Rust Based Virtio Backends for Hypervisor Agnostic Solutions

Alex Bennée / Viresh Kumar
Linaro

Introduction

- Viresh Kumar
  - Senior Kernel Engineer @ Linaro
  - Co-maintainer cpufreq, opp @ Linux
  - IRC: vireshk
- Alex Bennée
  - Senior Virtualisation Engineer @ Linaro
  - Project StratosTech Lead
  - IRC: stsquad/ajb-linaro

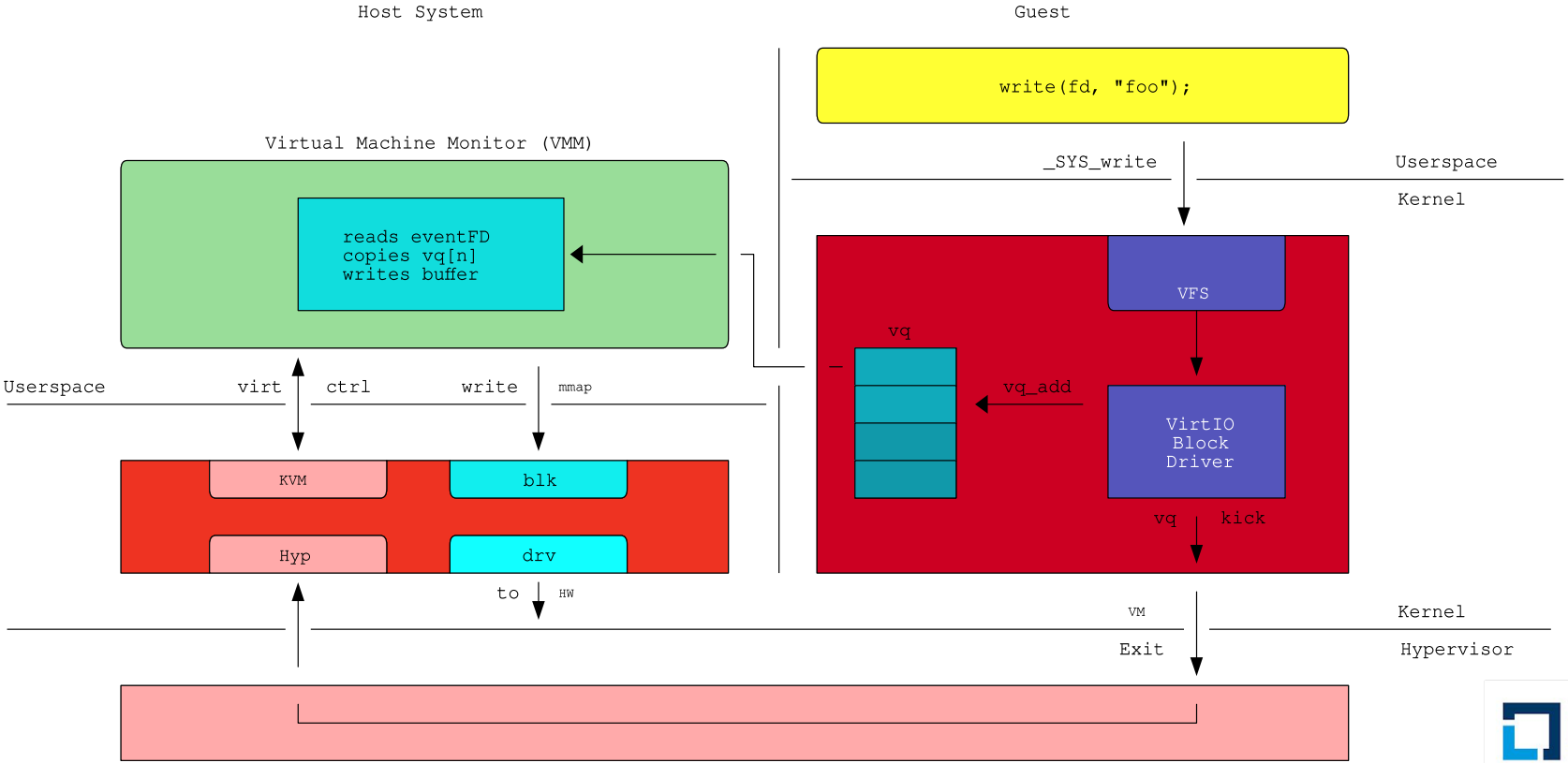
Topics

- Intro to VirtIO / Vhost / Vhost-user
- Linaro’s Project Stratos
- Future work

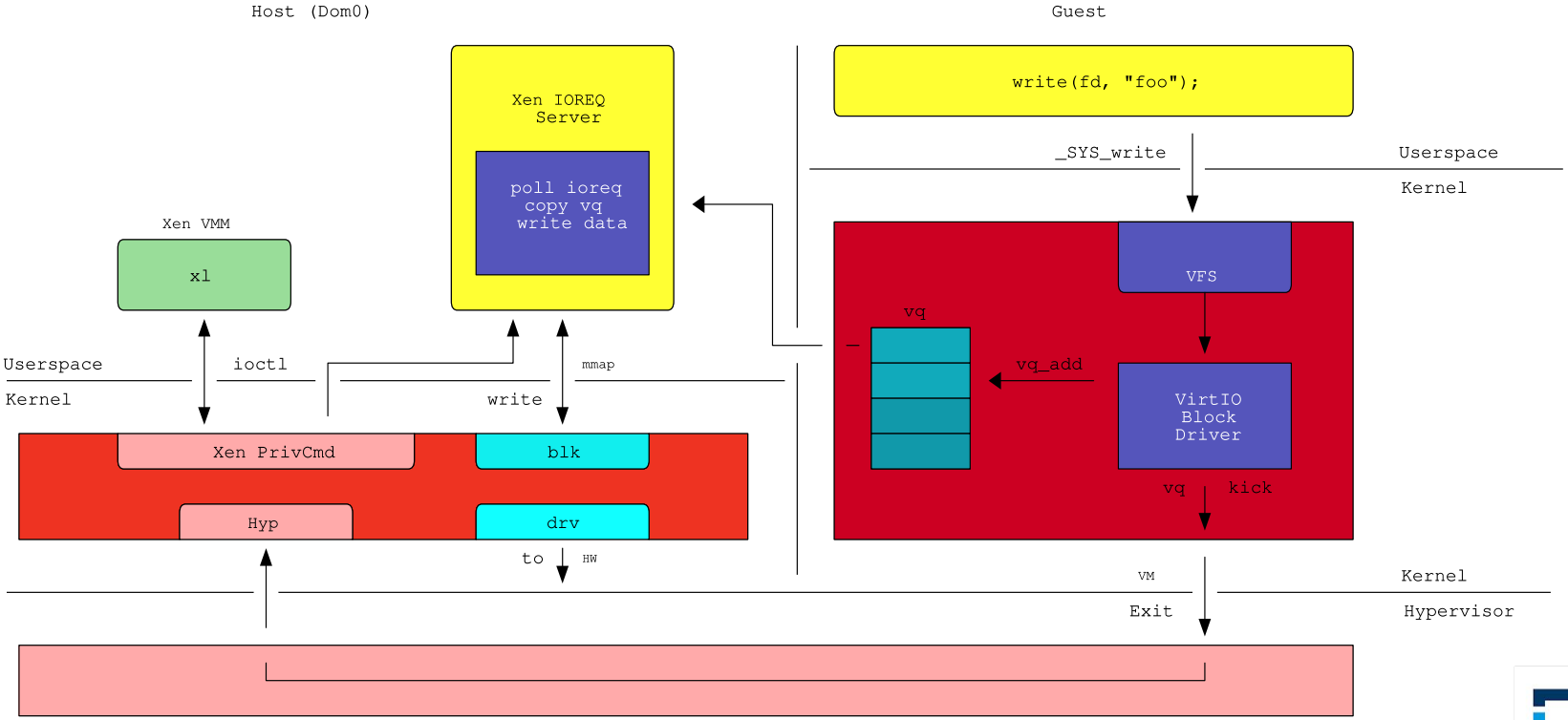
VirtIO

- Standardized open interface for VMs
- Maintained by OASIS
- Host - VirtIO Device
- Guest - VirtIO Driver
- Transport - PCI, MMIO, Channel I/O (s390x)

