# IOMMU for Xuantie-based Platforms

Siqi Zhao

Aug, 2022

# Contents

- Overview
- Features
- Prototype
- Future Work

# Overview

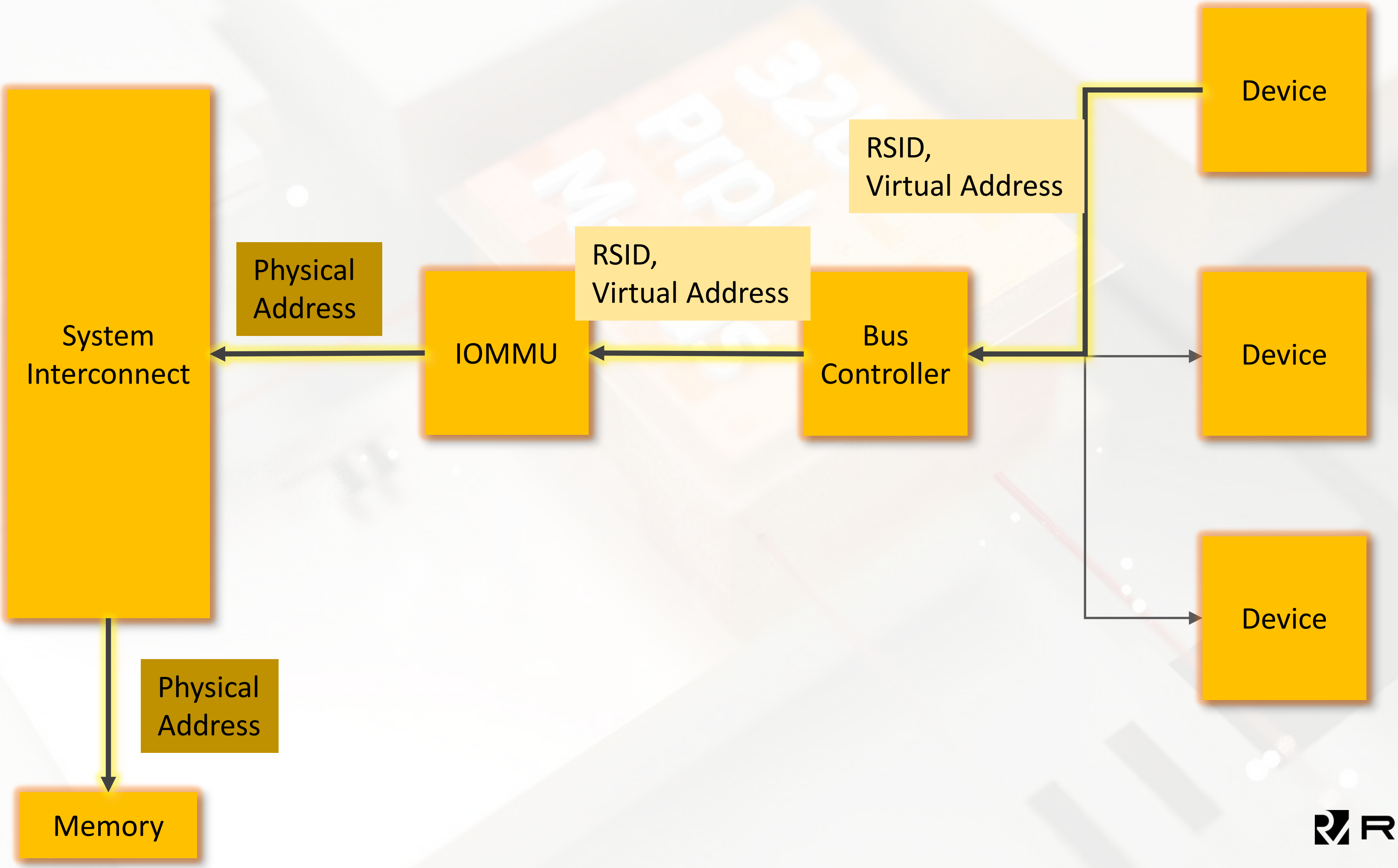
# Overview

- IOMMU is needed for RISC-V
  - Kernel protection
  - User space drivers
  - Virtualization
  - Security
- T-Head release a draft of its IOMMU spec in 2021, covering essential features for an IOMMU for RISC-V
  - Entered as one of the proposals for the IOMMU TG’s spec
  - Link: https://github.com/sqzsq/xuantie-iommu-spec
- An early prototype on QEMU for OS protection and user space driver was release in early 2022
- Since then, we have validated our design on a device passthrough prototype, and introduced additional features

