vDPDK

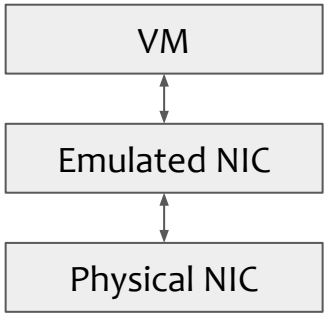
A Para-Virtualized DPDK Device Model for vMux

Dominik Kreutzer
Advisors: Peter Okelmann, Masanori Misono
Systems Research Group
https://dse.in.tum.de/

2024-11-18 – 2025-05-19

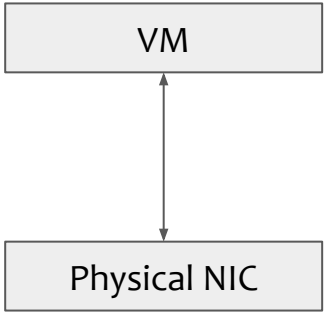
Virtual machine networking

- Networking typically via device emulation or passthrough



Emulation:

- Scalable
- Limited feature-set

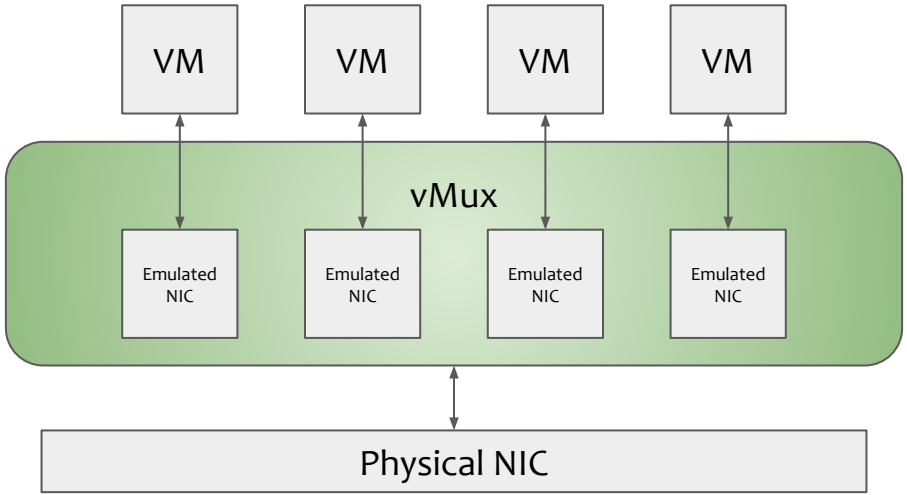


Passthrough:

- Full feature-set
- Not scalable

vMux

- Implementation of NIC virtualization outside of hypervisor
- DPDK backend to access host NIC
- Supports mediation of offloads



vMux problems

vMux is flexible and scalable, but:

- Emulated devices are based on real hardware
- Low performance due to emulation overhead
- Requires uniform feature-set between virtual and physical NIC:

Feature availability		
Physical NIC	Emulated NIC	Consequence
✔	✔	Feature can be used
✔	✘	Feature is not available to VM
✘	✔	Feature must be emulated with performance penalty

Problem statement

Can we pass NIC capabilities into VMs without compromising performance and scalability?

Proposal: vDPDK

A para-virtualized vMux device for guest DPDK applications

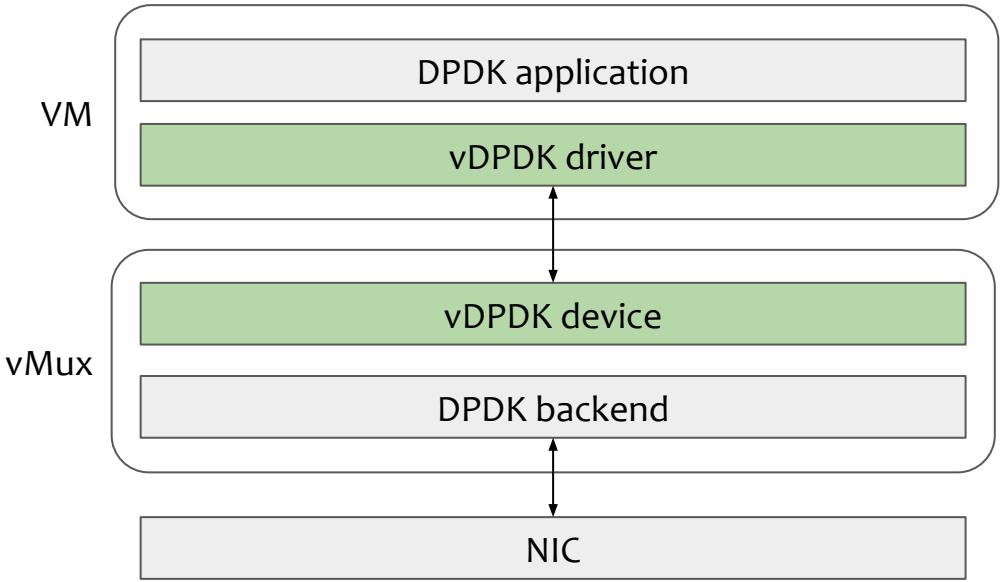
Features:

- Run-time discovery of host NIC capabilities
- Low emulation overhead
- Scale well to a high number of VMs

Outline

- ~~Motivation~~
- Design
  - Overview
  - Challenges
  - Key ideas
- Implementation
- Evaluation

Design overview



Design challenges

Generality

Run-time discovery of host NIC capabilities.