KVM VirtIO path



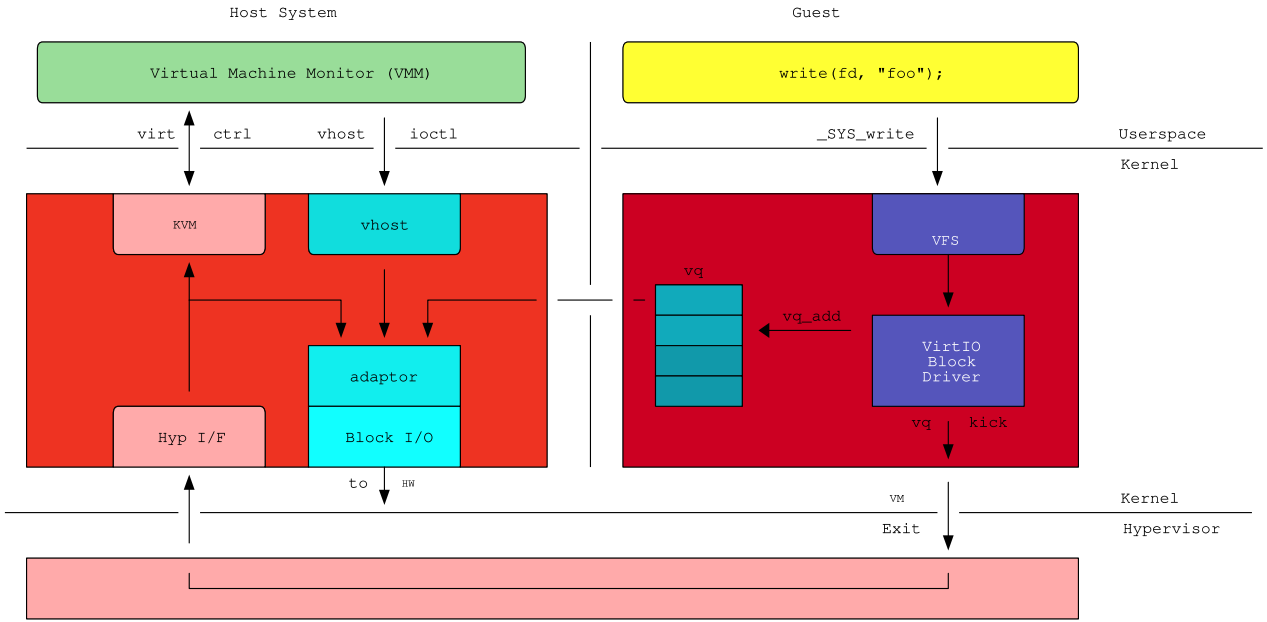
Xen VirtIO Path



Vhost

- Protocol to offload datapath processing.
- Processing happens in the host kernel.
- Implements control path via ioctls to host.

