi-layer userspace driver in Linux
- Device-passthrough framework
  - Device component
    - Handle the device access like PCI configuration space r/w, BAR mmap, interrupts, etc.;
    - Example: vfio-pci, vfio-platform;
  - IOMMU component
    - Handle the DMA isolation affairs;
    - Example: vfio iommu type1 driver;
  - Typical Device-passthrough Frameworks
    - VFIO, vDPA;



Device frameworks have their own iommu components today!

VFIO Recap

- A secure userspace driver framework
- Key concepts
  - Group FD
    - Get from /dev/vfio/iommu_group_id
    - Group is the smallest granularity for DMA isolation.
      - Relates to ACS, topology etc.
  - Container FD
    - Get from /dev/vfio/vfio
    - Provides an iommu context for userspace to program iommu
  - Device FD
    - Get via group fd after setting the corresponding group to a container
    - Provides device access



Container and Group[1]

```
container_fd = open("/dev/vfio/vfio", O_RDWR);
group_fd = open("/dev/vfio/42", O_RDWR);
ioctl(group_fd, VFIO_GROUP_SET_CONTAINER, &container_fd);
ioctl(container_fd, VFIO_SET_IOMMU, VFIO_TYPE1_IOMMU);
device_fd = ioctl(group_fd, VFIO_GROUP_GET_DEVICE_FD, name);
ioctl(container_fd, VFIO_IOMMU_MAP_DMA, &dma_map);
```

[1] http://events17.linuxfoundation.org/sites/events/files/slides/An%20Introduction%20to%20PCI%20Device%20Assignment%20with%20VFIO%20-%20Williamson%20-%202016-08-30_0.pdf

VFIO Recap (Cont.)

- Supported device types
  - PCI device, platform device, software-based mdev, hardware-based mdev (ADI), etc.
- Supported features in vfio iommu type1
  - DMA map/unmap
    - meta data management, page pinning, and related accounting
  - dirty bitmap retrieving
  - dynamic vaddr update, etc.

Challenges for Supporting new IOMMU Features

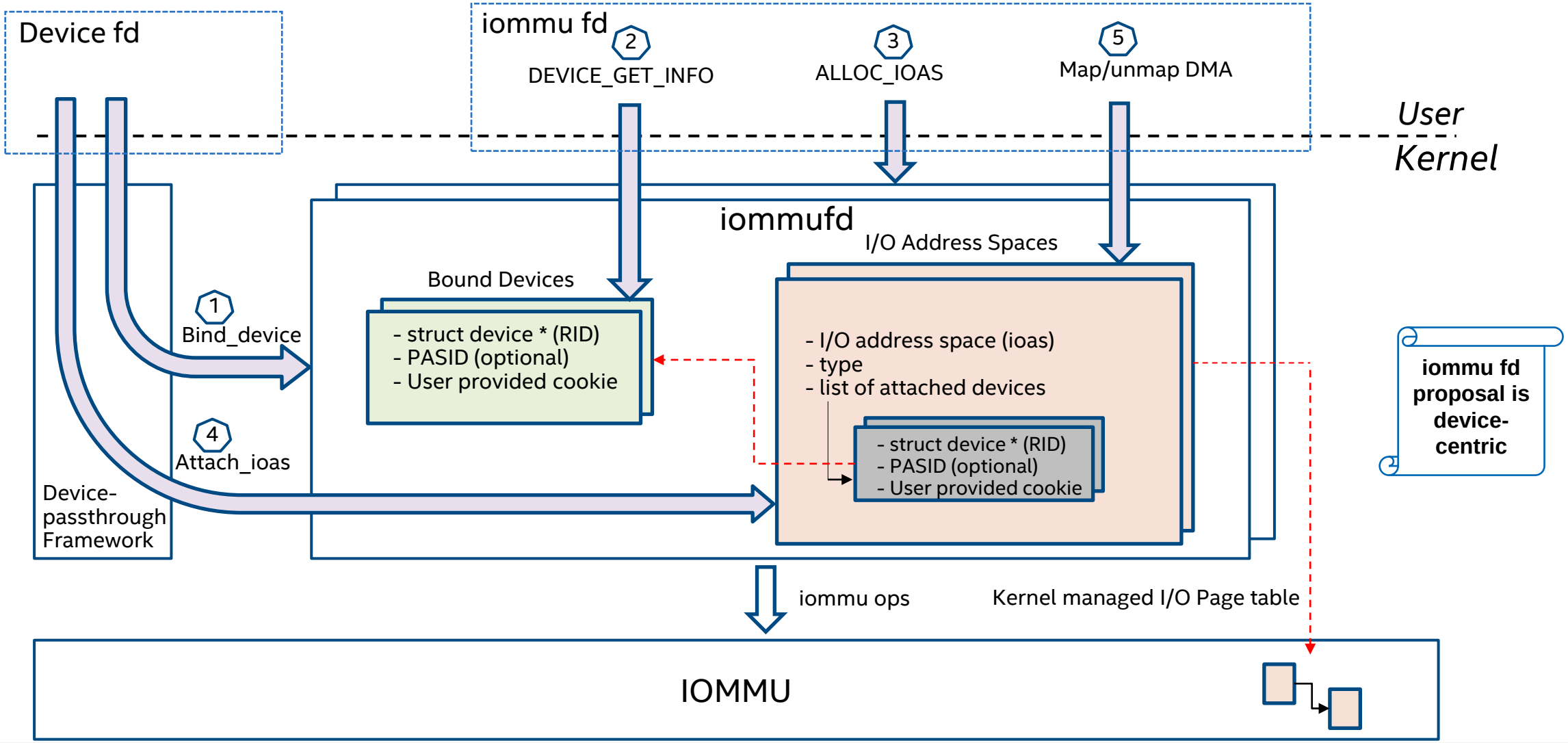
- Subdevice passthrough
  - Needs to support PASID (Process Address Space ID)/SSID (Sub-Stream ID)
- SVA virtualization
  - Needs to support user-provisioned I/O page table (nested on a hypervisor managed I/O page table),
- I/O page table dirty bit
- I/O page faults