Re-use vendor-independent DPDK API

Performance

High throughput with low latency.

Fast-path for packet TX/RX

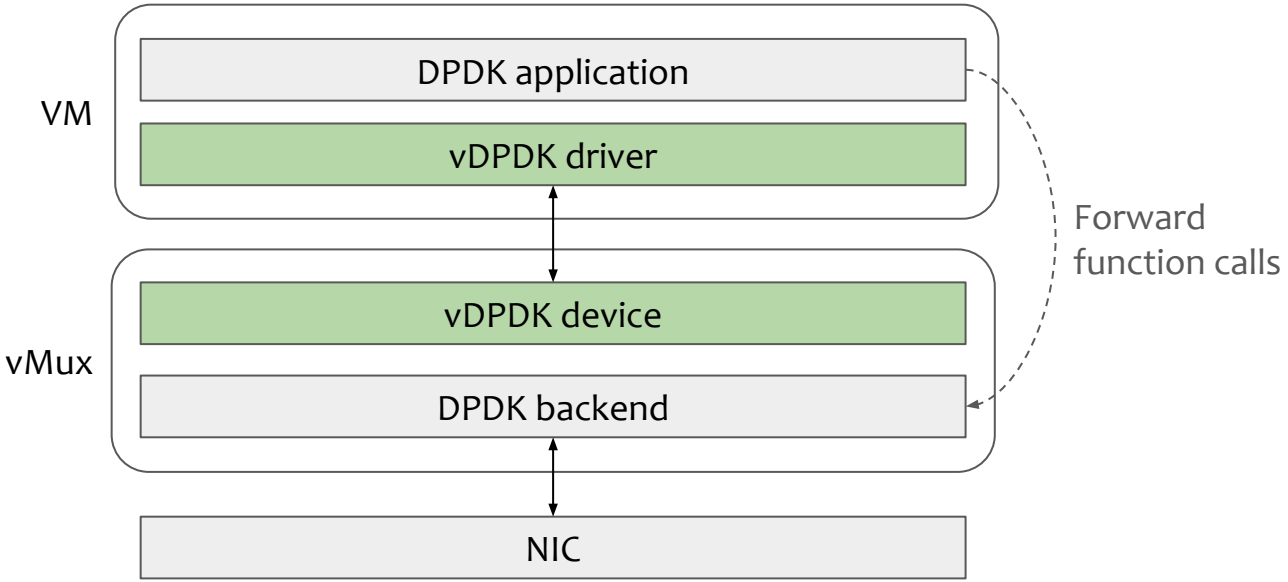
Scalability

Scale to high number of VMs.