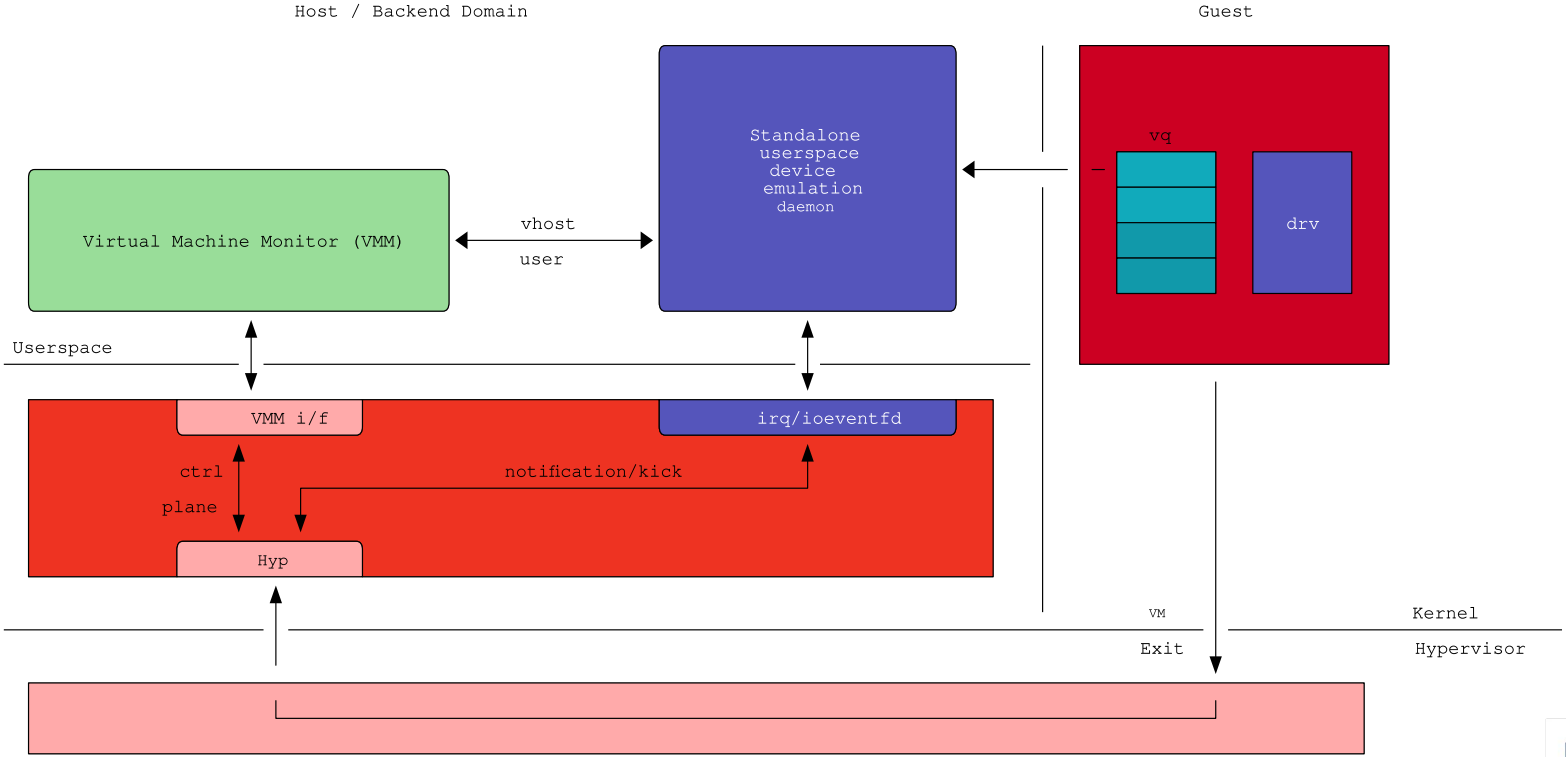
Vhost VirtIO path



Vhost-user

- Processing happens in host userspace.
- Control path via Unix domain socket.
- Defines front-end and back-end.
- Front-end shares its virtqueues (VMM).
- Back-end consumes the virtqueues.

Vhost-user VirtIO path



Linaro's Project Stratos

- Hypervisor Agnostic VirtIO
  - Decouple backends from Hypervisor
- Standardization and Upstreaming
  - Upstreaming VirtIO specification
  - Reviewed and implemented kernel drivers
  - Linux userspace vhost-user daemons
  - Rust bindings for libgpiod
  - Qemu / Xen support

Rust based vhost-user daemons

- Rust
  - Performance, safety and concurrency.
  - Safely handling untrusted guest data
  - Rust is cool

- Rust-vmm
  - Rust framework for building VMMs
  - Common components to share
    - CrosVM, Firecracker, Cloud Hypervisor

