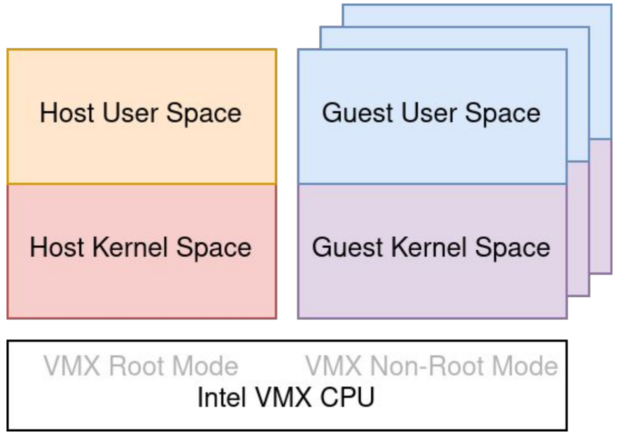
Rethinking IO emulation architectures for VMs

Sandro-Alessio Gierens
Advisor: Peter Okelmann
Chair of Decentralized Systems Engineering
https://dse.in.tum.de/

15.05.2022 – 15.09.2022

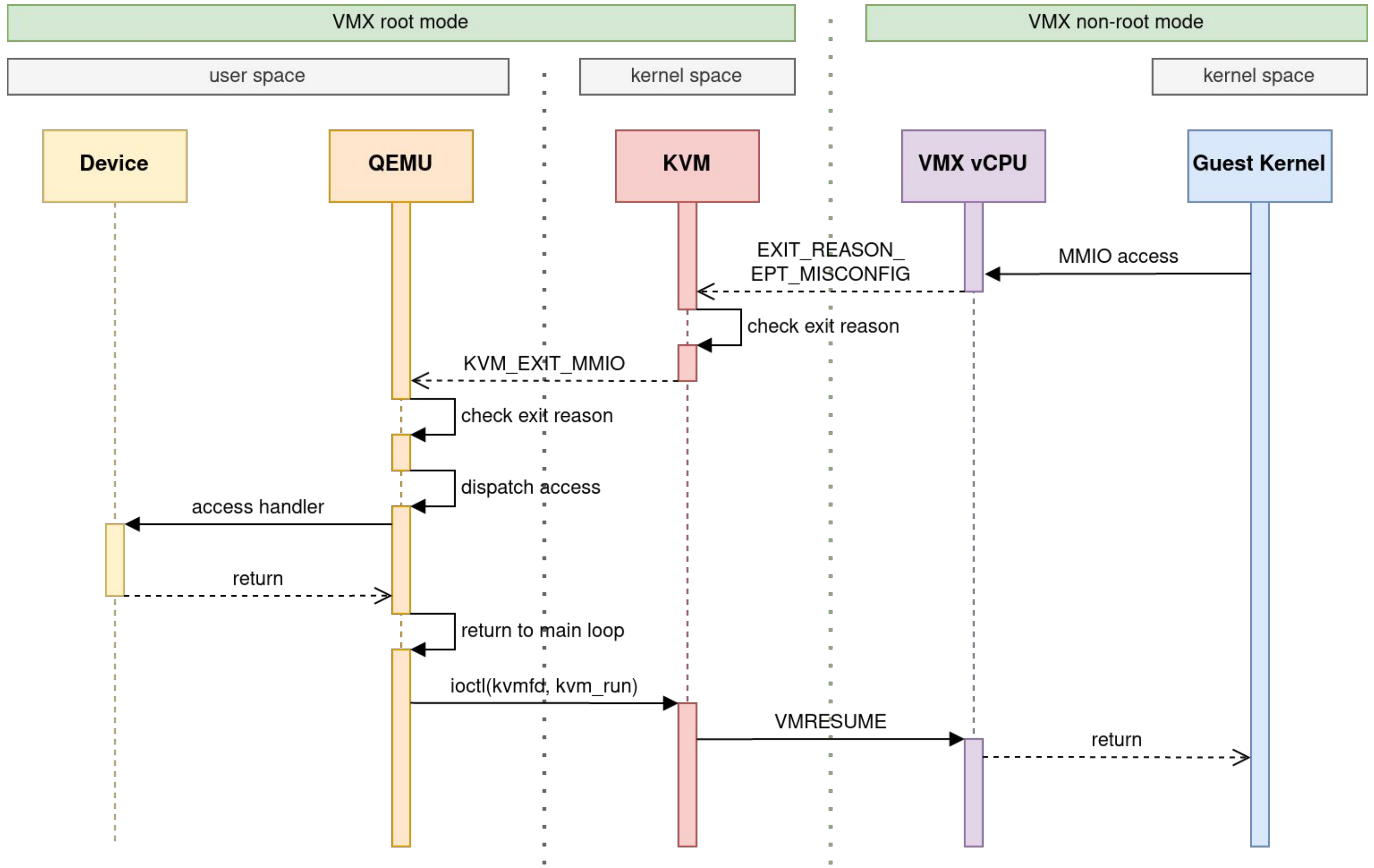
Motivation: I/O in System Virtualization

