KVM FORUM

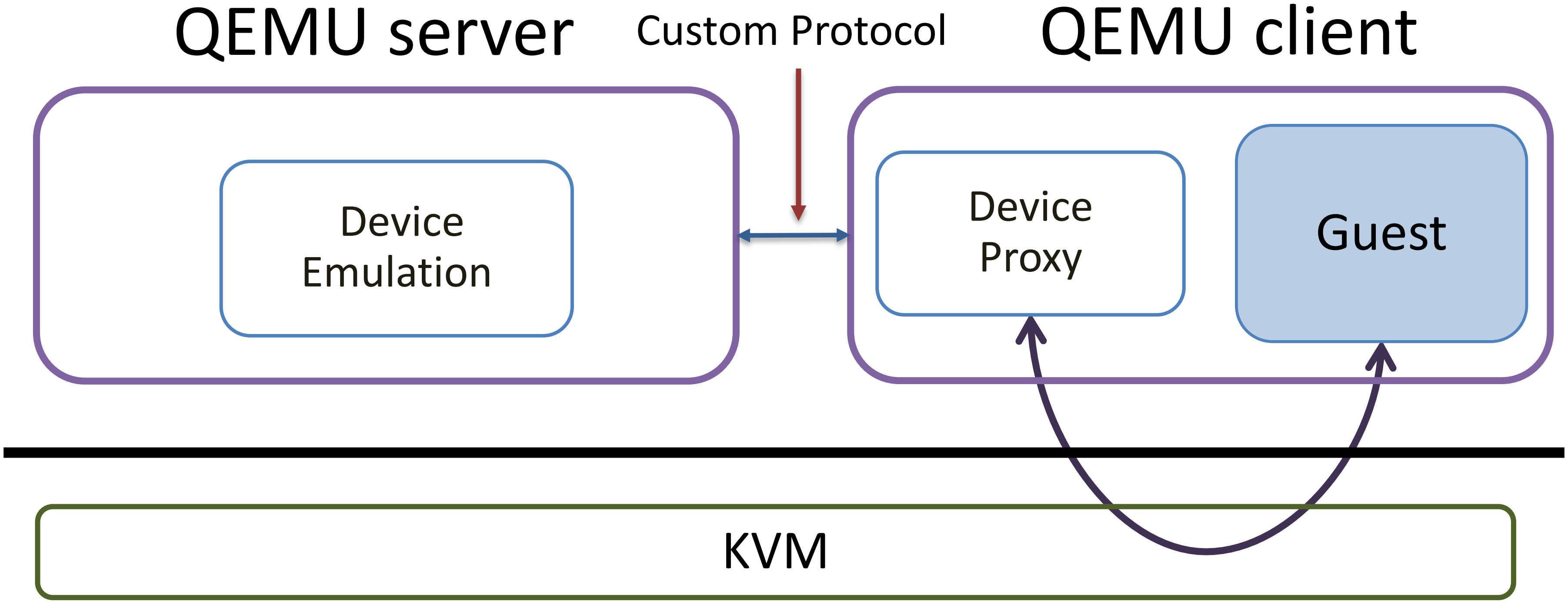
VFIO-User

Remote Device Emulation using VFIO

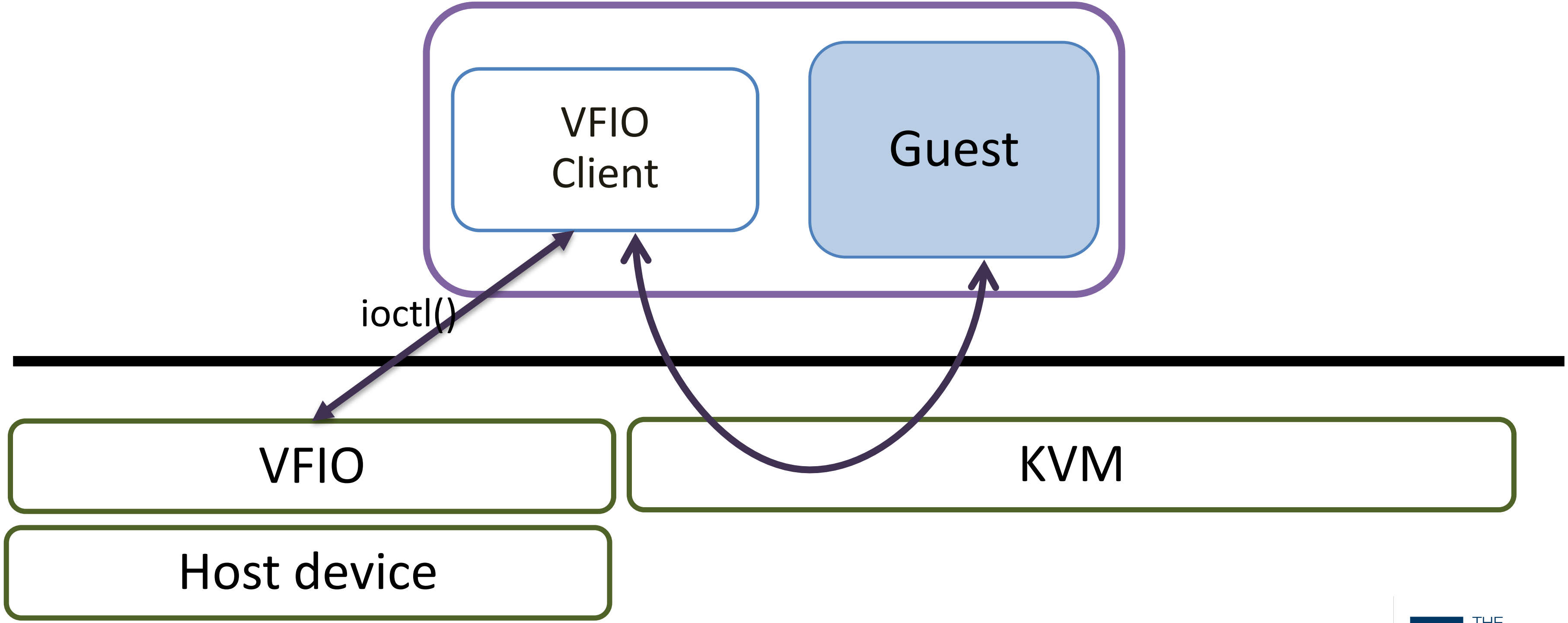
John Johnson
Jag Raman
Elena Ufimtseva

THE LINUX FOUNDATION

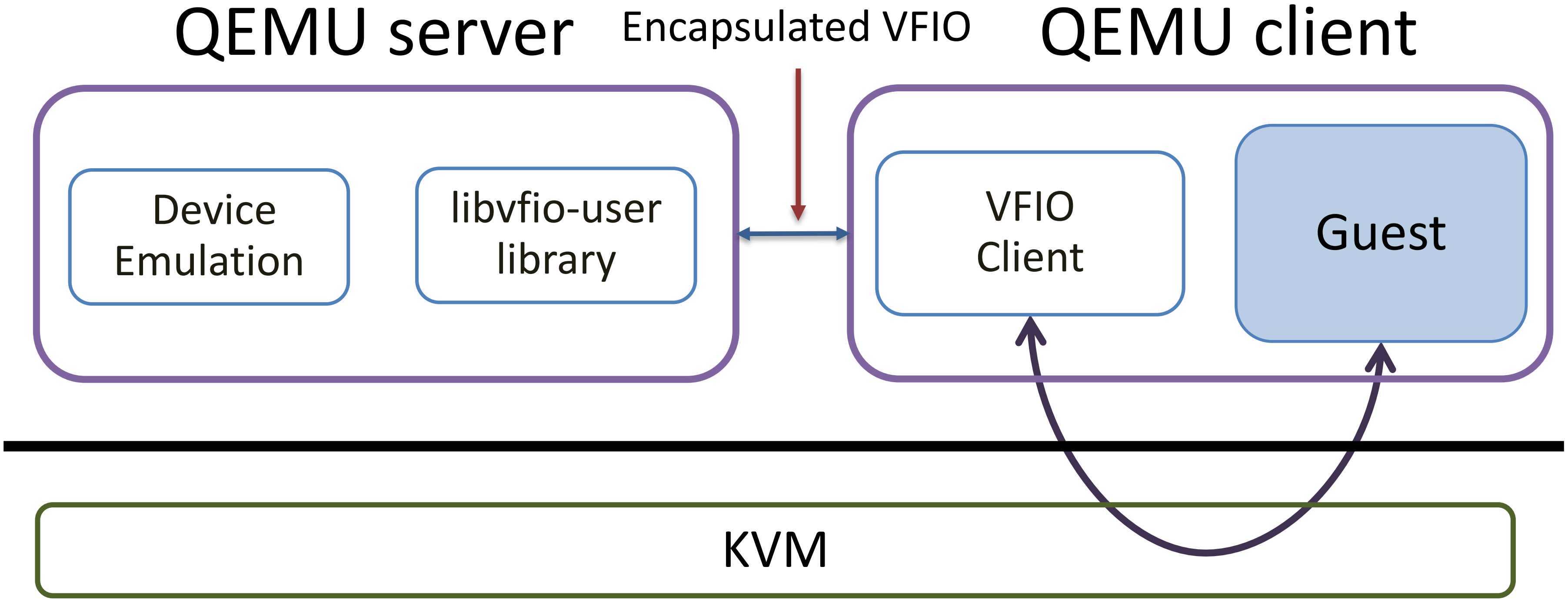
Multi-process QEMU in 6.0



Existing VFIO



VFIO-User


QEMU server
Encapsulated VFIO
QEMU client
Device Emulation
libvfio-user library
VFIO Client
Guest
KVM

libvfio-user and friends

- QEMU Client and Server
  - https://github.com/oracle/qemu.git
- libvfio-user library
  - https://github.com/nutanix/libvfio-user.git
  - C binding used by QEMU server and SPDK
  - Other language bindings, such as RUST are possible
  - Checkout update from Nutanix today
- SPDK
  - https://github.com/spdk/spdk.git
  - Intel already presented high-performance NVMe offload