Limit concurrent busy-polling

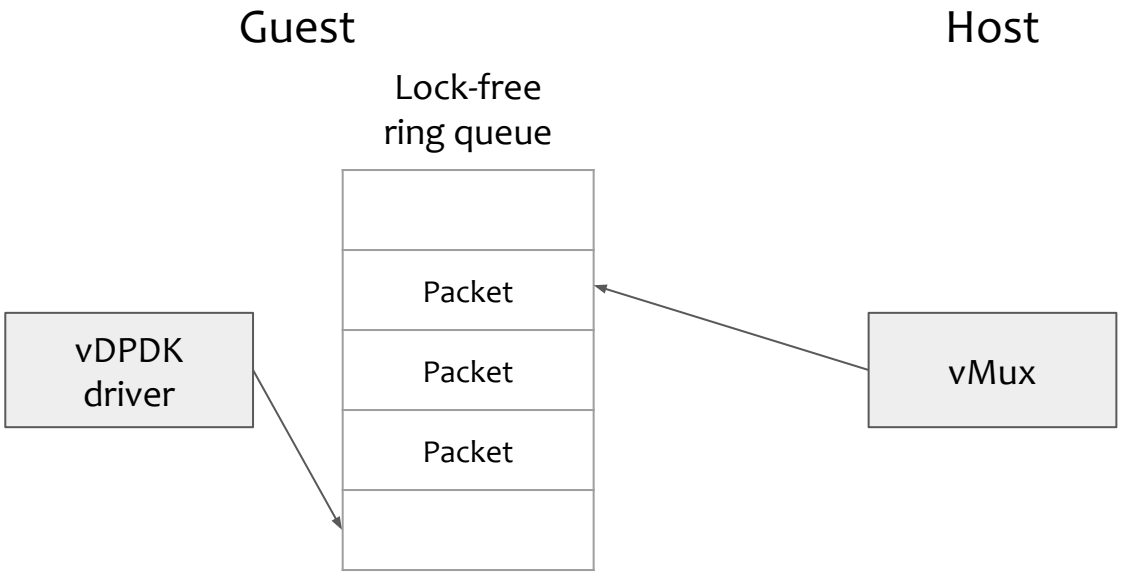
Generality

Key idea: re-use DPDK API by forwarding guest function calls to host backend



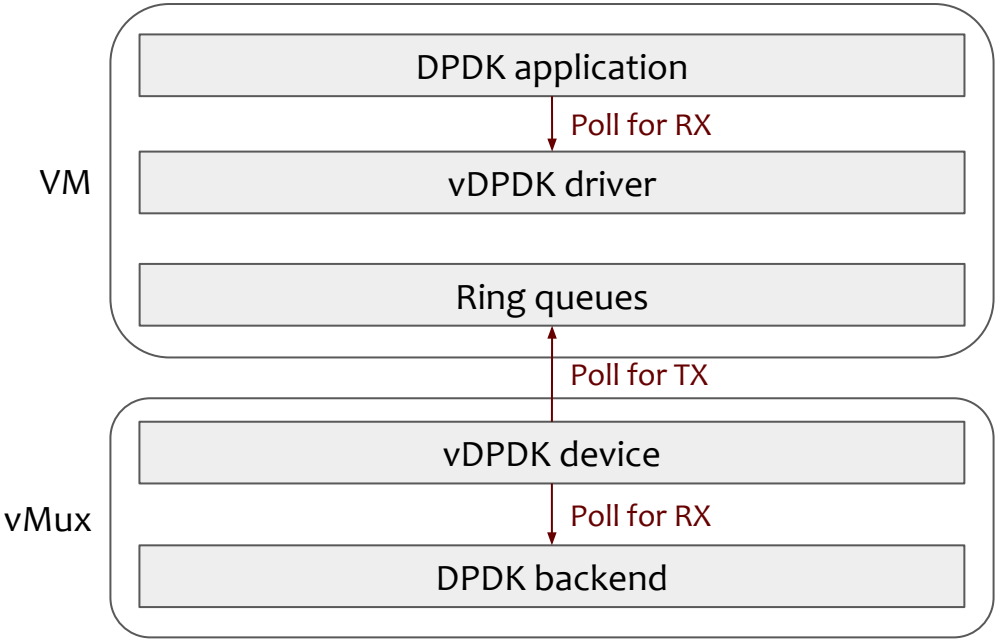
Performance

Key idea: asynchronous and batchable protocol for packet TX/RX



Scalability

Key idea: limit concurrent polling in vMux with scheduling



Outline

- ~~Motivation~~
- ~~Design~~
- Implementation
- Evaluation

Implementation

- Virtual device setup via **vfio-user** protocol
- Shared memory via **PCI BAR** memory regions
- Packet data access via **DMA**
- Guest signalling via interrupts

Outline

- ~~Motivation~~
- ~~Design~~
- ~~Implementation~~
- Evaluation

Evaluation

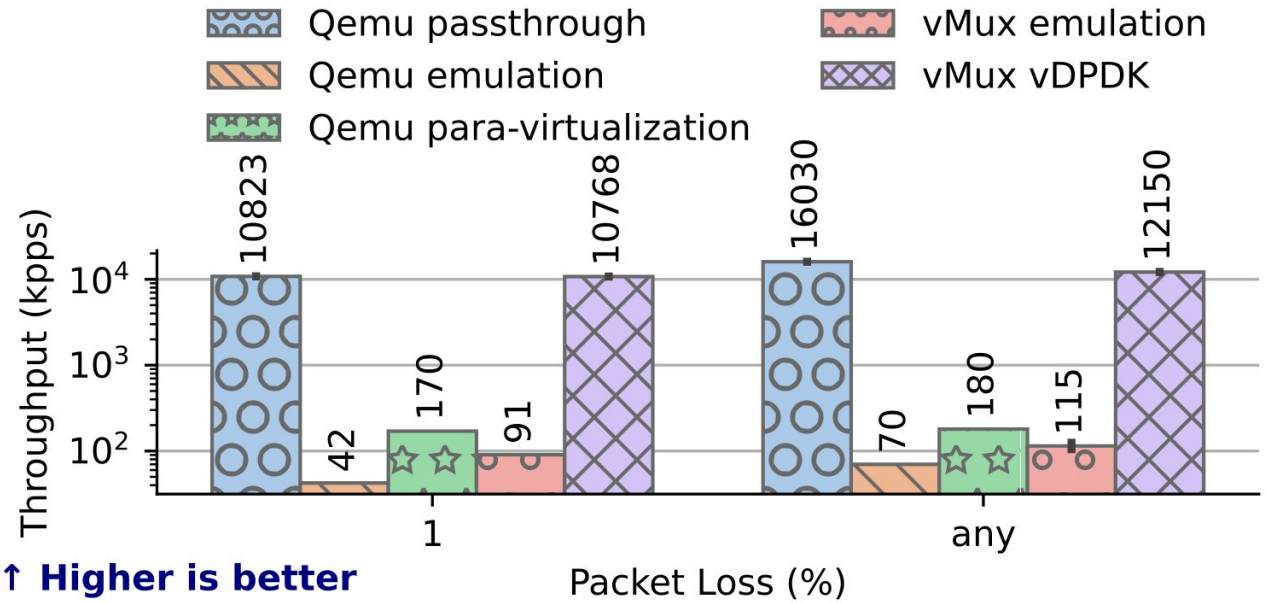
- DPDK benchmarks
  - Throughput and latency
  - Offloaded packet classification
- Non-DPDK benchmarks
  - TCP throughput
  - Cloud serving
  - Microservices

Evaluation

- DPDK benchmarks
  - Throughput ~~and latency~~ ← Performance
  - ~~Offloaded packet classification~~
- Non-DPDK benchmarks
  - ~~TCP throughput~~
  - Cloud serving ← Scalability
  - ~~Microservices~~