- System virtualization cornerstone of modern IT
- CPU and memory virtualization efficient due to processor extensions

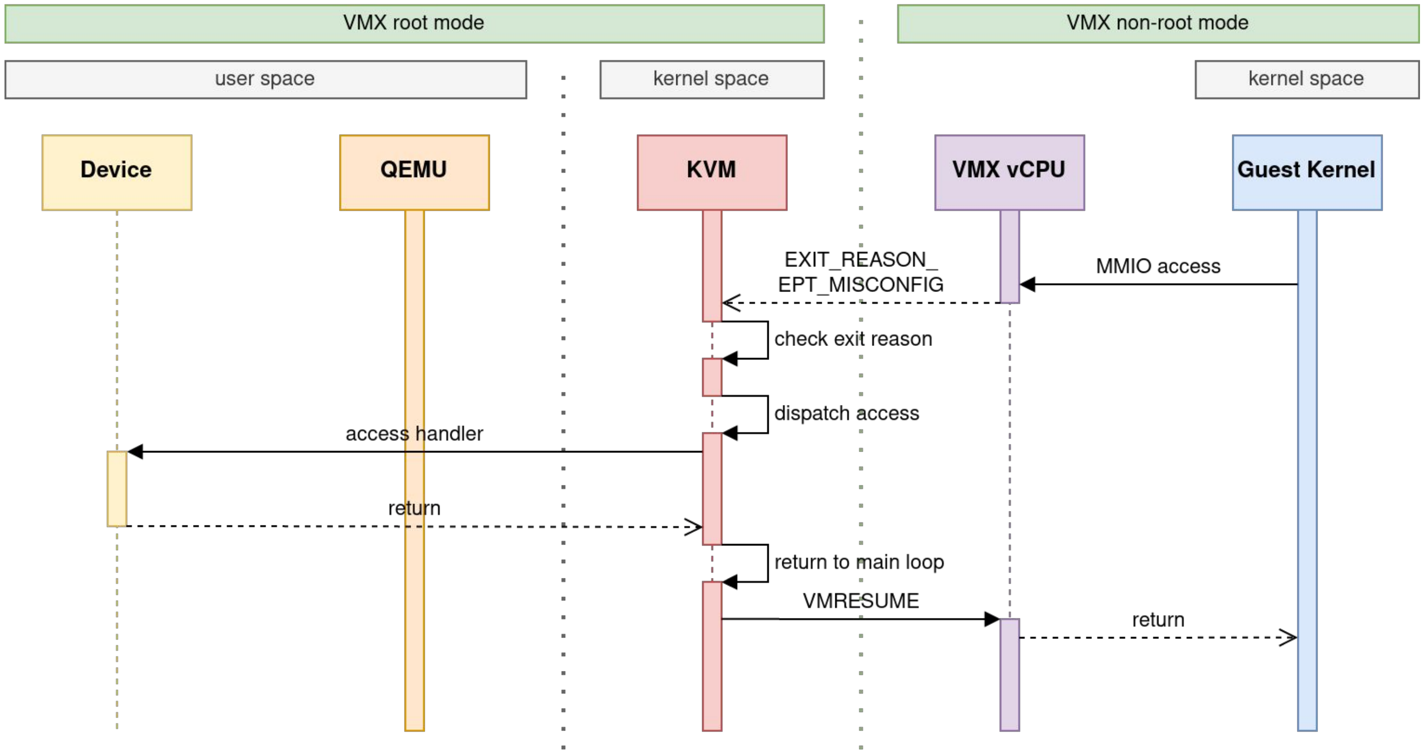


- Similar approaches for I/O expensive and lack flexibility
- full- and para-virtualized I/O still common => performance penalty

State of the art: virtualized MMIO



Recent Proposal: ioregionfd-enhanced MMIO