# Features

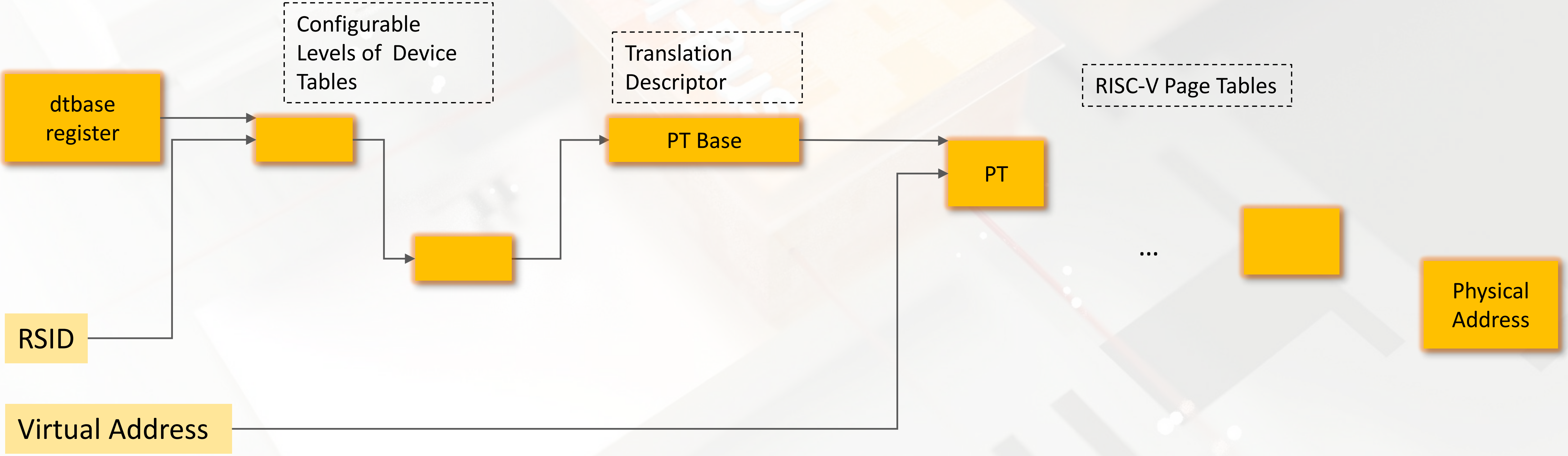
# IOMMU Translation

- An IOMMU translates the addresses in the DMA requests from the peripheral devices



# Extensible Structure

- The device table can be easily scaled up by adding more levels
  - Depending on how the devices under the management of the IOMMU is identified
- Separate the device context from the device tables for flexibility
  - Provides possibility for software to separately manage the two
- Compatible with RISC-V page tables for address translation