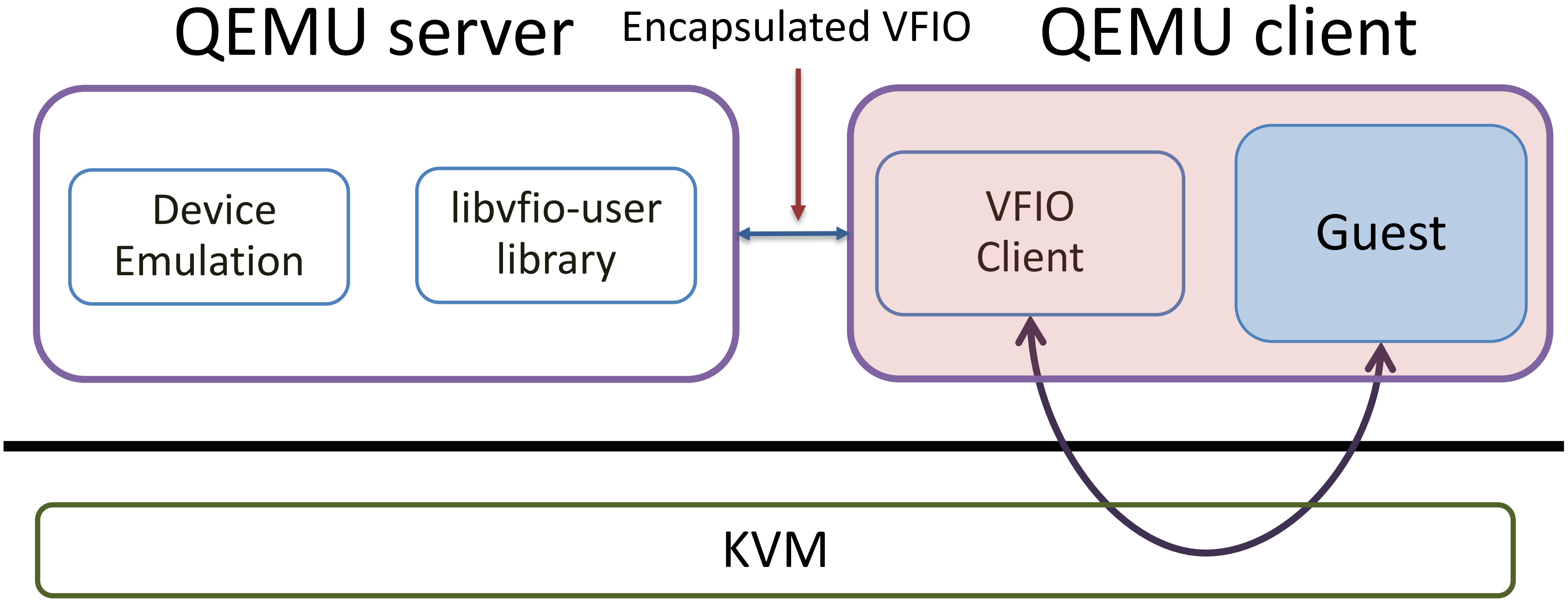
VFIO-User vs. multi-process QEMU

- Uses established QEMU VFIO client instead of a custom-made ‘proxy’ object
  - Most of the code in the ioctl() implementation can be re-used in the socket implementation
  - Leverage existing VFIO features like IOMMU and migration support
  - No duplicated maintenance effort
- The protocol is changed to an encapsulation of the ioctl() structures sent to the kernel VFIO driver

VFIO-User vs. VFIO

- User space only - no kernel modifications are needed
  - It is VFIO-User, after all
  - No kernel driver modifications
  - No /sys or /dev/vfio files are used
    - no privileged configuration changes needed

VFIO-User


QEMU server
Encapsulated VFIO
QEMU client
Device Emulation
libvfio-user library
VFIO Client
Guest
KVM

VFIO-User Client implementation

- We shared as much code as possible with VFIO ioctl() implementation
  - Defined new abstract super-class for both types
  - Biggest differences are in option parsing and in setup/teardown of the device object
  - Most others are low-level checks of whether to issue an ioctl() or send a message over the socket