ioregionfd replaces the context switch by inter-process communication.

Research Gap

- Due to novelty only one application: QEMU remote device
    - counters additional IPC overhead
    - shows promising results
- Research question:

Is ioregionfd a viable solution to improve guest MMIO performance in general?

Research Goals

Implement and evaluate ioregionfd for VirtIO

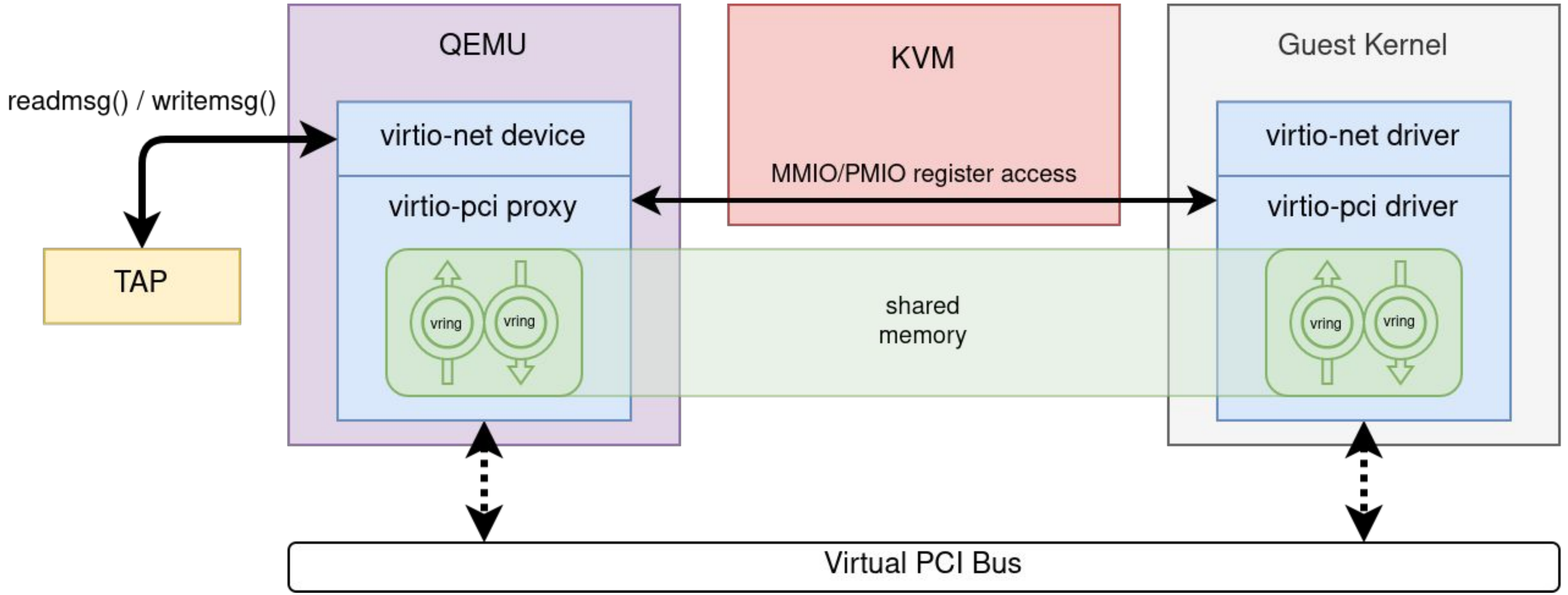
Further system design goals:

- easy to use
- per-device activation
- mostly device-agnostic

Outline

- ~~Motivation~~
- Background
  - VirtIO
- Design
- Implementation
- Evaluation

Background: VirtIO



ioregionfd has to be applied in the VirtIO **frontend / bus proxy.**

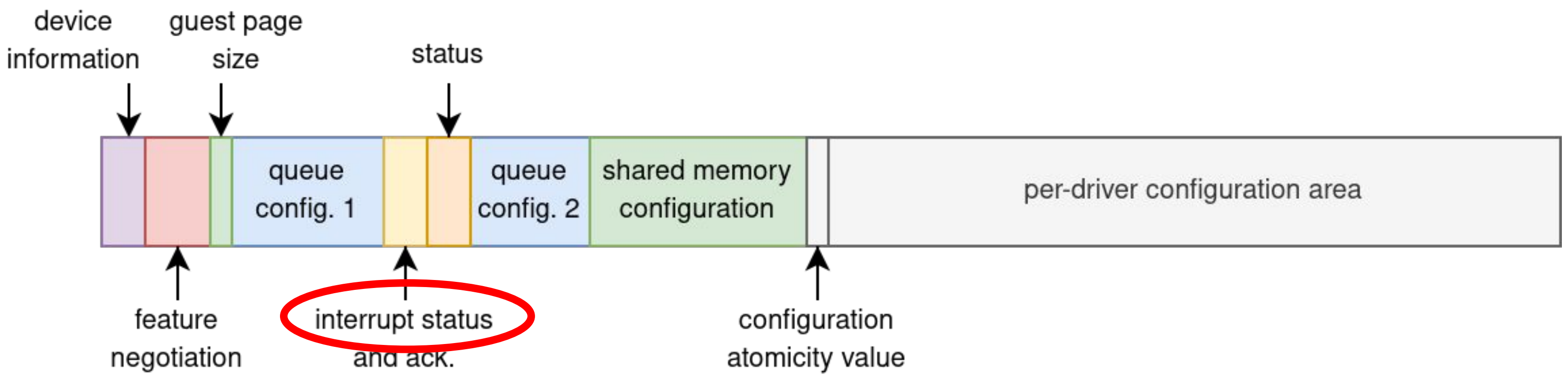
Outline

- ~~Motivation~~
- ~~Background~~
- Design
  - VirtIO-MMIO Registers
  - Device-agnostic Architecture
- Implementation
- Evaluation