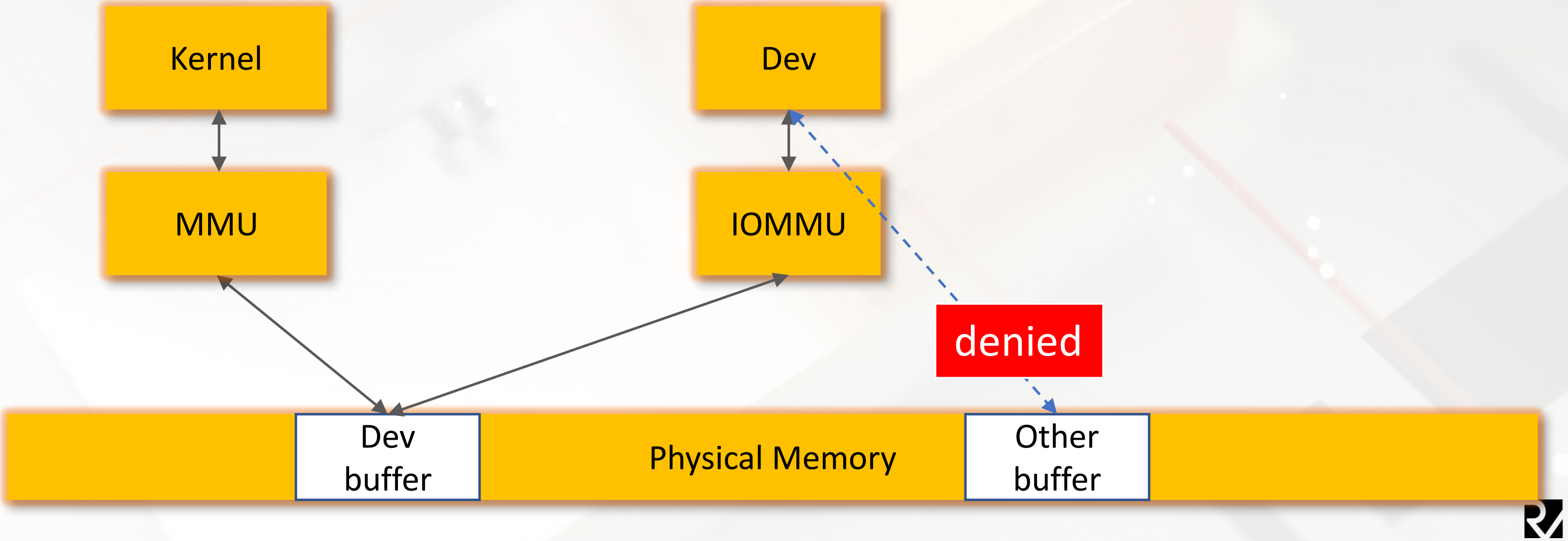


# Hardware Implementation

- There are multiple ways to implement the IOMMU in hardware
  - Private to device
  - Monolithic and shared
  - Distributed and shared
  - …
- One possible way to integrate the IOMMU is a distributed implementation
- Each device is equipped with a device-specific TLB
- A shared TLB sits with the Page Table Walk logic, serving as a second level of TLB

