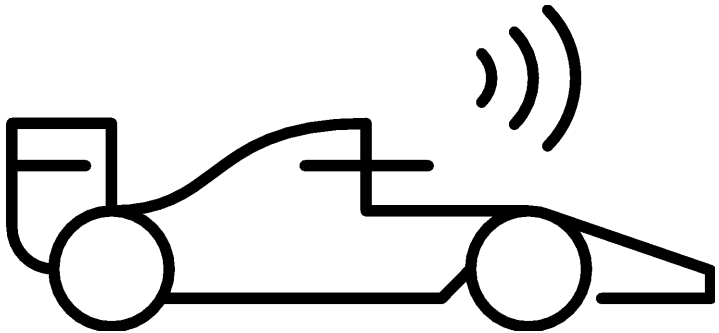
Advancements for seL4 Virtualization Support
CAmkES and ~~seL4cp~~ Microkit

Markku Ahvenjärvi
seL4 Summit 2023, Minneapolis

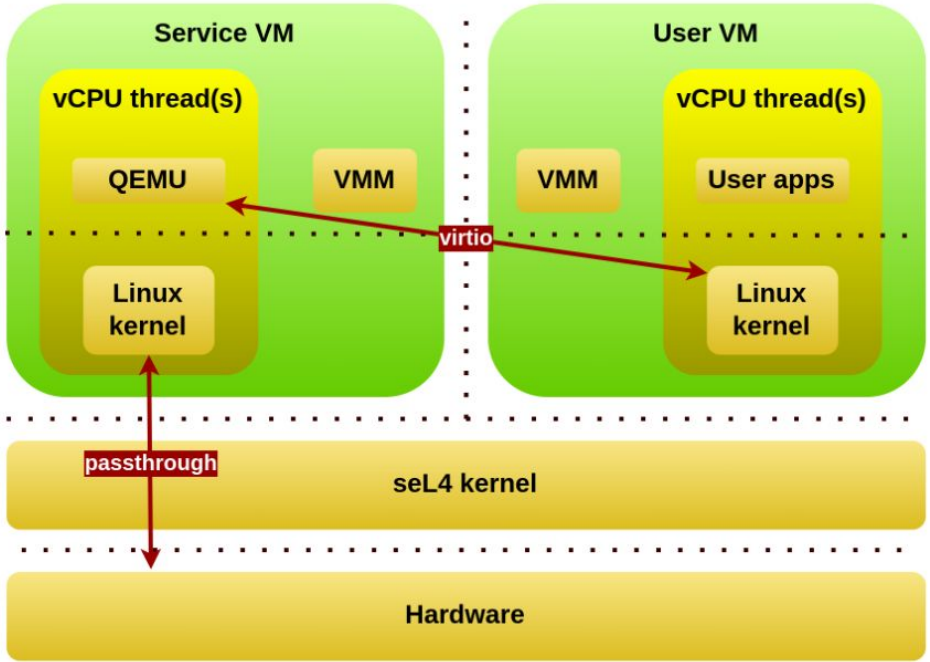
Outline

- Introduction / Recap
- Progress since the last summit
- What next
- Wrap up



Introduction / Recap

- **Using Linux based device models to extend seL4 virtualization features**
- QEMU as device model
- Raspberry PI 4b
- **Fully open source:** https://github.com/tiiuae/tii_sel4_build
- Paper accepted to IEEE TrustCom 2023
- Check Hannu's talk from the last summit: *Using QEMU to extend seL4 VirtIO support*

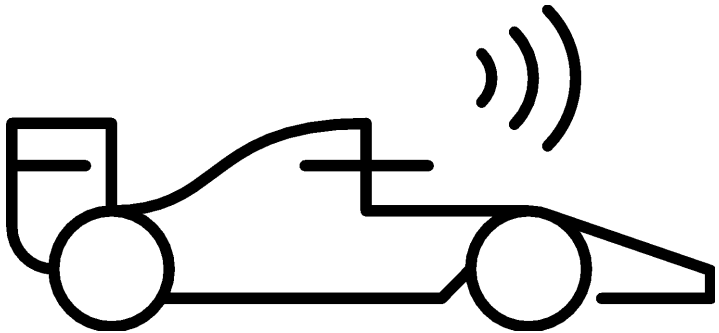


Technology Innovation Institute

What we've worked on

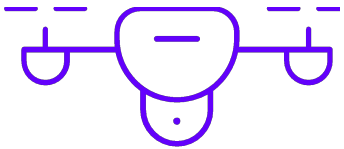
seL4-virt Linux kernel module

- **API for Linux device models and VMMs**
  - Simplifies integration with the Linux VMMs
- **Abstracts communication** with the seL4 VMMs
  - Underneath still uses CAmkES dataports exposed via vPCI (about to change)
- **Supports multiple Guest VMs**
- **Supports *ioeventfd* and *irqfd* signaling**