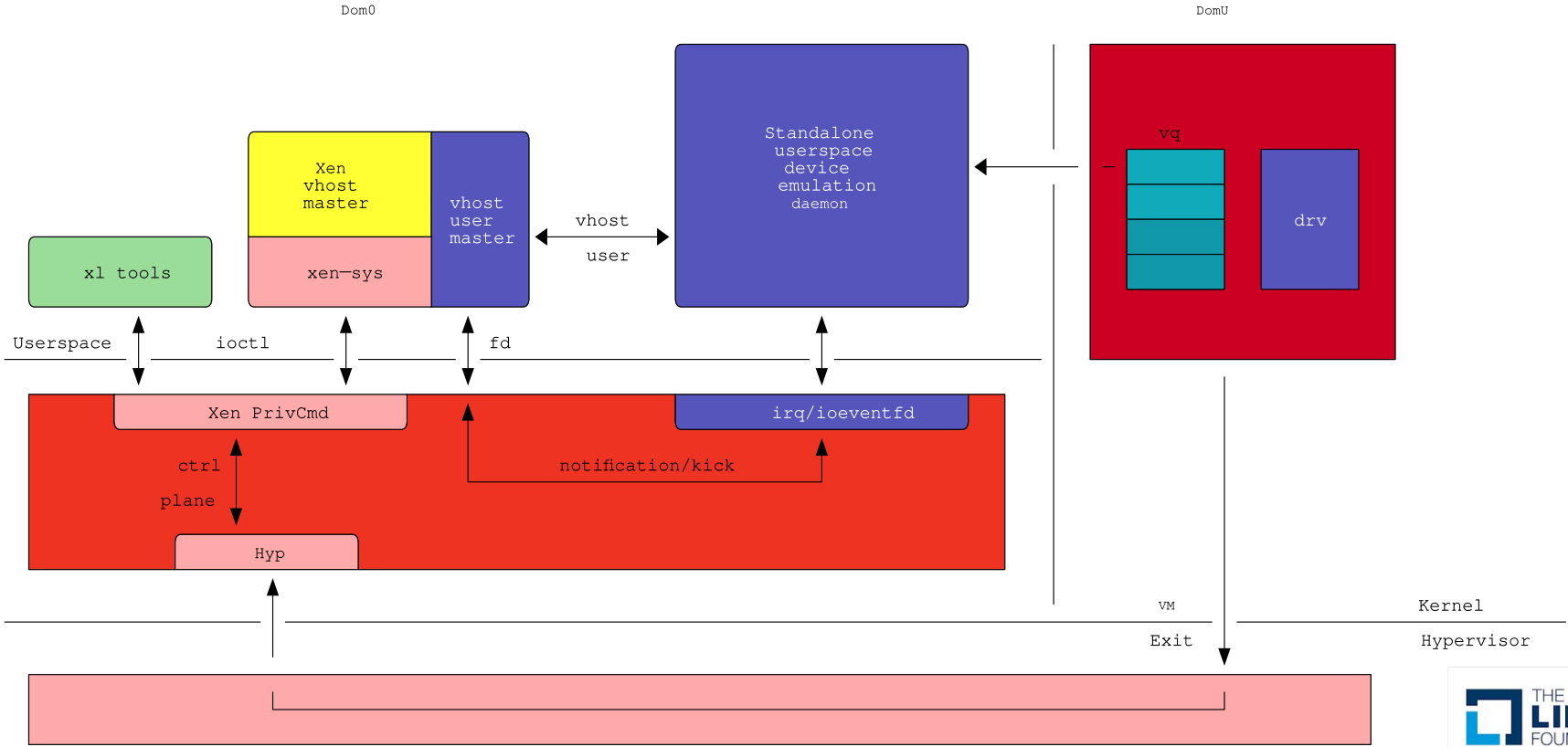
Vhost-device Implementations

- Upstreamed (rust-vmm / vhost-device)
  - gpio
  - i2c
  - rng
- WIP
  - rpmb
  - scmi
  - video
  - vsock
  - scsi

Xen vhost-master Implementation

- Backends developed with QEMU initially.
- Unmodified backends tested with Xen.
- Working implementation available now.
- Backends are truly Hypervisor agnostic.
- WIP to upstream to rust-vmm.

Xen vhost-master (Cont.)



Xen vhost-master (Cont.)

- Components
  - Xen-sys
    - Xen Hypercalls
    - Ioctls via kernel
    - Direct bare metal
  - Vhost user-master
    - Master side of vhost-user protocol
    - Reused from cloud-hypervisor
  - Xen vhost-master
    - Xen specific vhost-master implementation
    - Based on EPAM's virtio-disk implementation

Xen’s User vs Kernel Pages

- Xen doesn’t use /dev/shm
- /dev/xen/privcmd, uses kernel memory.
  - Kernel sets user PTE to invalid for short time
  - Xen gets EFAULT on access