Design: VirtIO-MMIO Registers

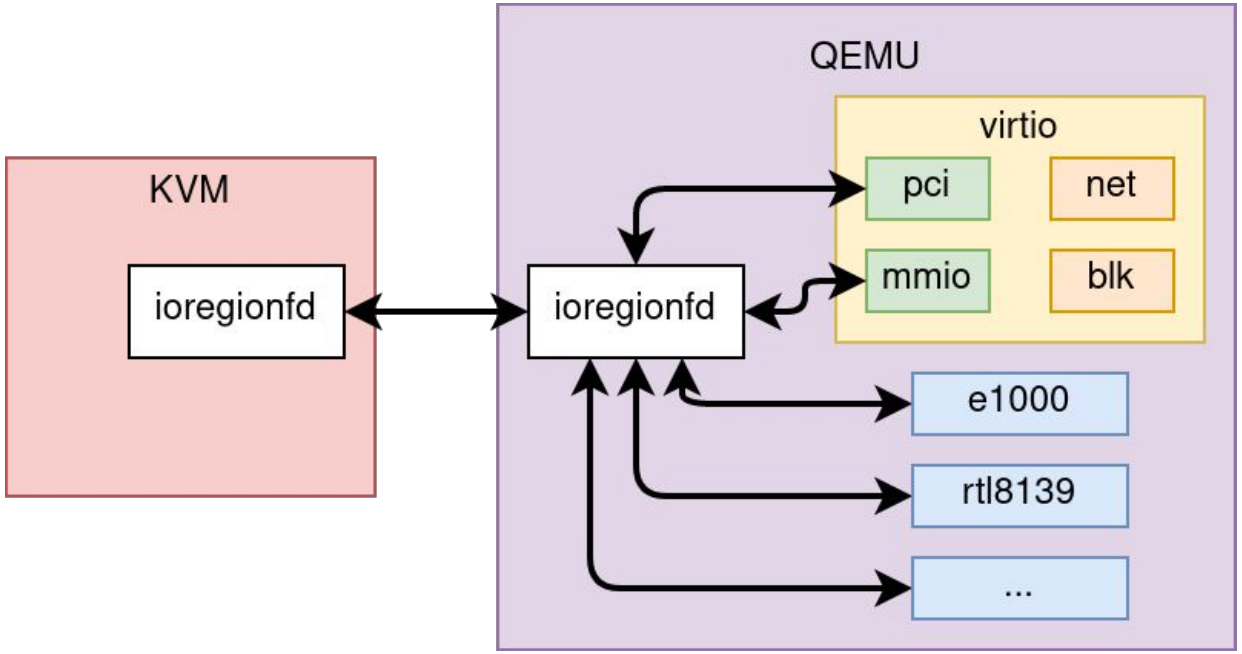
- VirtIO-PCI **no** periodic MMIO accesses
- VirtIO-MMIO has frequent MMIO accesses

We **focus on** the **VirtIO-MMIO** bus, its frequent MMIO accesses promise a bigger impact.


device information
guest page size
status
queue config. 1
queue config. 2
shared memory configuration
per-driver configuration area
feature negotiation
interrupt status and ack.
configuration atomicity value

Design: Device-agnostic Architecture

Many QEMU devices have similar MMIO/PMIO handlers.



Outline

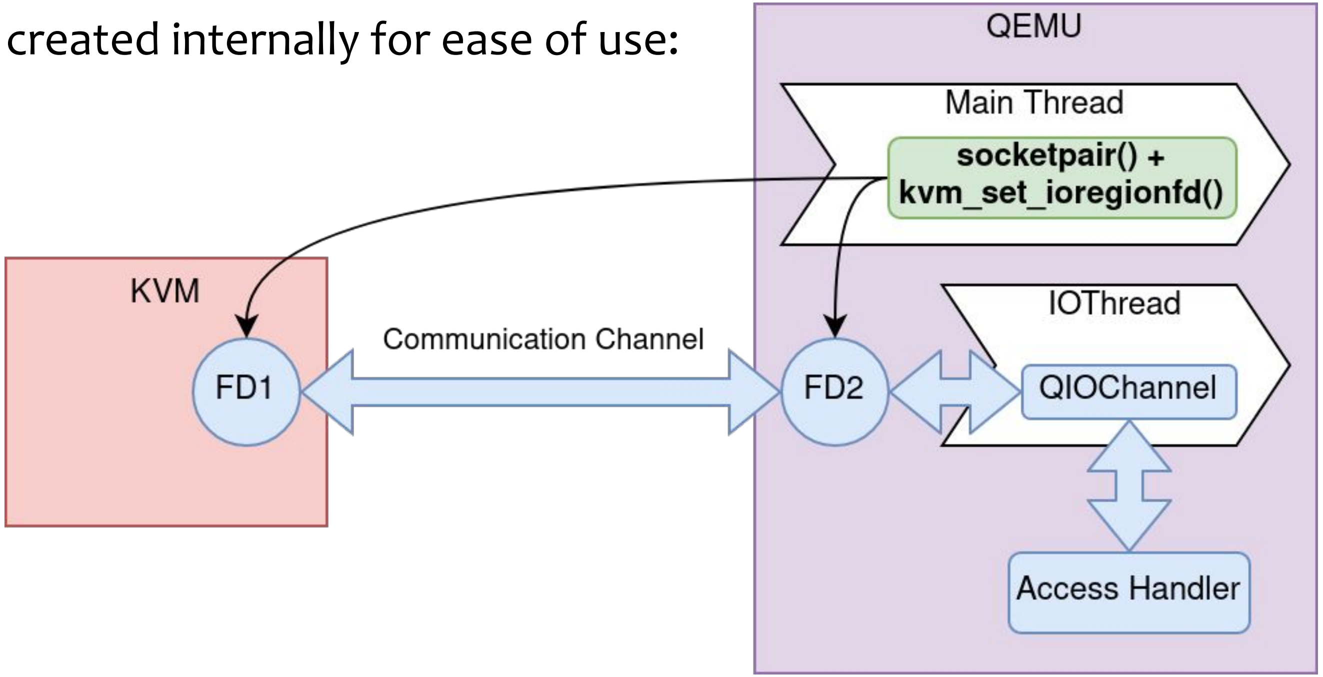
- ~~Motivation~~
- ~~Background~~
- ~~Design~~
- Implementation
    - CLI option and File-Descriptors
    - Generic Handler
- Evaluation