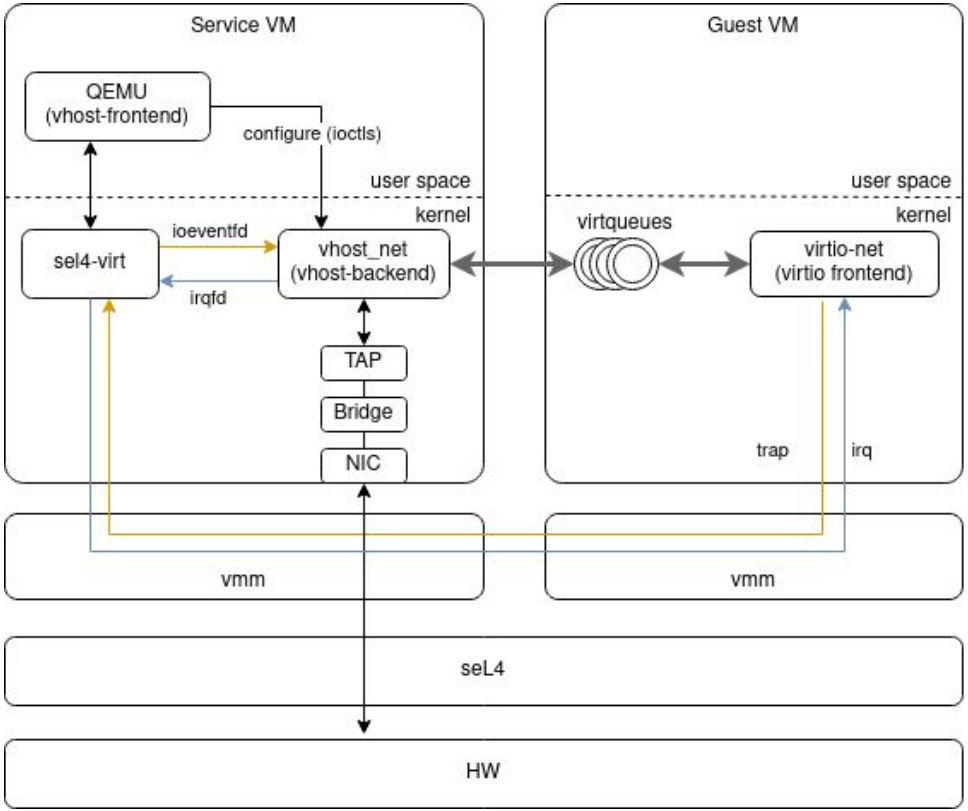


Vhost and Vhost-user
In a nutshell

- **A mechanism to offload VirtIO *virtqueue* processing** to an external process
  - *Vhost*: backend in a kernel module
  - *Vhost-user*: backend in an another user space process
- ***Vhost frontend*** (e.g. QEMU) takes care of **VirtIO initialization and feature negotiation**, and hands over
  - Memory region configuration (for mmap)
  - *virtqueue* configurations (within the region)
  - *eventfd* file descriptors (*ioeventfd* and *irqfd*)
- ***Vhost backend*** implements the device
  - **Processes virtqueues** (e.g. net packets to/from the guest)
  - **Listens *ioeventfd* for notifications from the guest**
  - **Writes *irqfd* to notify the guest**