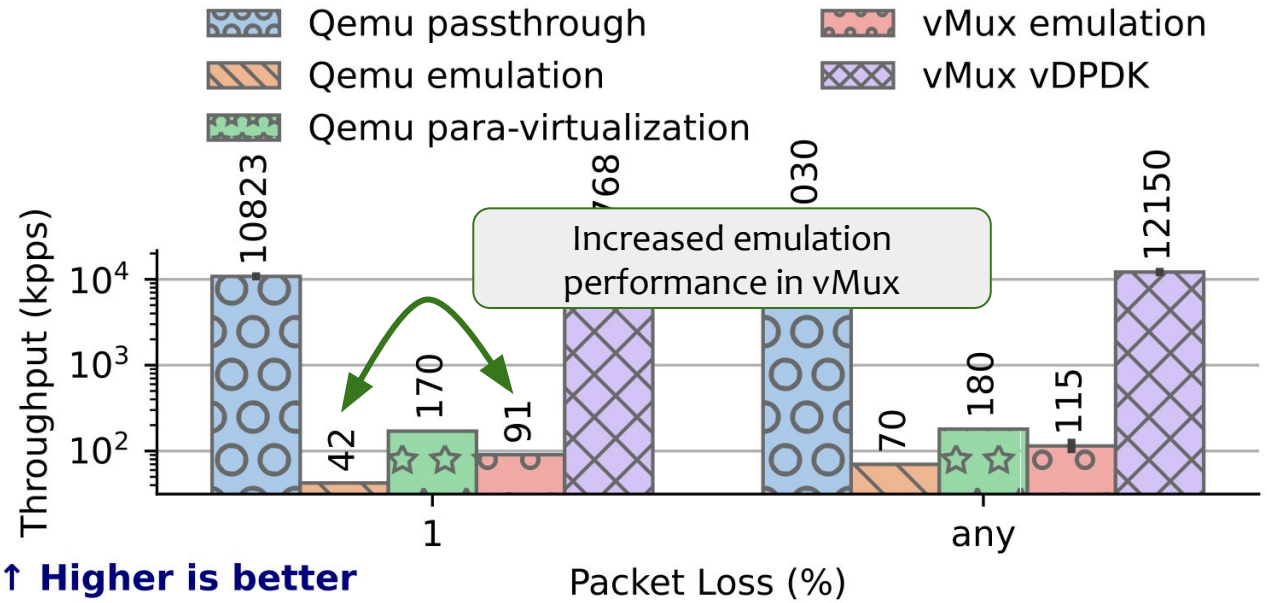
Evaluation: Throughput

- Setup: external load generator sends packets to reflector running on VM



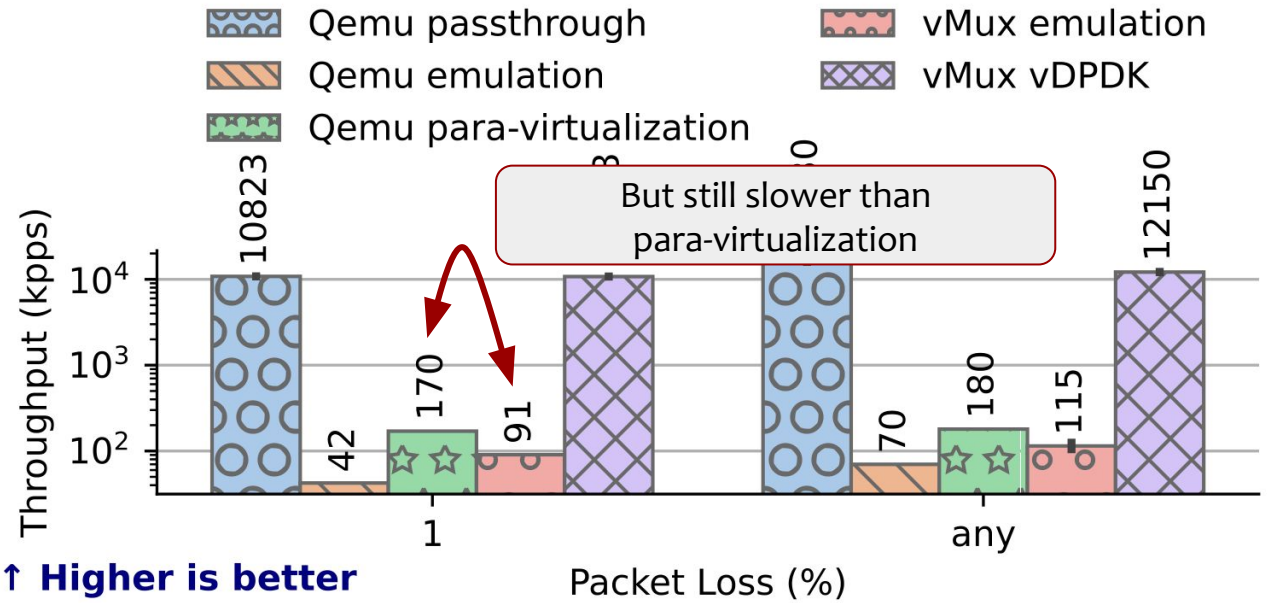
Evaluation: Throughput

- Setup: external load generator sends packets to reflector running on VM