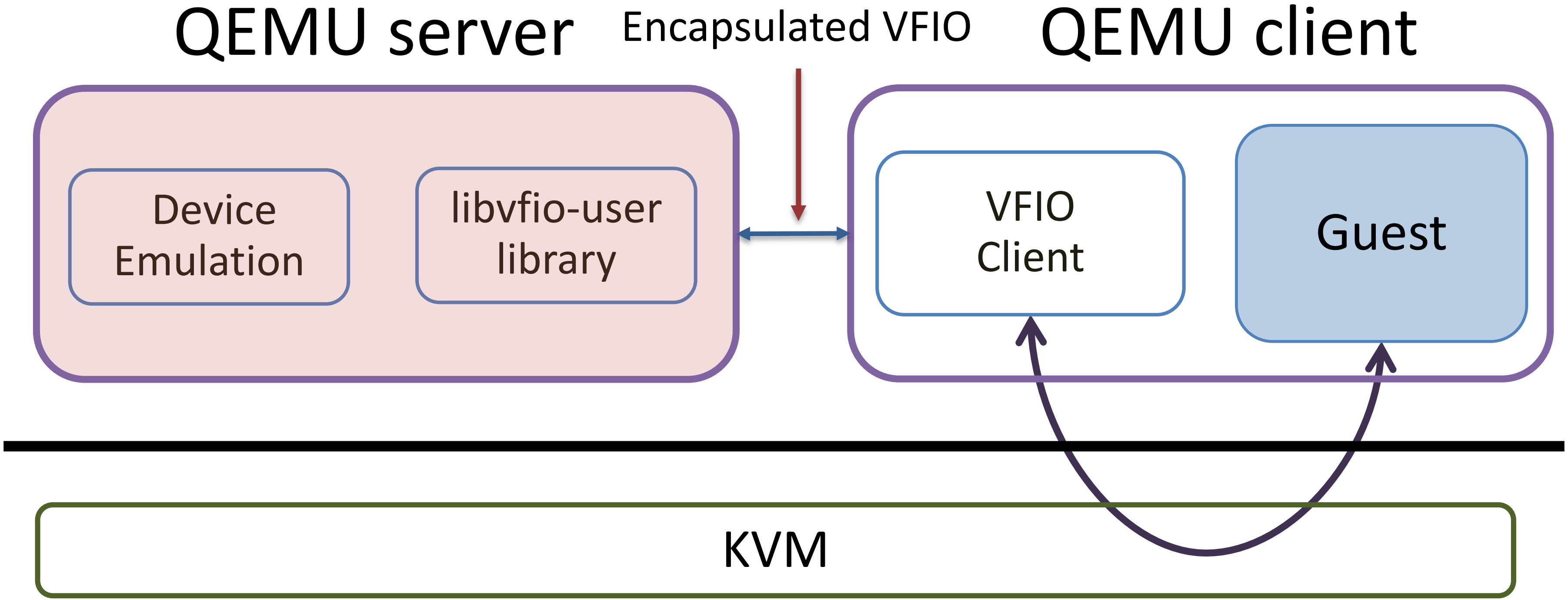
VFIO-User Client implementation

- Use an iothread to receive packets from server
  - Incoming packets are classified as:
    - replies that signal waiting CPU threads
    - requests to be processed by the VFIO client
  - All devices currently handled by single thread but can easily be changed if scalability is an issue

VFIO User Client implementation

- Do not want to hold BQL while blocking for server replies
  - Use per-socket mutex instead
  - Have to be careful not to drop BQL when messages are sent by address space change transactions
    - these transactions are serialized by BQL
    - send messages async, then wait for all when transaction commits

VFIO-User

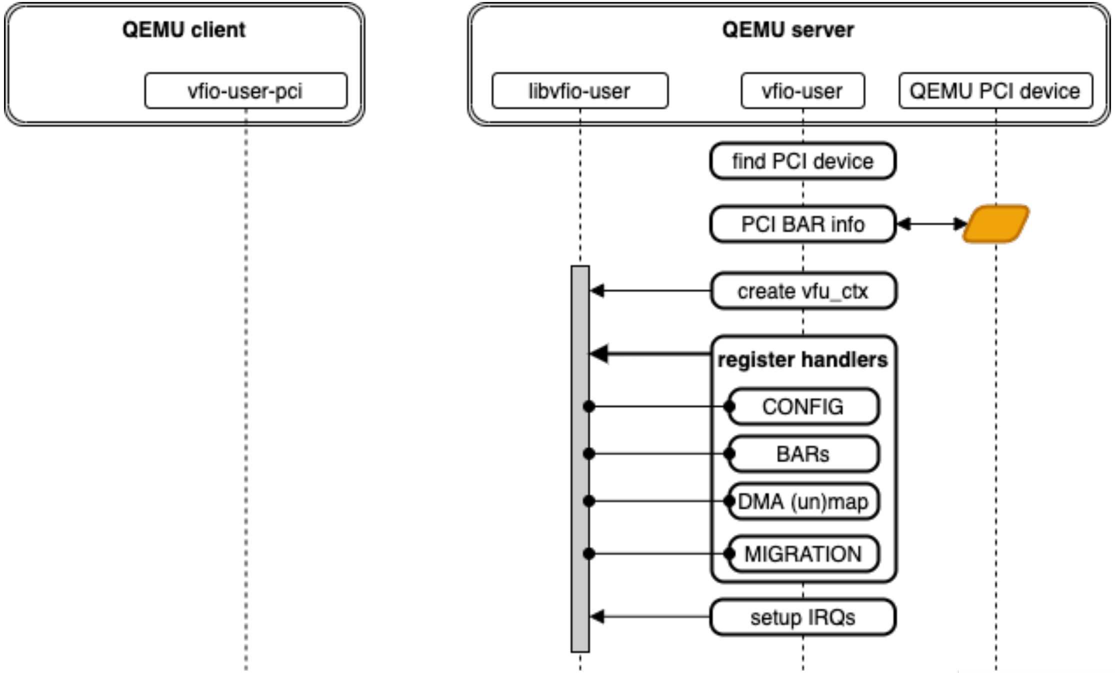

QEMU server
Encapsulated VFIO
QEMU client
Device Emulation
libvfio-user library
VFIO Client
Guest
KVM