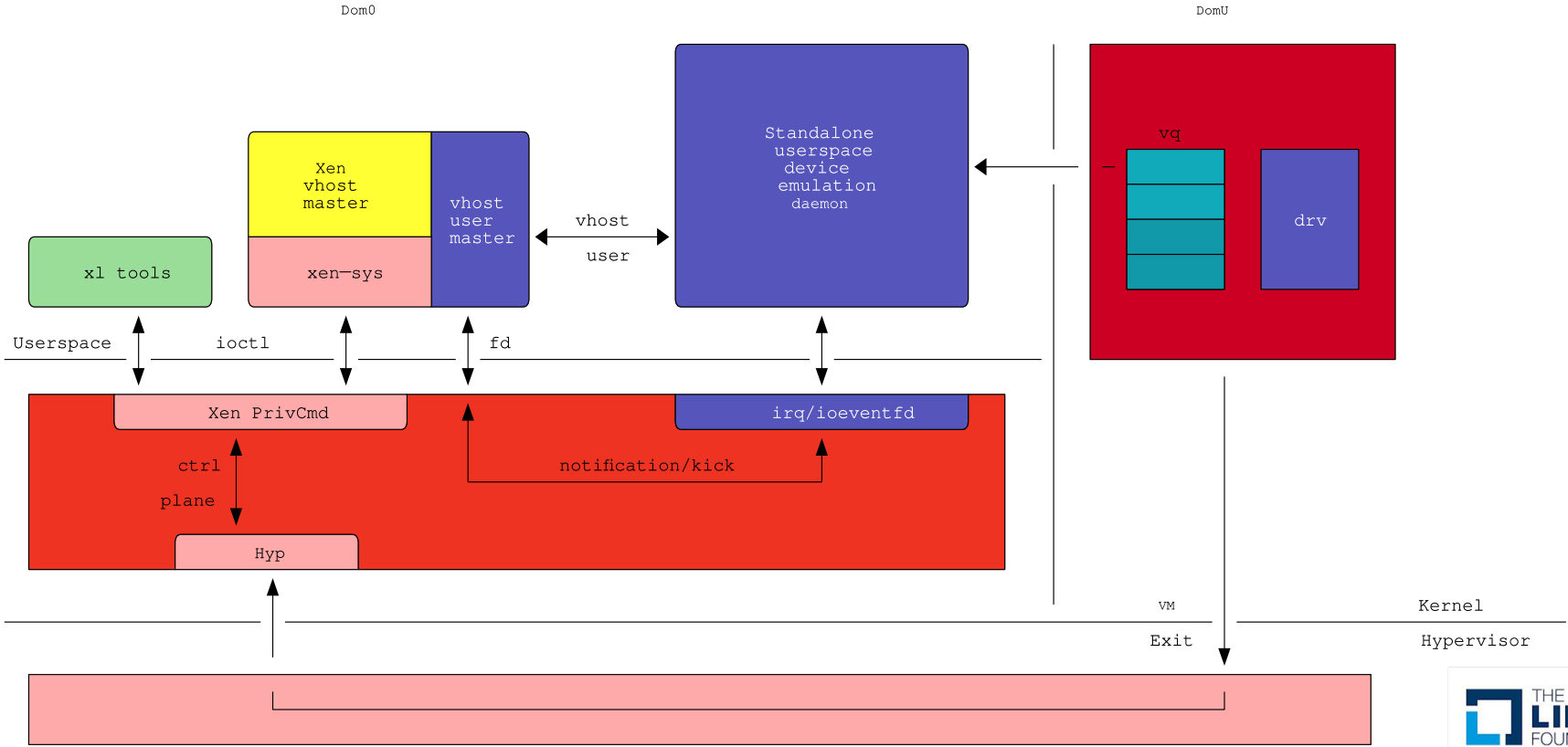
Adapting Xen's Mmap API

- Require mmap() followed by ioctl().
- Rust backends call standard mmap().
- Breaks hypervisor agnosticism.
- Workaround: hacked kernel to perform everything from mmap().