Implementation: CLI option and File-Descriptors

- CLI option for per-device activation:

```
-netdev tap,id=if1,ifname=tap1 -device virtio-net-device,netdev=if1,use-ioregionfd=yes
```

- File-descriptors created internally for ease of use:



Implementation: Generic Handler

- Common signature for MMIO handlers:

```
static uint64_t virtio_mmio_read(void *opaque, hwaddr offset, unsigned size);

static void virtio_mmio_write(void *opaque, hwaddr offset, uint64_t value, unsigned size);
```

- Basic idea: split handler into:
  - Big Device-agnostic core handler
  - Tiny Device-specific proxy handler

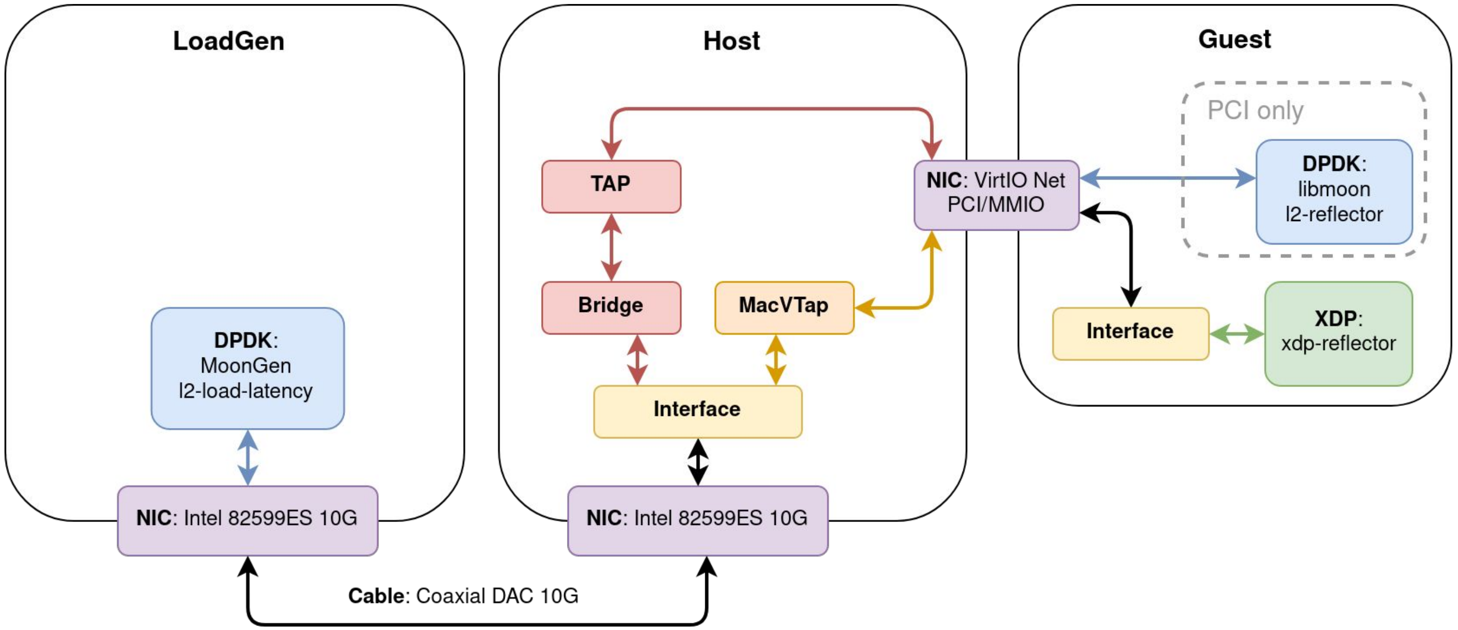
```
static void virtio_mmio_ioregionfd_handler(void *opaque) {

     ioregionfd_qio_channel_read(opaque, virtio_mmio_read, virtio_mmio_write);

}
```