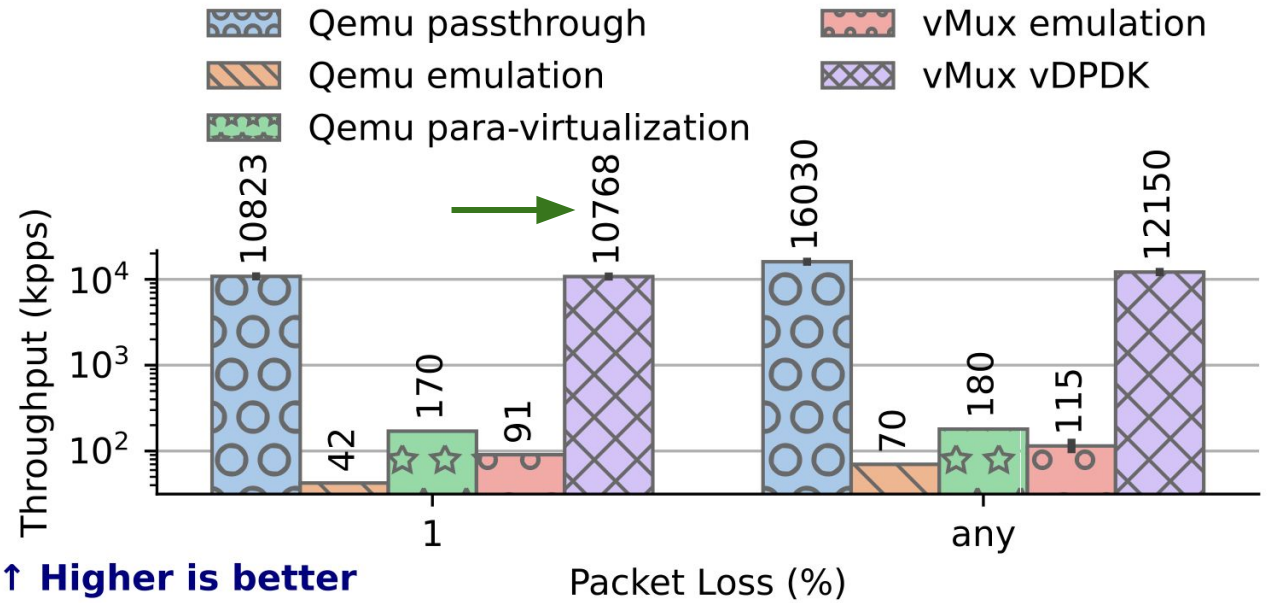
Evaluation: Throughput

- Setup: external load generator sends packets to reflector running on VM



Evaluation: Throughput

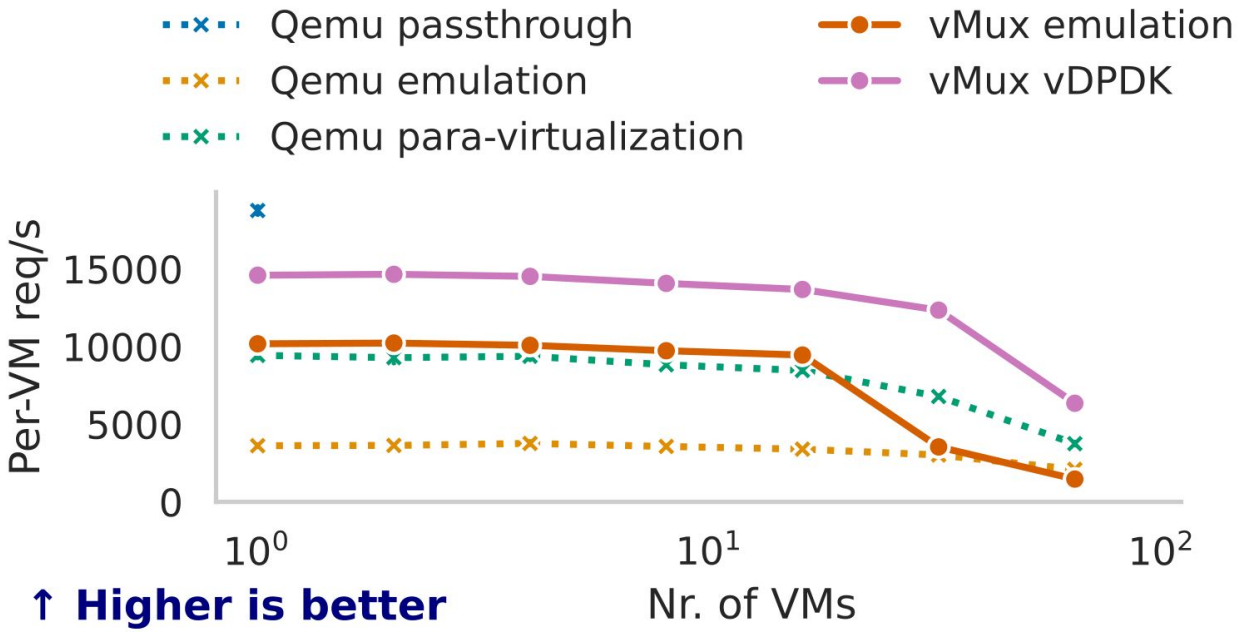
- Setup: external load generator sends packets to reflector running on VM