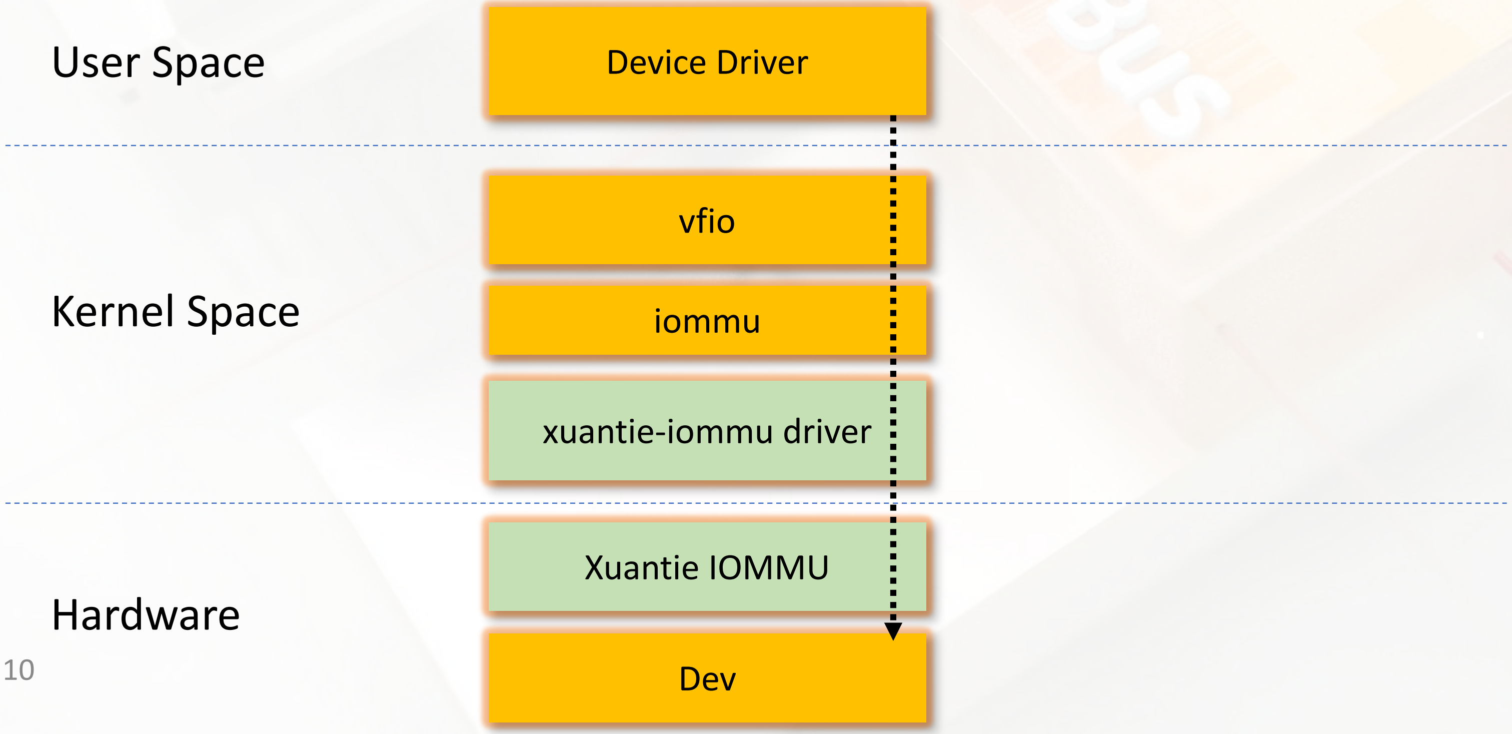
# OS Protection

- The OS kernel utilize the IOMMU to impose restriction on the range of physical address a device can access
  - Security
  - Reliability
  - Debug
  - Memory utilization
  - …
- The Linux kernel’s IOMMU layer provides APIs for kernel space drivers to manage the buffer
- Our IOMMU driver fully conforms to the Linux kernel IOMMU layer, no change to the device driver is required.