Not scale to develop the new iommu features for each device-passthrough framework

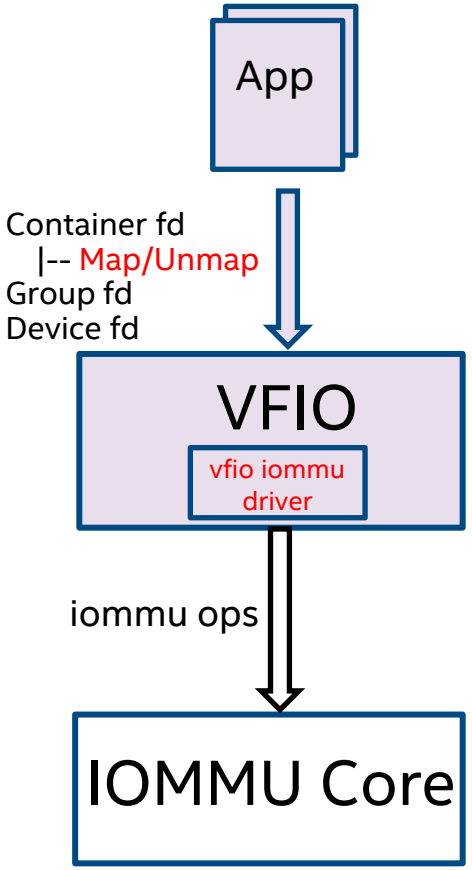
IOMMU FD Based DMA Isolation Framework

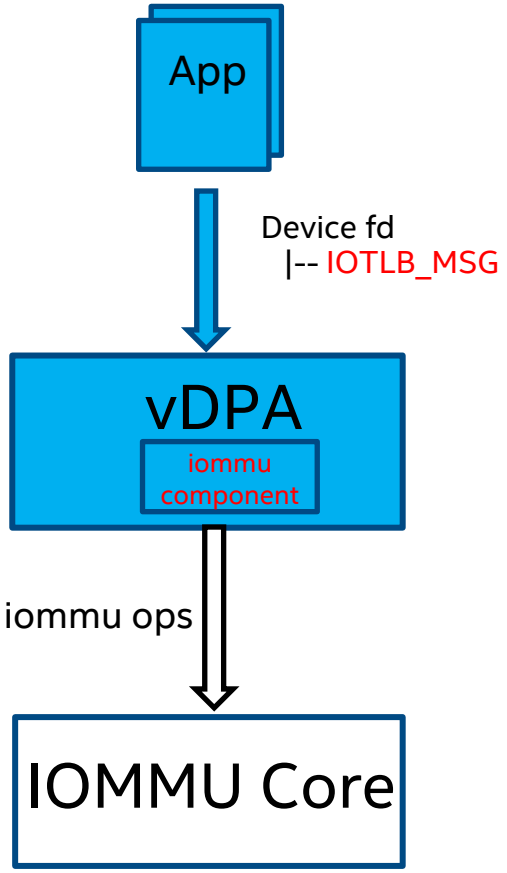
- IOMMU FD
  - Got per /dev/iommu opening
  - The container holding multiple I/O address spaces
  - Support the iommu operations (e.g., DMA map/unmap) from userspace
- IOAS
  - An iommufd-local software handle representing an I/O address space
  - Allocated via iommufd
- Protocols with device-passthrough frameworks
  - Bind/unbind device to iommufd
  - Attach/detach selected IOAS to device
- Device types to support
  - PCI device, platform device, software-based mdev, hardware-based mdev (ADI), etc.