VFIO User Server implementation

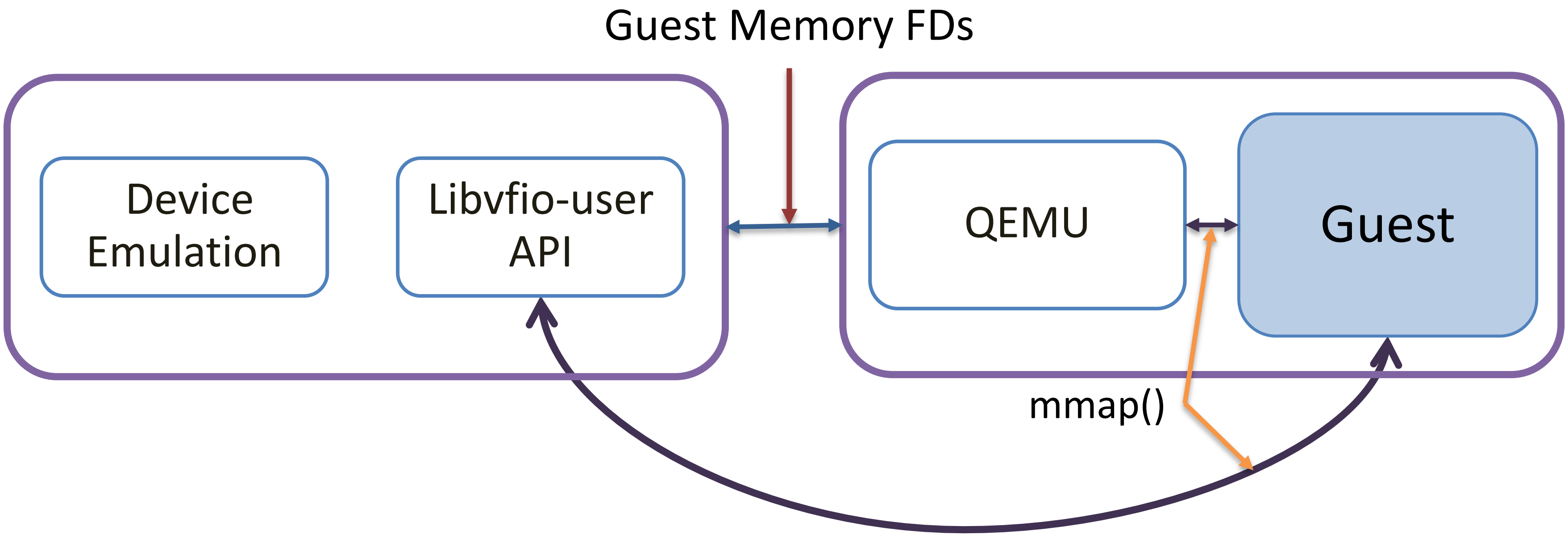
- Consists of the following major components
  - ‘x-remote’ machine
  - pci-host bridge
  - IOHUB
  - Libvfio-user
  - vfio-user object

QEMU Server Init

- Create **vfu_ctx**
  - device handle
  - named socket
- Register call-backs
  - CONFIG
  - BARx
  - DMA map/unmap
  - Migration
- Driven by QEMU main-loop