Outline

- ~~Motivation~~
- ~~Background~~
- ~~Design~~
- ~~Implementation~~
- Evaluation
  - Measurement Setup
  - Key Results

Evaluation: Measurement Setup

- Two identical machines (FUJITSU Primergy CX250 S2, 2x Intel Xeon E5-2670 v2, 2x Samsung M393B1K70CH0-CH9 8 GB 1333 MHz ECC DDR3, Samsung 870 EVO SATA SSD)
- MoonGen for latency measurements

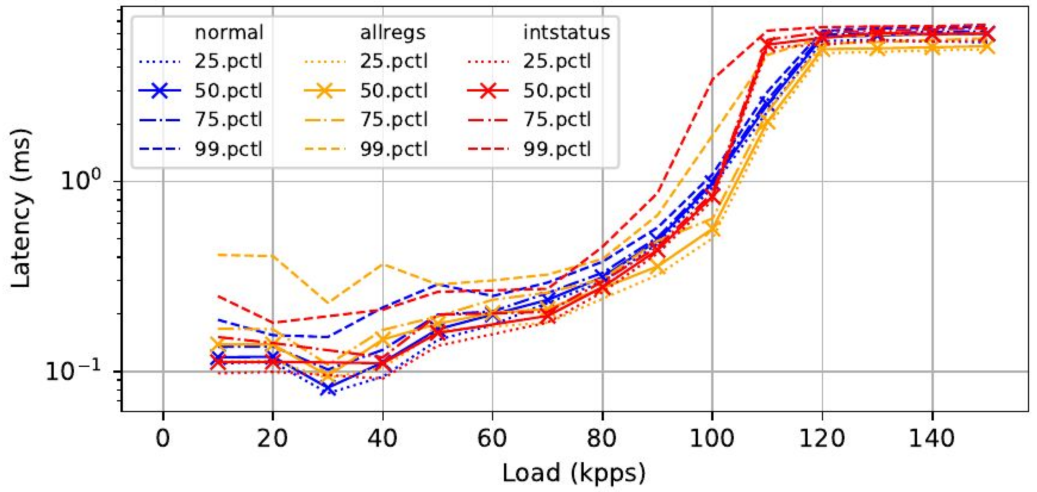


autotest: custom software generating and running test based of configuration file

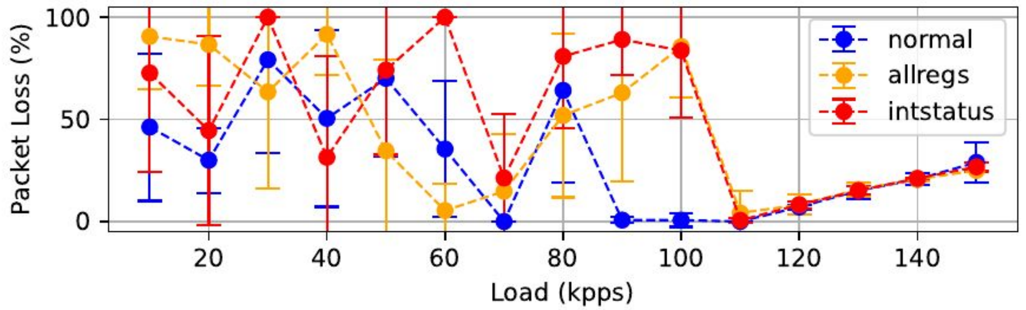
Evaluation: Key Results I

ioregionfd:

- Small changes in latency behaviour



- Trend towards higher packet loss
- High packet loss **before load limit**



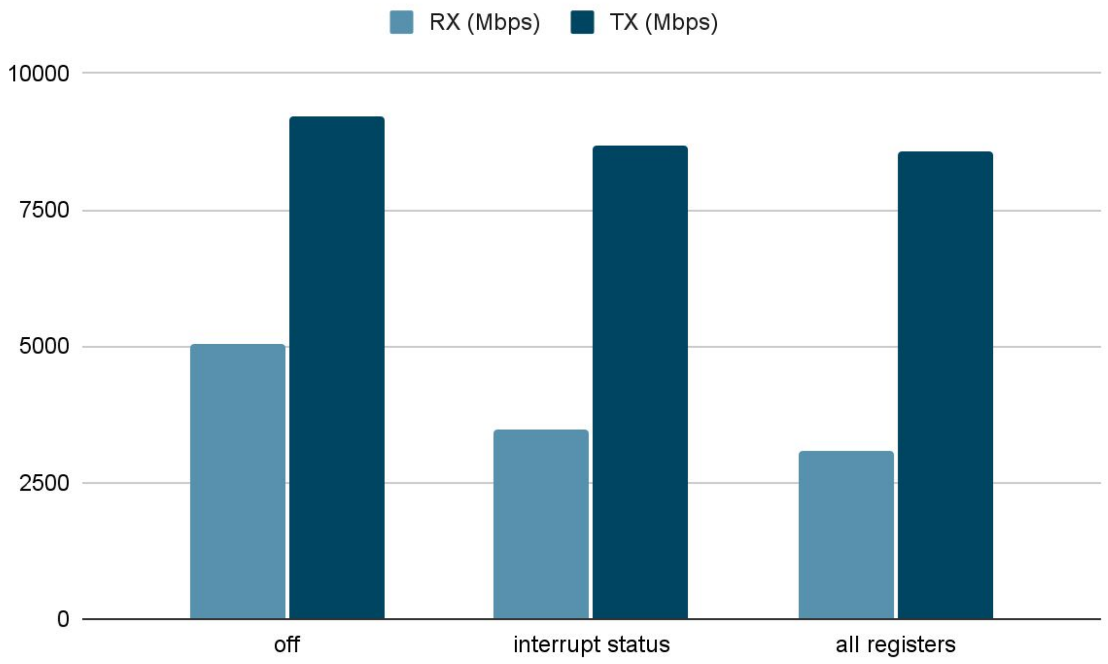
Evaluation: Key Results II

Ruling out other parameters:

- **vhost:** no influence at all
- **packet size:** larger packets widen differences, no influence otherwise
- **interface:** both types behave differently, but show same problems
- **reflector:** lower load limit and higher latency with XDP, but no packet loss on PCI
- **bus:** no unusual packet loss with PCI, but **significant packet** loss before load limit **with MMIO bus**

Evaluation: Key Results III

iPerf3 Throughput:



- **significant drop in throughput** especially in receive performance
- **no packet loss** observed with iPerf3

Evaluation: Key Results III

Other Devices:

- **e1000**
  - connectivity bug prevented any measurements
- **rtl8139**
  - load limit too low for proper MoonGen measurements
- **virtio-blk-device**
  - hardware issue prevented meaningful measurements

Conclusion

Implementation fulfills our requirements

- easy of use, per-device activation and generic nature

No conclusive answer to question about ioregionfd's performance impact yet

- small changes in latency behavior with MoonGen
- but get alarmingly high packet loss on the VirtIO MMIO bus, reason unclear
- trend towards higher packet loss with ioregionfd
- performance drop in throughput with iPerf3

Good basis for future work

- find root cause of VirtIO MMIO bus packet loss
- test more devices (virtio-blk), and parameters (queue size, posted writes)
- upstreaming ioregionfd

Questions?