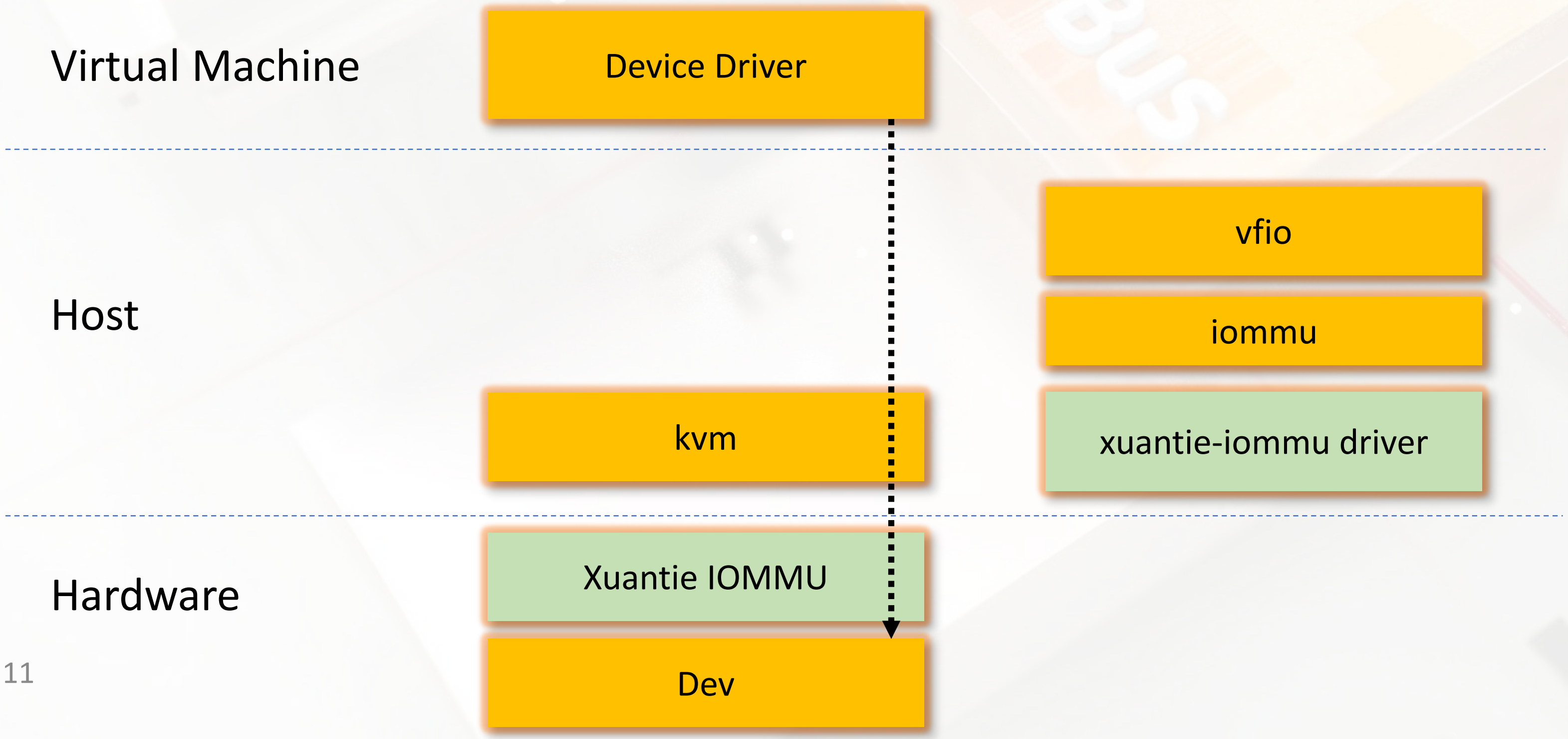
# User Space Driver

- The OS kernel can also let a user space application directly control a device, in which case a user space driver is execution, for example, DPDK and SPDK etc.
- The existing VFIO framework allows user space programs to directly operation on PCIe devices and platform devices.
- Our IOMMU prototype is VFIO enabled and runs a user space driver
- No other change is required for the user space driver



# Device Passthrough

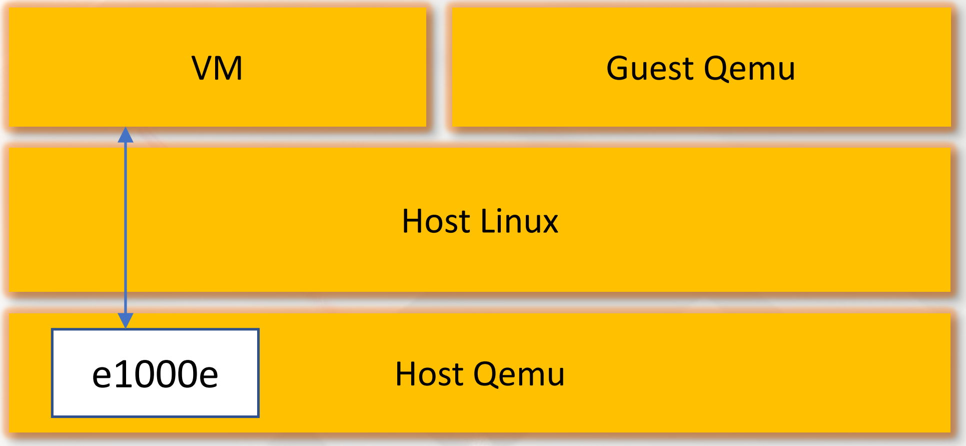
- For efficiency, virtual machines is given direct access to peripheral devices, i.e. passthrough devices.
- DMA from these devices are restricted to the memory of the virtual machine it belongs to.
    - IOMMU’s interposing on the DMA requests can achieve this.
- Our IOMMU fits into the KVM/VFIO facility for device passthrough
- No change in the KVM module or VFIO framework is required



# Prototype

# Implementation

- We have evaluated our design on QEMU + Linux for the following use cases
  - Host usage
  - Device passthrough
- For device passthrough we enabled the hypervisor extension on QEMU and setup KVM
  - The passthrough device is an e1000e NIC emulated by host-QEMU
  - We enabled necessary VFIO facility to let the VM directly use the NIC



# OS Protection

- Host usage demo

# Device Passthrough

- device passthrough demo

# Future Work

# Future work

- More features
  - PCIe support
  - AIA integration
  - Debug / trace
- Virtual IOMMU
- Hardware implementation
- Security / Confidential Computing
- Device co-operation

# END