Basic Flow of IOMMU FD Proposal

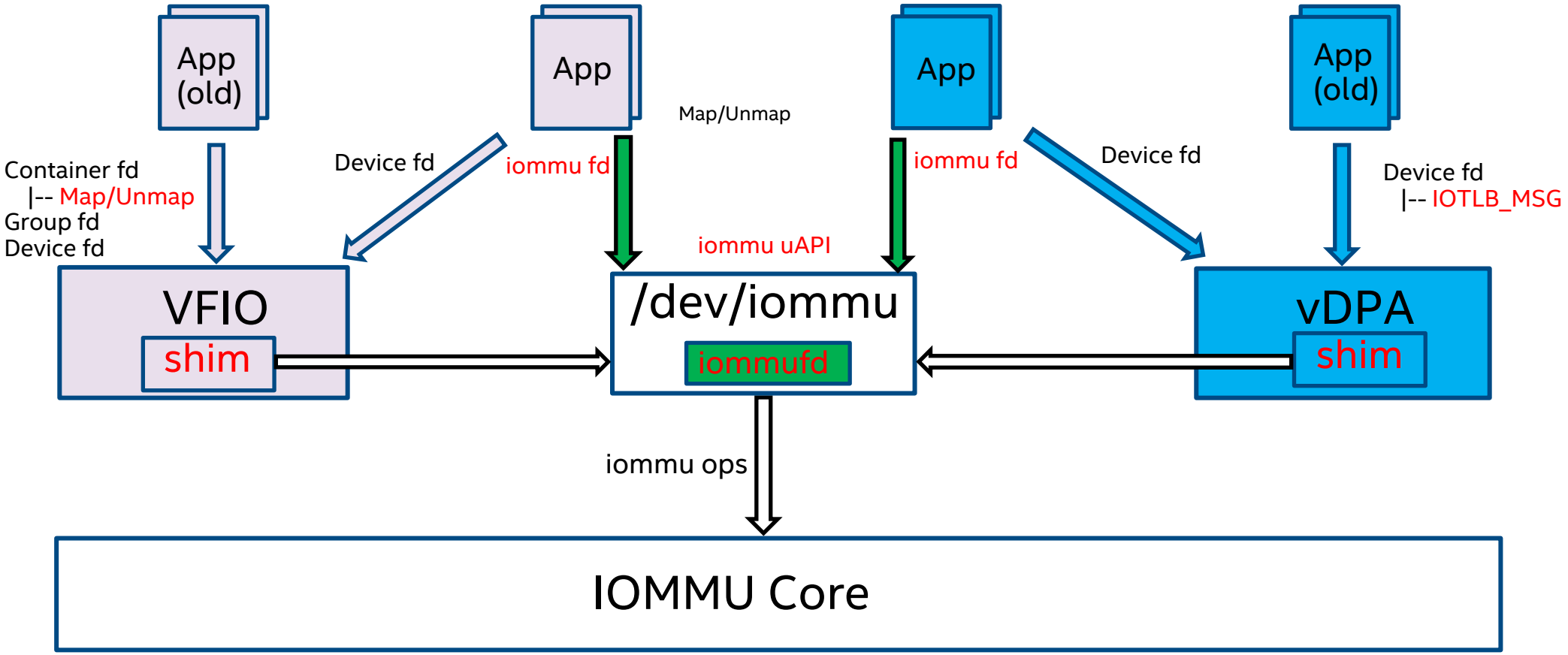


Adapting VFIO/vDPA to IOMMU FD

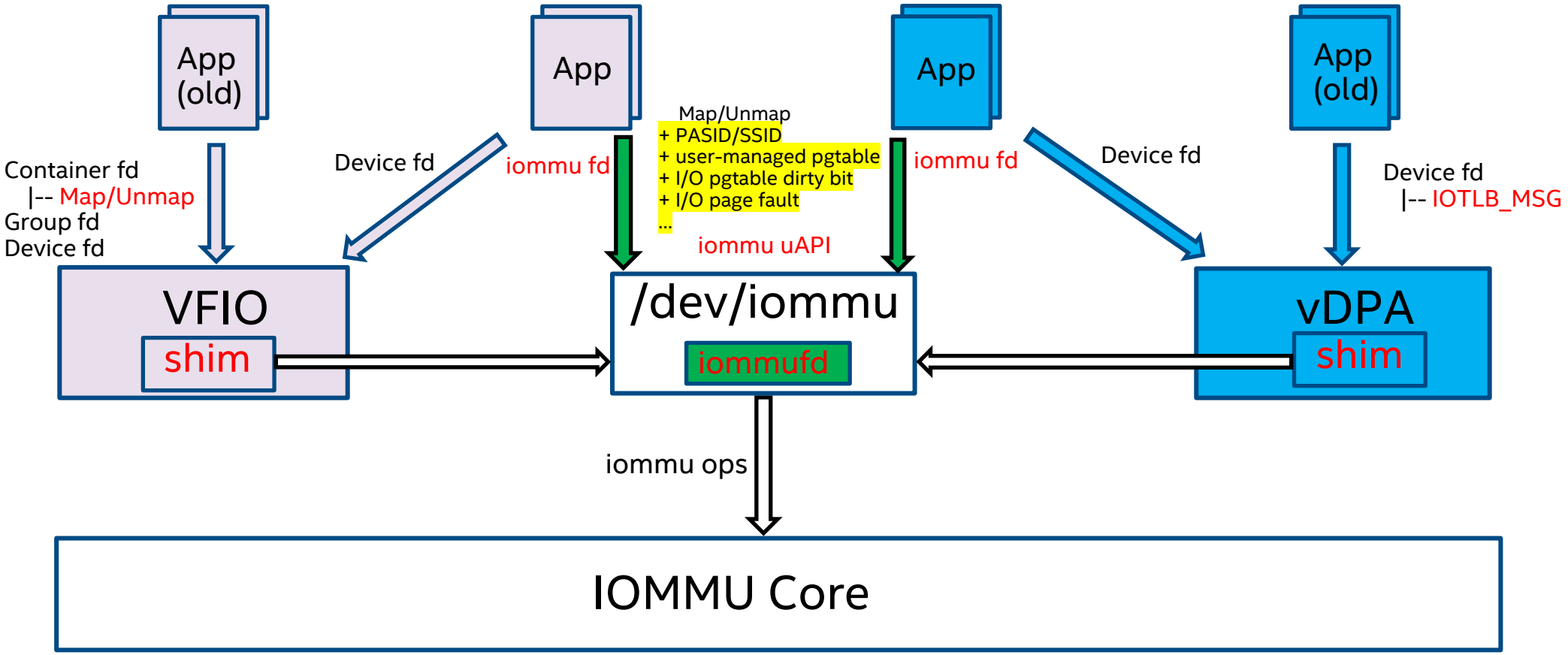




Adapting VFIO/vDPA to IOMMU FD (Cont.)



Adapting VFIO/vDPA to IOMMU FD (Cont.)



Efforts for Adapting VFIO to IOMMU FD

- Introducing device centric interface to VFIO
  - Exposing vfio_device under /dev/vfio/devices/
    - Only allow device access after secure DMA context is setup (done after binding vfio_device to an iommufd)
      - iommufd uses iommu_device_init[exit]_user_dma() for the secure DMA context establishment
  - Allowing co-existence of the legacy group/container interface and the new device-centric interface
    - Userspace can only access device via one of the two interfaces but not both at the same time
- Implementing the protocols defined by iommufd
  - Bind/unbind device to iommu fd
  - Attach/detach ioas to device
- Abstracting the vfio iommu type1 driver to be used by iommufd
  - Features in type1 driver are needed in iommufd
  - over 3000LOC with ~80% related to dma management
- Ensure backward compatibility
  - Existing userspace application should still work

