Nic-Emu

Hyper-scalability of Network Interface Cards for Virtual Machines

Florian Dominik Freudiger
Advisor: Peter Okelmann
Chair of Computer Systems
https://dse.in.tum.de/

15.07.2023 – 15.11.2023

Motivation: Firecracker

- Virtual machine monitor (VMM) managing microVMs

- Scalability:
  - Thousands of virtual machines may run on the same machine
  - Hundreds of VMs may be started within seconds

- Virtual machines need fast networking

https://firecracker-microvm.github.io/

Existing networking solutions

- NIC passthrough:
  - High performance
  - Limited scalability

- NIC emulation:
  - Limited performance
  - High scalability

⇒ Hybrid solution needed

Hybrid approach

- Switch between networking methods dynamically
  - Re-assign VMs by introducing multiplexer in-between

⇒ vMux
  - Multiplexing in userspace
  - NIC Passthrough via Virtual Function I/O (VFIO) framework
  - NIC Emulation via behavioral models

- Previous behavioral models are not easily controllable

Extensible and modular NIC behavioral model needed

Nic-emu

Userspace NIC emulation

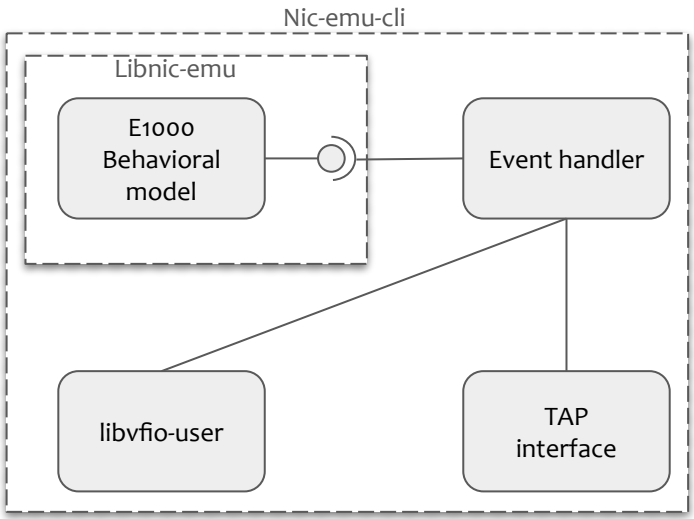
System design goals:

- Driver support
- Modularity
- Performance

System overview

- Library: Libnic-emu
  - Behavioral model: Intel E1000
    - Simple real NIC, cannot use paravirtualized NIC for multiplexing with real NICs
    - QEMU Reference implementation

- Standalone: Nic-emu-cli
  - Command-line-interface
  - Hypervisor integration: libvfio-user
    - PCI device implementation framework
    - Supports QEMU
  - Send and receive using TAP interface



Implementation

- Written in Rust
  - Since libvfio-user only exposes C headers, libvfio-user-rs wrapper has been created
- Behavioral model
  - 23/130 registers emulated
  - VLAN control, debug and statistical registers omitted
- Libnic-emu
  - Available as static library for C/C++ and as Rust crate
- Nic-emu-cli
  - Can process libvfio-user commands, packets and timers in any order using linux I/O multiplexing