vMux vDPDK: close to passthrough performance

Evaluation: Scalability

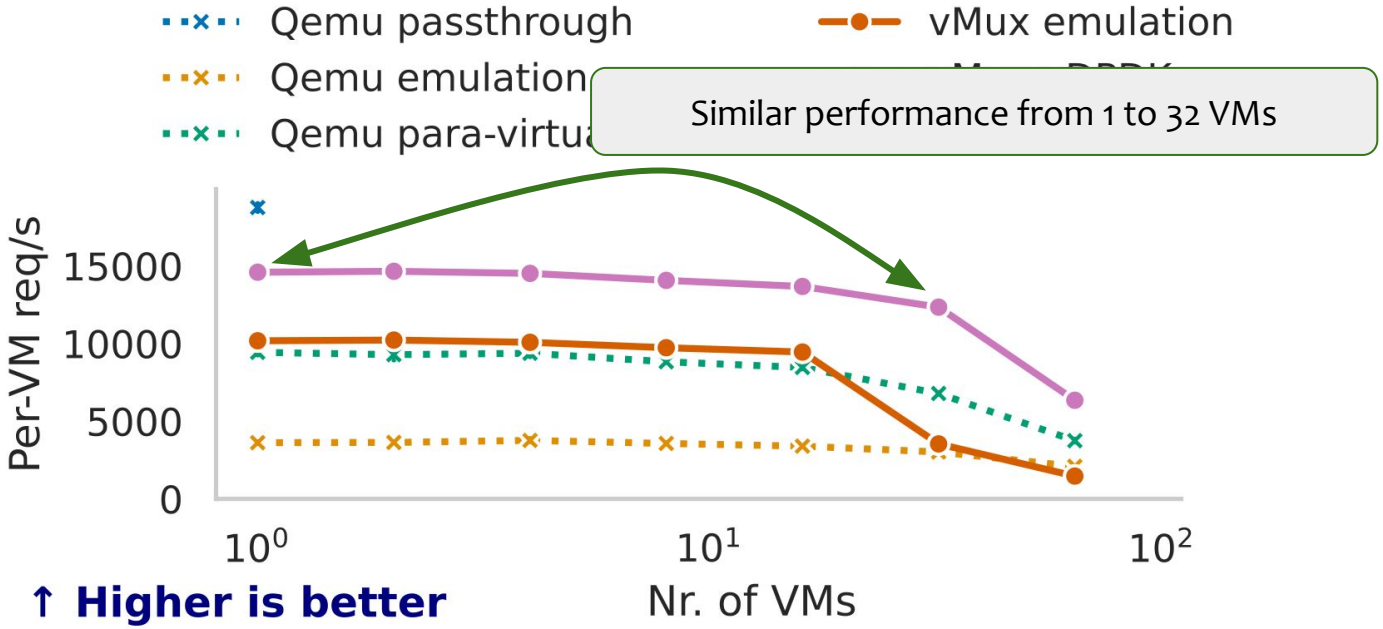
- Setup: Redis-based YCSB[1] benchmark, VMs send requests to external server



[1] Yahoo! Cloud Serving Benchmark: https://github.com/brianfrankcooper/YCSB

Evaluation: Scalability

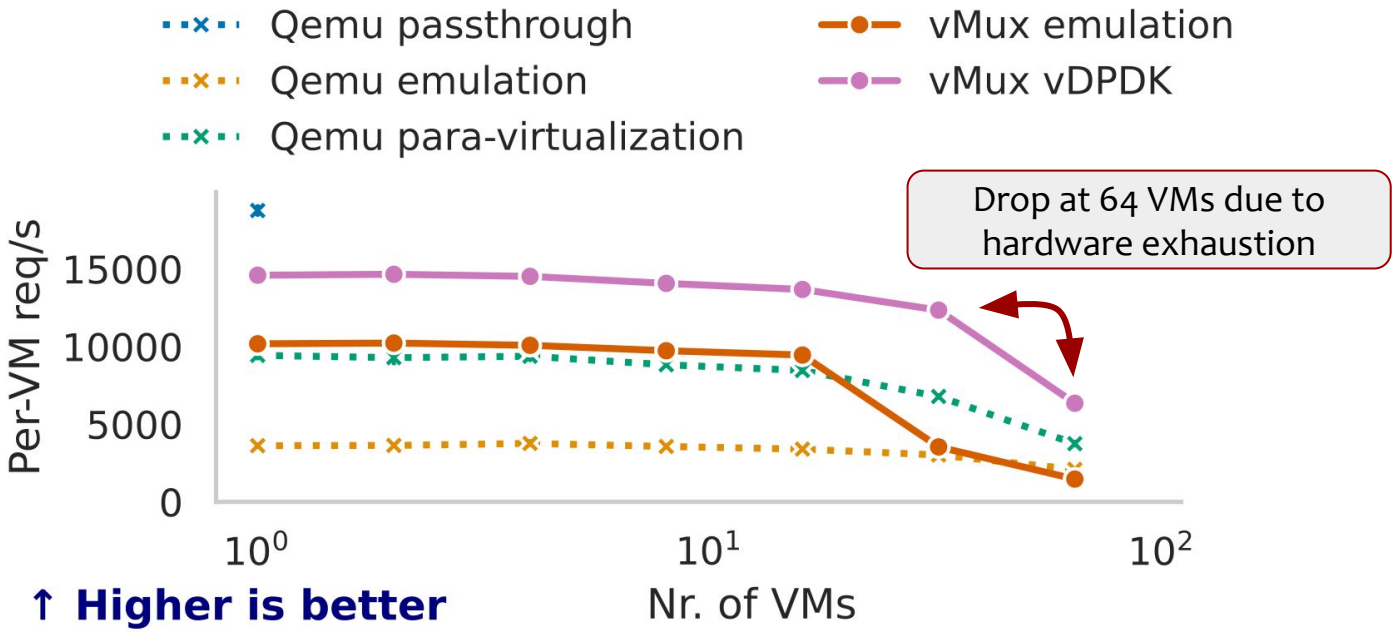
- Setup: Redis-based YCSB[1] benchmark, VMs send requests to external server