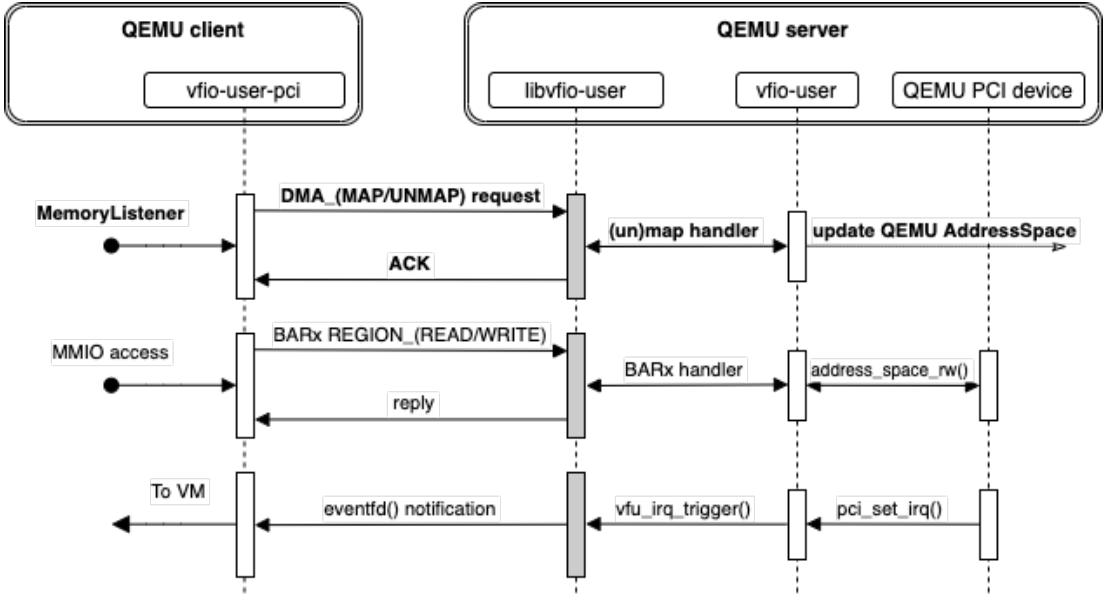


VFIO User DMA



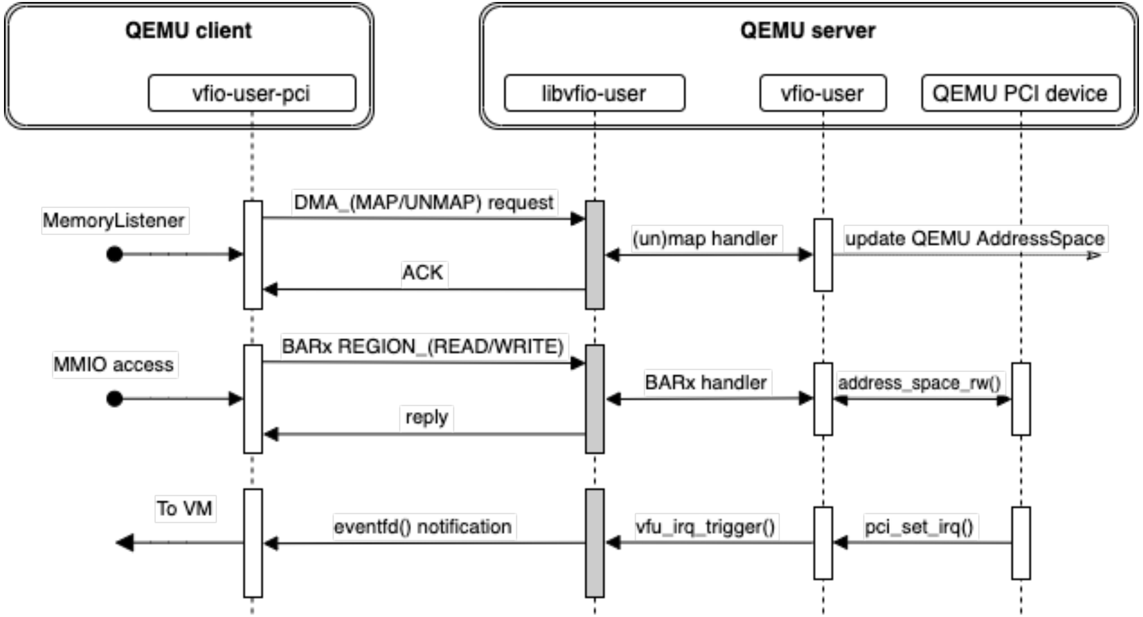
VFIO-User servicing VM

- DMA Map / Unmap
  - MemoryListener notifies RAM updates
  - supports IOMMU enabled guests
  - send fd to allow mapping guest RAM in server