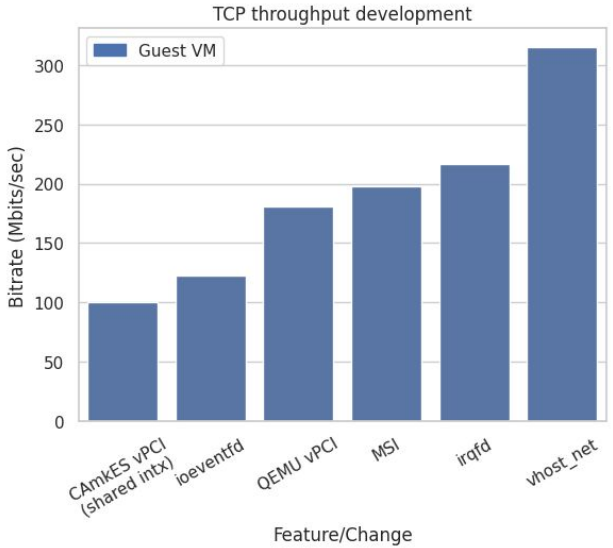
Overview
vhost_net



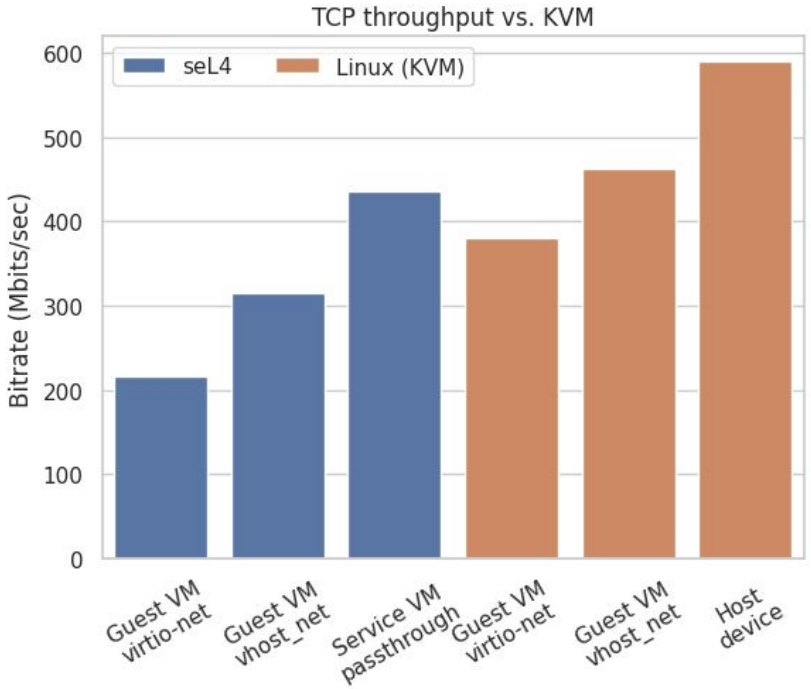
NOTE: Simplified diagram for clarity!

Vhost and Vhost-user on seL4
QEMU and Raspberry Pi 4B

- **QEMU *virtio-pci* expects *irqfd* notifications to be Message Signaled Interrupts (MSI)**
- **No MSI support in CAmkES vPCI**
  - Using QEMU vPCI instead
- **No MSI support in interrupt controller**
  - Emulating ***GICv2m*** device in QEMU: **maps MSIs to interrupts**
- *vhost_net* TCP **throughput 315 Mbits/s**
  - Still using CAmkES cross-connector **(extra traps involved)**
- We are working on
  - Improving inter-vm communication
  - vPCI MSI and GICv2m support to seL4
  - *irqfd* abstractions to QEMU *virtio-pci* (very KVM specific)

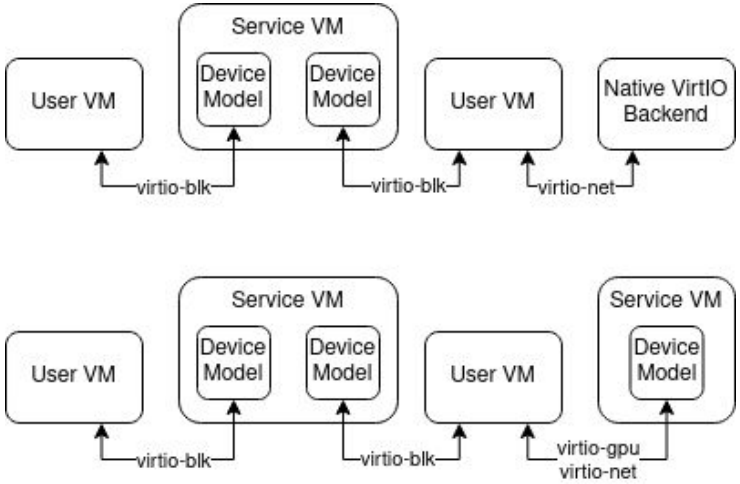


Current network throughput vs. KVM



Multiple Guest VMs

- Designed such that
  - **Multiple sources of VirtIO backends supported**
  - **Backends not part of VMM** (own PD)
  - Backend can be
    - **service-vm with a device model** or
    - **seL4 native VirtIO backend**
  - Uses *virtio-pci* transport, backends are PCI devices
- Currently supports one Service VM serving multiple guests (1:N)
  - **Each guest with own device model instance**
- We are working on
  - **Guest VMs being served by multiple Service VMs** (M:N)
  - ~~sel4cp~~ Microkit support



Other stuff we've done

- **System wide tracing**
  - Collects **trace from all Linux VMs and seL4 threads**
  - Uses **ftrace format**
  - Flame graphs! 🔥
- **Restrict Service VM access to Guest VMs memory**
  - Bounce buffers
  - Performance impact 📉