[1] Yahoo! Cloud Serving Benchmark: https://github.com/brianfrankcooper/YCSB

Evaluation: Scalability

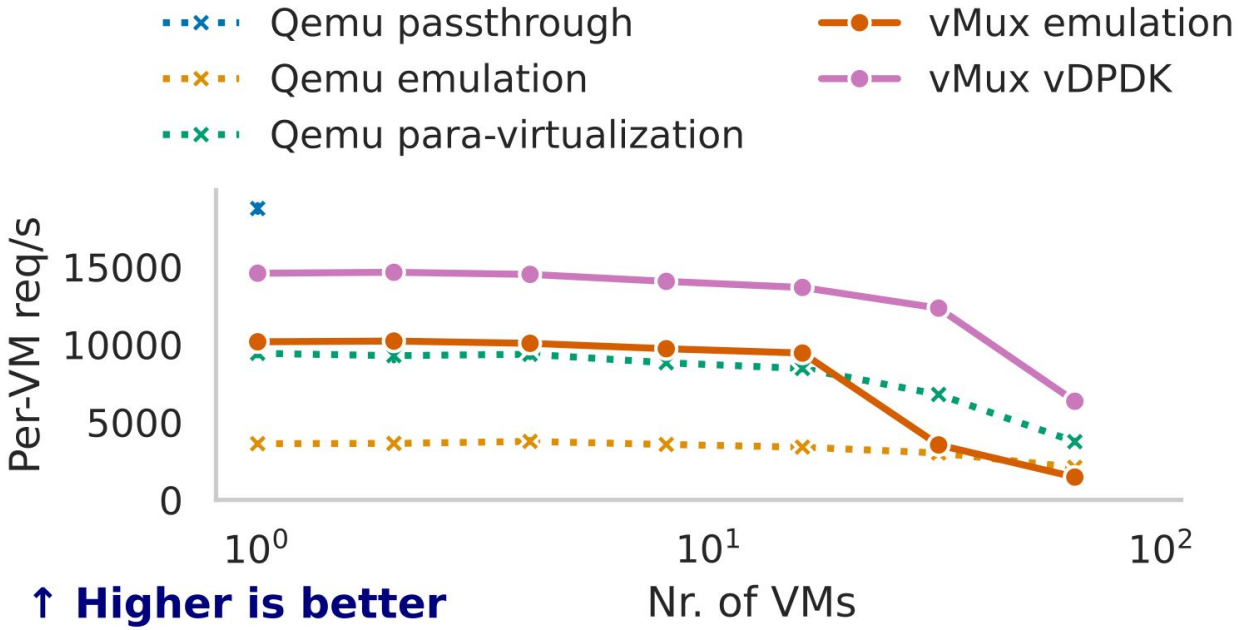
- Setup: Redis-based YCSB[1] benchmark, VMs send requests to external server



[1] Yahoo! Cloud Serving Benchmark: https://github.com/brianfrankcooper/YCSB

Evaluation: Scalability

- Setup: Redis-based YCSB[1] benchmark, VMs send requests to external server



vMux vDPDK: maintains high performance when scaling

[1] Yahoo! Cloud Serving Benchmark: https://github.com/brianfrankcooper/YCSB

Conclusion

Can we pass NIC capabilities into VMs without compromising performance and scalability? ⇒ Yes!