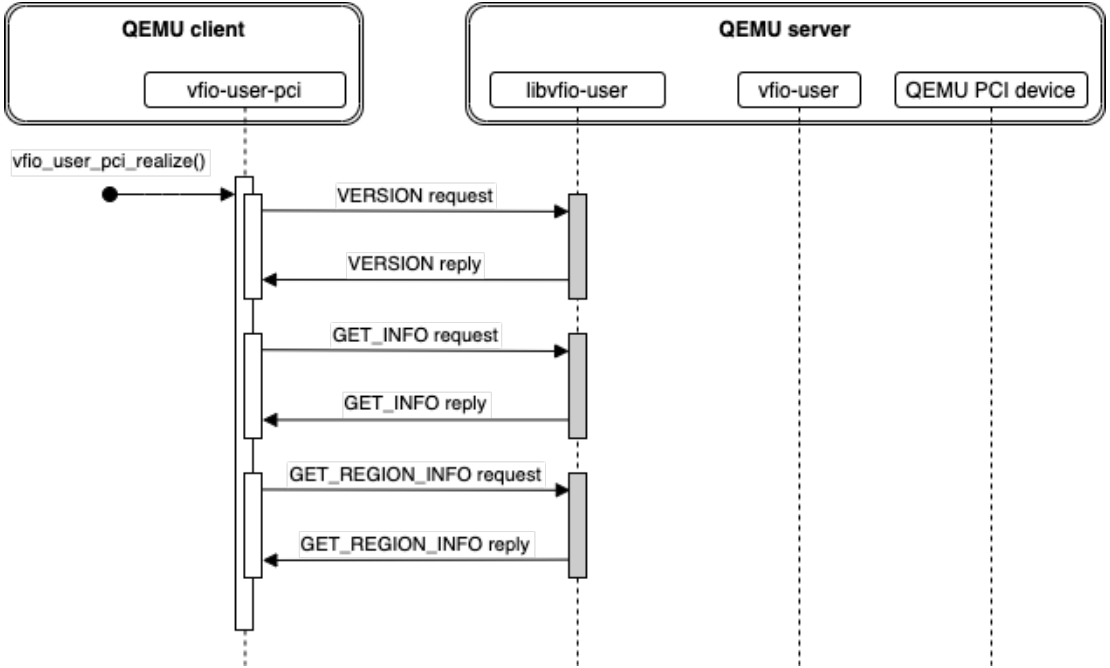
VFIO-User servicing VM …

- BARx access
  - REGION_READ & RRGION_WRITE commands
  - similar command for CONFIG space access
- Interrupts
  - signal eventfd



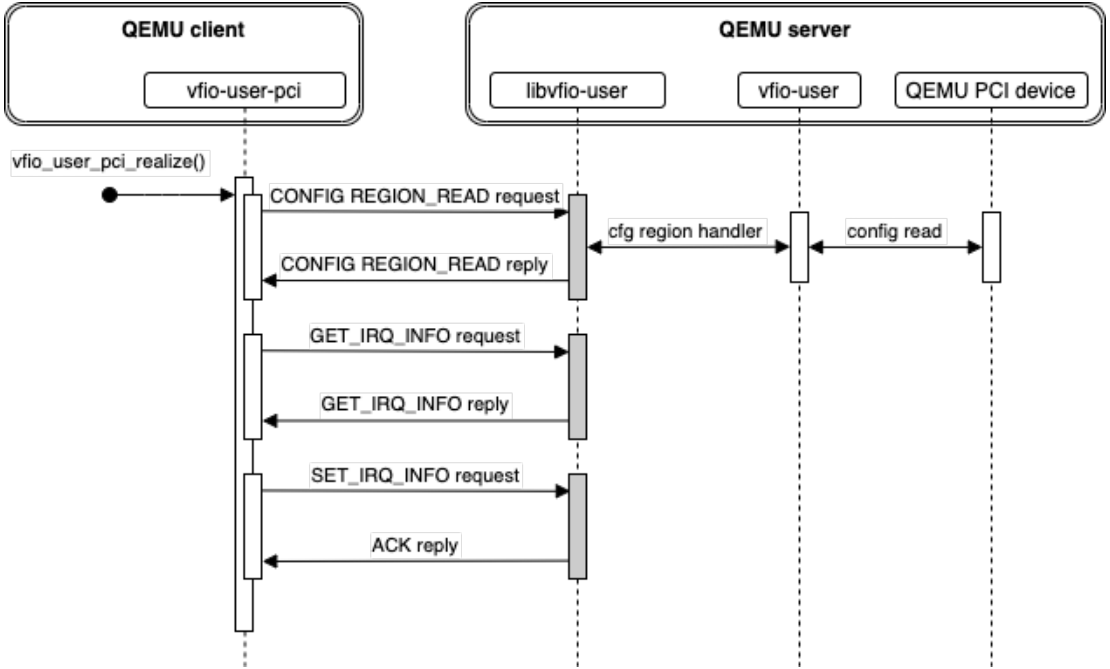
QEMU Client Init

- VERSION command
  - client proposes version
  - server returns compatible version
  - server also returns the capabilities it supports
- GET_INFO command
  - gets device description such as #regions, #IRQs
- GET_REGION_INFO
  - description of region
  - server can return fd for memory mapping