Technology Innovation Institute

Next steps

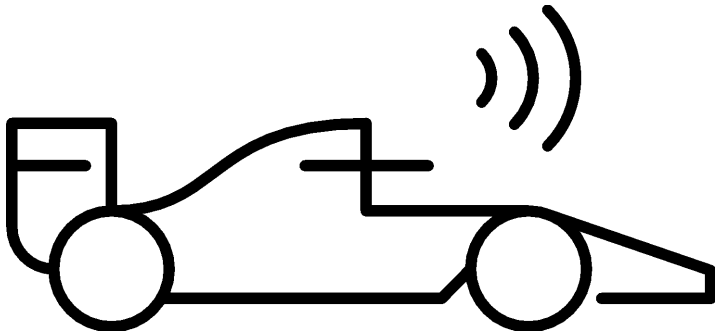
Next steps
VirtIO devices with vhost/vhost-user (without a device model)

What?

- **Driver VM without a full-blown device model**
- **vhost/vhost-user frontends** for seL4 with Rust
- Largely based on the work we've done with QEMU
    - Not a replacement, limited to available backends

Why?

- **Reduce the device model to what is strictly required**
- **Enable collaboration** with the other hypervisors and open source communities
    - rust-vmm community
    - Project Orko (Linaro) is developing hypervisor agnostic vhost-user backends to rust-vmm
- **Access to production quality VirtIO backends**
- **A place to contribute new VirtIO device backends**



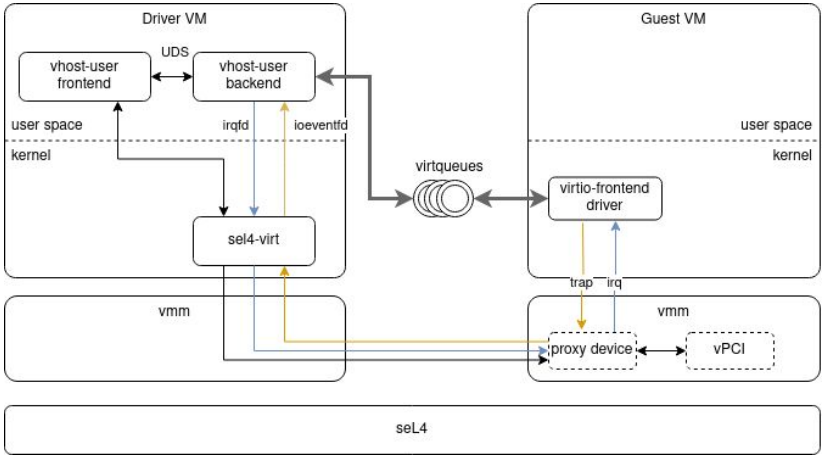
Next steps
VirtIO devices with vhost/vhost-user

How?

- **Dedicated *vhost-frontend* applications**
  - Code reuse (rust-vmm crates etc.)
- **Use existing *vhost/vhost-user* backends**
- **Support multiple Guest VMs**
- **No access to Guest VMs memory**
  - Bounce buffers
- Use *Kani* for verifying critical parts
- Targets *microkit*, *qemu-virt (Aarch64)* and *RPi4b*

When?

- The work has started, but at an early stage

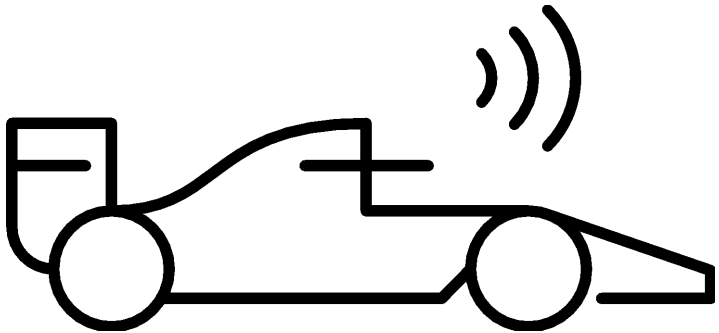


Technology Innovation Institute

Wrap up

Using Linux based device models
Our experiences

- **Easy access to high quality and stable device backends**
  - Support for majority of virtio backends
  - Just pass-through the devices and you're good
- **Can reach good performance**
  - Obviously doesn't compete with native performance
- **Can be extended with sDDF drivers** for native workloads
  - Not intended to compete, but to complement
- **Could be extended to be a VMM**
  - **Assumes control of VM lifecycle and the resources**
  - **Needs native components** (resource management)
  - Different role, different API
  - More choices for VMMs given good-enough abstractions (crosvm, cloud-hypervisor)



Technology Innovation Institute

Thank you!

Questions?