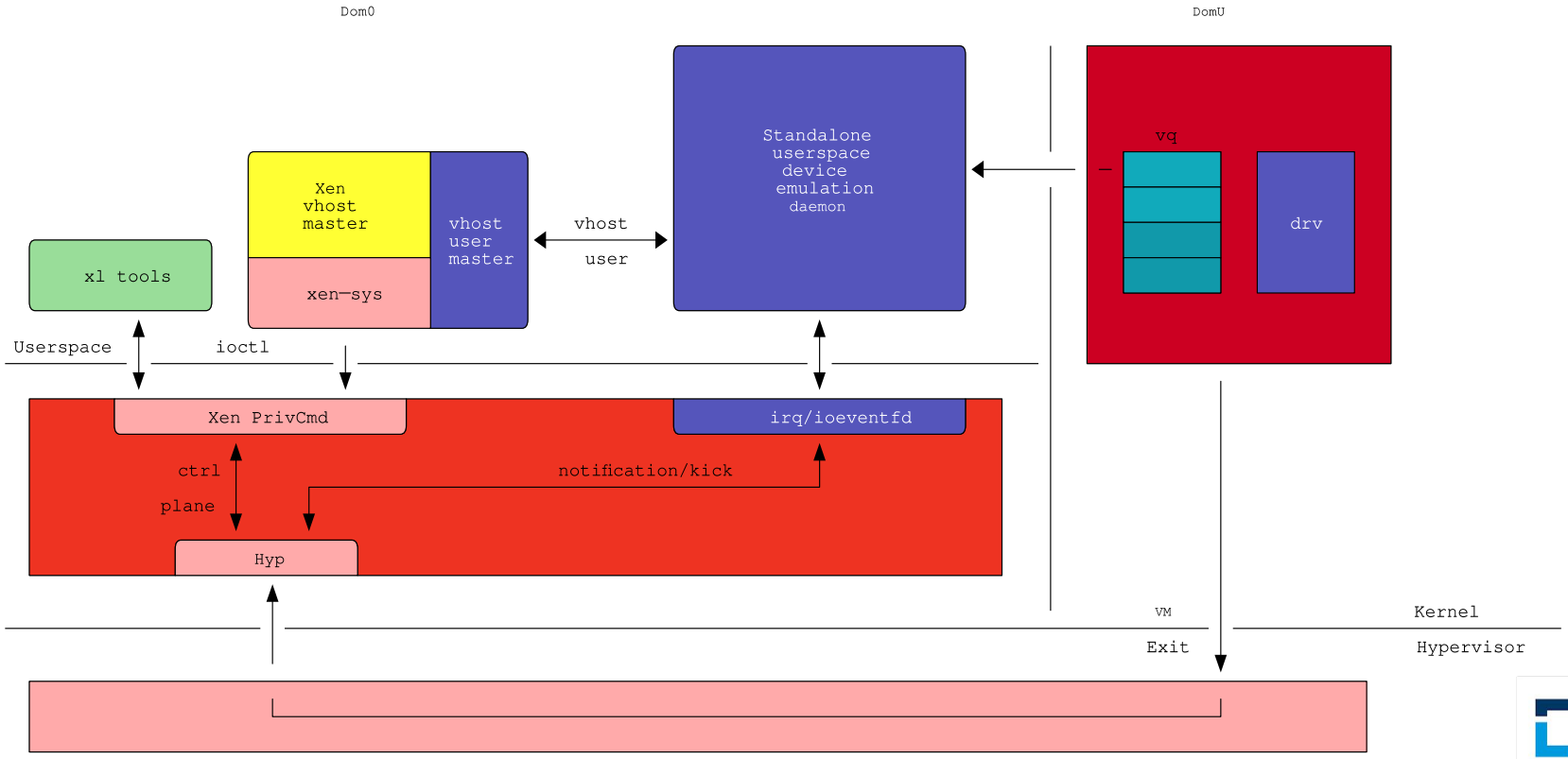
What's next

- Upstreaming
  - vhost-user-master (from Cloud Hypervisor)
  - xen-vhost-master (from Stratos)
- Standard mmap()
- Direct irqfd / eventfd routing
- Guest memory-space privatization

Indirect irqfd / eventfd



Direct irqfd / eventfd



Guest memory-space privatization

- Backend can access entire guest space.
- Limit to virtqueues and buffers.

Approach	**Pro**	**Con**
Per request map (grant / iommu)	Fits existing models	Slow?
Carved out regions (swtlb)	Single mapping	No zero copy, dimensioning
Fat virt queues (data in vq)	Only map virtqueues	API assumptions, low data rate only

Join Us!

- Project Stratos
- Mailing list
- Fortnightly Stratos Sync
- Fortnightly Rust-vmm Sync

KVM FORUM