Efforts for Adapting VFIO to IOMMU FD (Cont.)

- Status
  - RFCv1was sent out
    - https://lore.kernel.org/kvm/20210919063848.1476776-1-yi.l.liu@intel.com/#t
    - Team-work across Intel, Nvidia, Redhat, ARM, AMD, etc.
    - Only PCI devices and only basic DMA map/unmap feature supported by iommufd in RFCv1.
    - QEMU part change was not sent out yet
- Helpful links
  - The original discussion inspired iommufd
    - https://lore.kernel.org/linux-iommu/20210330132830.GO2356281@nvidia.com/
  - iommufd uapi proposal discussion
    - https://lore.kernel.org/kvm/BN9PR11MB5433B1E4AE5B0480369F97178C189@BN9PR11MB5433.namprd11.prod.outlook.com/

Summary

- Existing device-passthrough framework implementations doesn't scale as new IOMMU features step in
- IOMMU FD proposal is a unified DMA isolation framework for device-passthrough frameworks
- IOMMU FD model is device-centric, thus requires device-passthrough framework to support device-centric user-interface
- Refactoring device-passthrough frameworks to IOMMU FD creates a good base for supporting new IOMMU features like Scalable IOV, SVM, I/O page table dirty bit, I/O page fault etc.

Q&A


intel