Evaluation

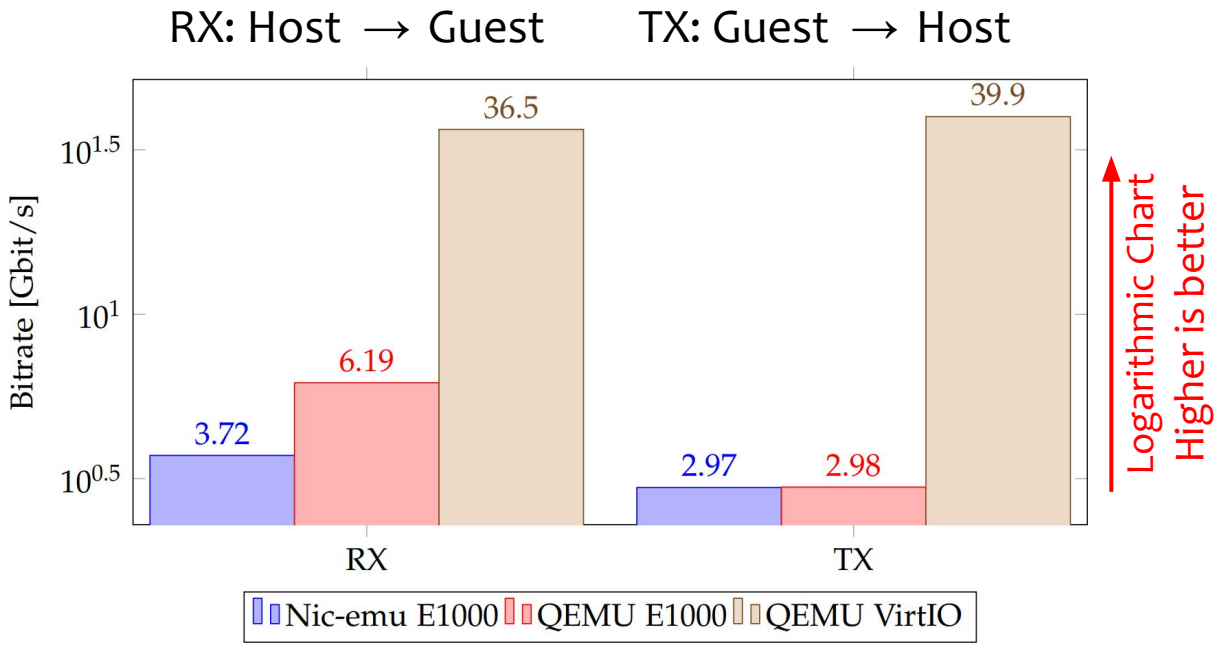
- What is the performance overhead of moving emulation into userspace?
  - Comparing QEMU reference NIC emulation

- What is the performance overhead of not using paravirtualization?
  - Comparing both E1000 NICs of QEMU and Nic-emu to QEMU VirtIO

Evaluation

- Experimental setup (all hosts):
  - Intel Xeon Gold 5317 CPU (3.00 GHz, 12 cores)
  - 256GB DDR4 DRAM @ 2933 MT/s
- Bandwidth measurement
  - Host ⇔ Guest system
  - Via iPerf3 TCP transmission
- Latency measurement
  - Dedicated load-generator ⇔ VM Host ⇔ VM
  - Via Moongen packet generator & XDP reflector

Bandwidth Test



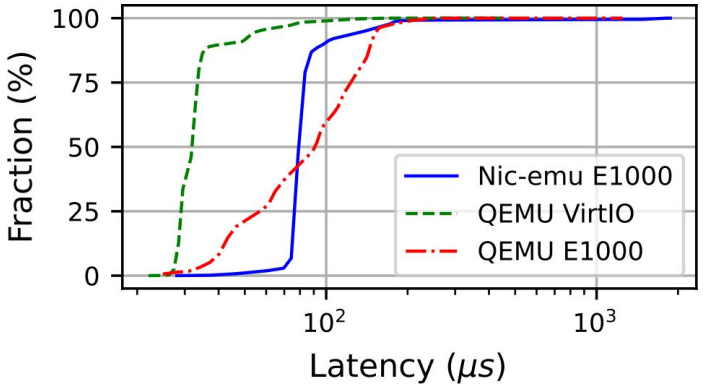
Nic-emu only slower than QEMU in RX

VirtIO greatly surpasses both E1000 implementations

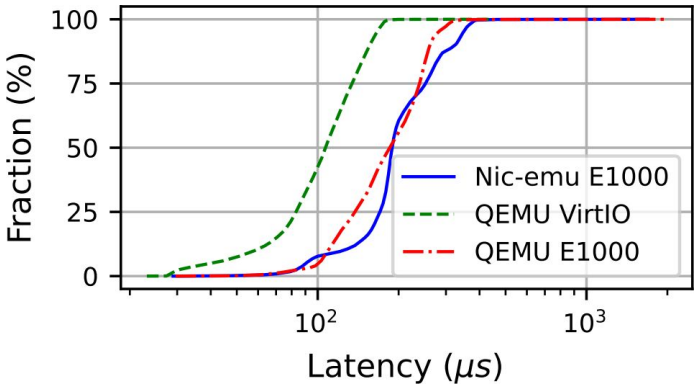
Latency test

60B packets at

Low load: 10kpps



High load: 100kpps



Lower is better

Nic-emu and QEMU E1000 perform similar on average

VirtIO remains faster

Summary

- Emulated E1000 NIC of Nic-emu performs very similar to its counterpart in QEMU in most cases
  - ⇒ Overhead of moving NIC emulation into userspace is minor

- Para-virtualization remains much faster
  - ⇒ Overhead of emulating real NICs needs to be overcome

Try it out!

https://github.com/vmuxIO/nic-emu