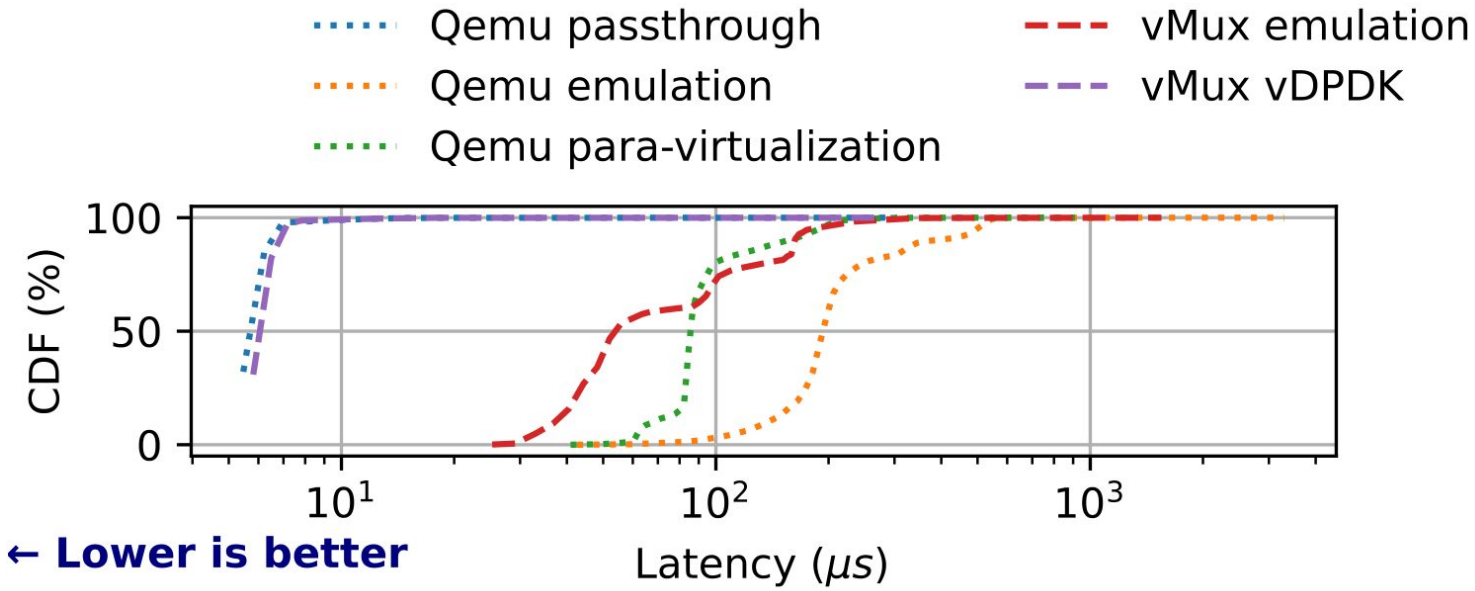
vDPDK:

- Generality: Forwarding of vendor-neutral DPDK API to vMux backend
  - Detect and use NIC capabilities at runtime
- Performance: Fast ring queues for TX/RX
- Scalability: Built-in multi-VM scheduling

Backup

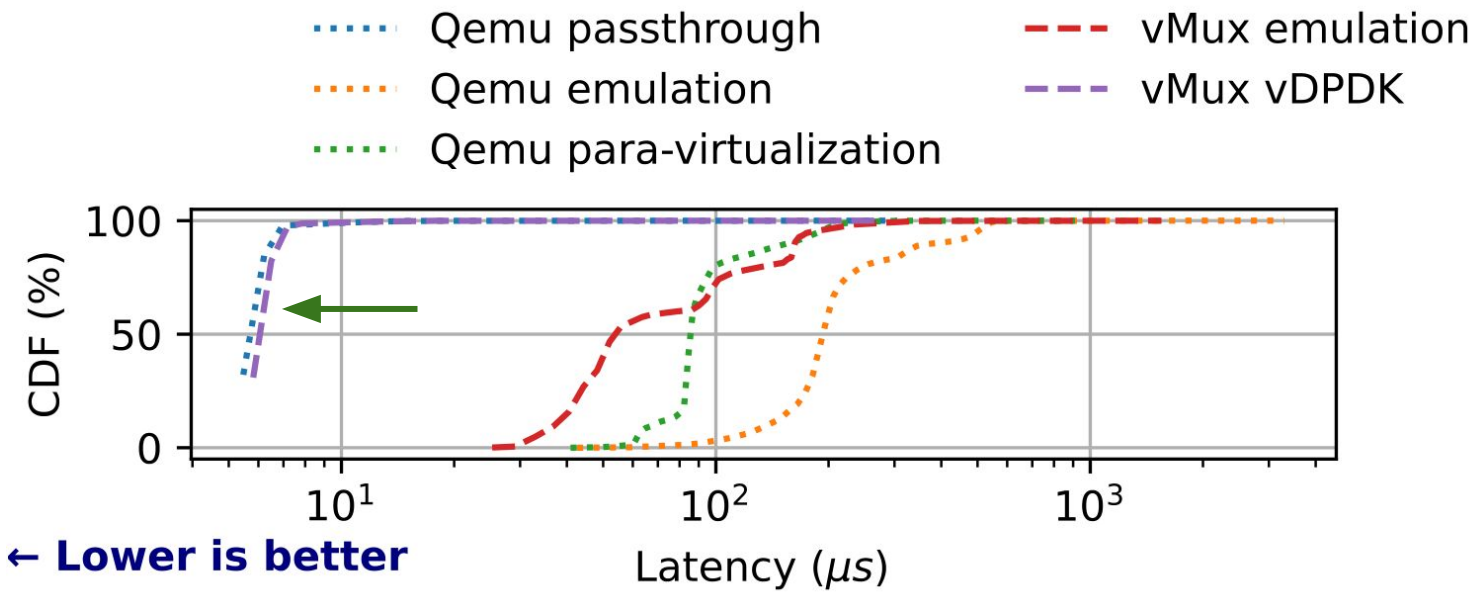
Evaluation: Latency

- Setup: external load generator sends packets to reflector running on VM



Evaluation: Latency

- Setup: external load generator sends packets to reflector running on VM



vMux vDPDK: consistent and low latency

Generality

Key idea: forwarding of guest DPDK function calls to host DPDK backend