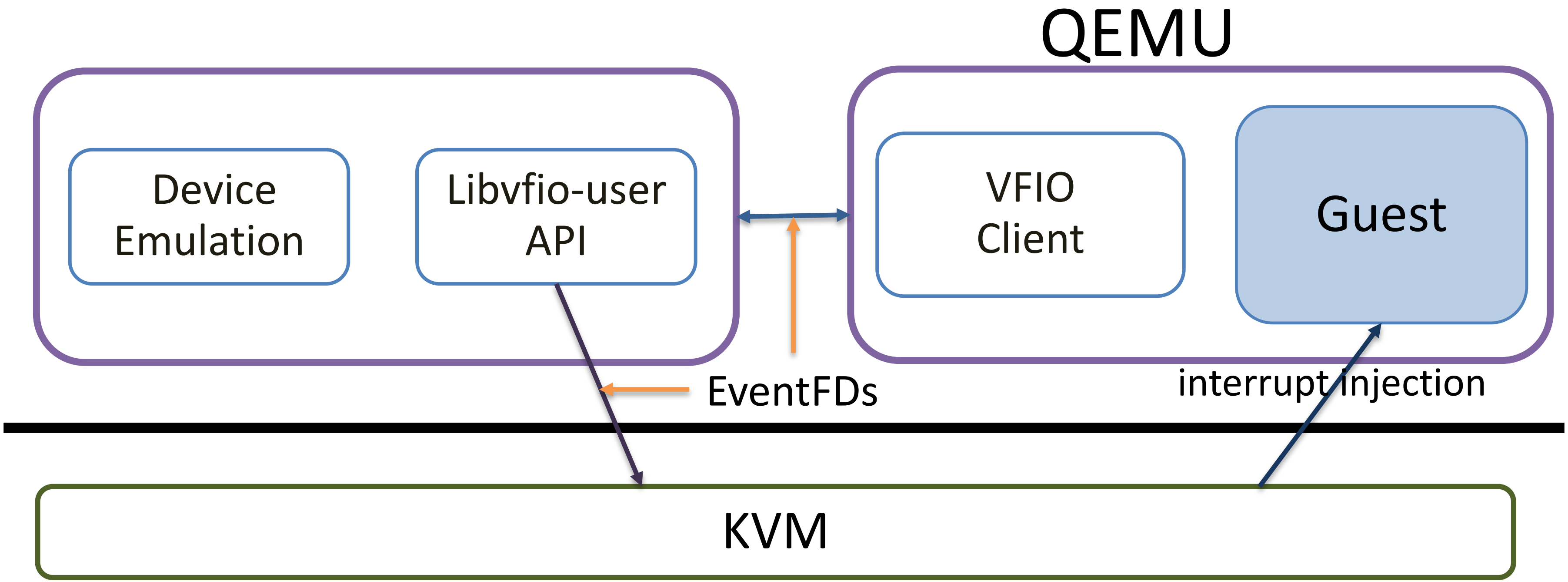


QEMU Client Init …

- CONFIG_REGION_READ
  - read entire config space from server; shadow copy
- GET_IRQ_INFO
  - returns #IRQ vectors
- SET_IRQ_INFO
  - send IRQ info
  - send eventfd to be used with KVM_IRQFD



VFIO User Interrupts



QEMU

Device Emulation

Libvfio-user API

VFIO Client

Guest

EventFDs

interrupt injection

KVM

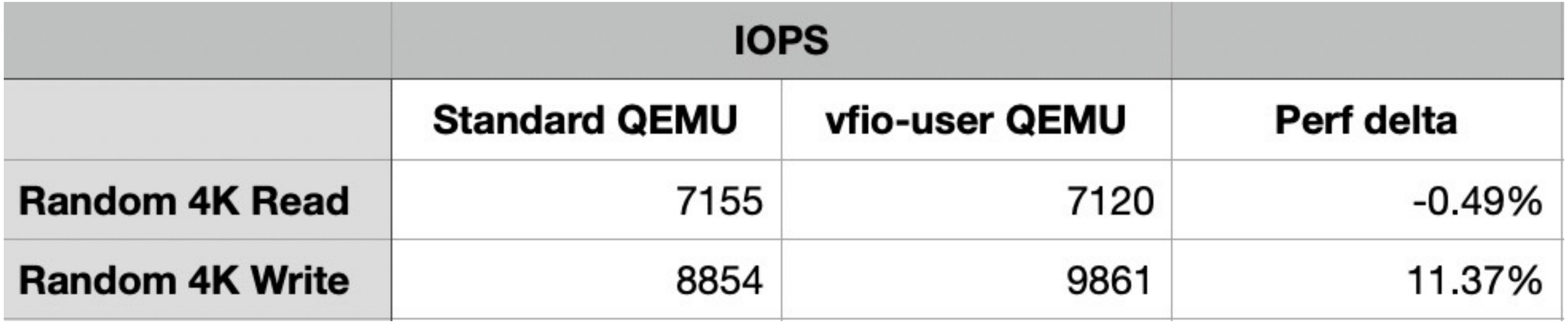
DMA_READ, DMA_WRITE commands

- Requests from server read from or write to guest memory
  - Used when guest memory is not backed by a file descriptor
  - Also used with ‘secure-dma’ command line option
    - indicates the client does not want the server to directly access guest memory
    - DMA_MAP never includes an FD if set

DIRTY_PAGES command

- Sent from client to server during migration to retrieve a bitmap of pages dirtied by DMA
  - Server then clears the mask for the next incremental request
- There also is an option to DMA_UNMAP that asks for the dirty bitmap of the area being unmapped

Performance numbers



	IOPS		
	Standard QEMU	vfio-user QEMU	Perf delta
Random 4K Read	7155	7120	-0.49%
Random 4K Write	8854	9861	11.37%

Futures

- ioregionfd
- New socket types
  - VSOCK? TCP?
- Non-PCI bus support
  - ISA? USB?
- bdrv_inactivate_all()

Demo

KVM

FORUM