A short walk through on virtualization and specialized virtualization cards

Yizhou Shan
syzwhat@gmail.com
Nov 2021 @ UCSD

WukLab

UC San Diego

Remote Memory Reading Group

DISCLAIMER
This slide is made based on public knowledge and reflects my own thoughts. In short, it could be wrong.

Outline

- A short history on virtualization
- Virtualization Essense
- Case Studies
  - QEMU + KVM
  - Xen
- Virtualization in the Cloud
- Virtualization cards

A short history on virtualization

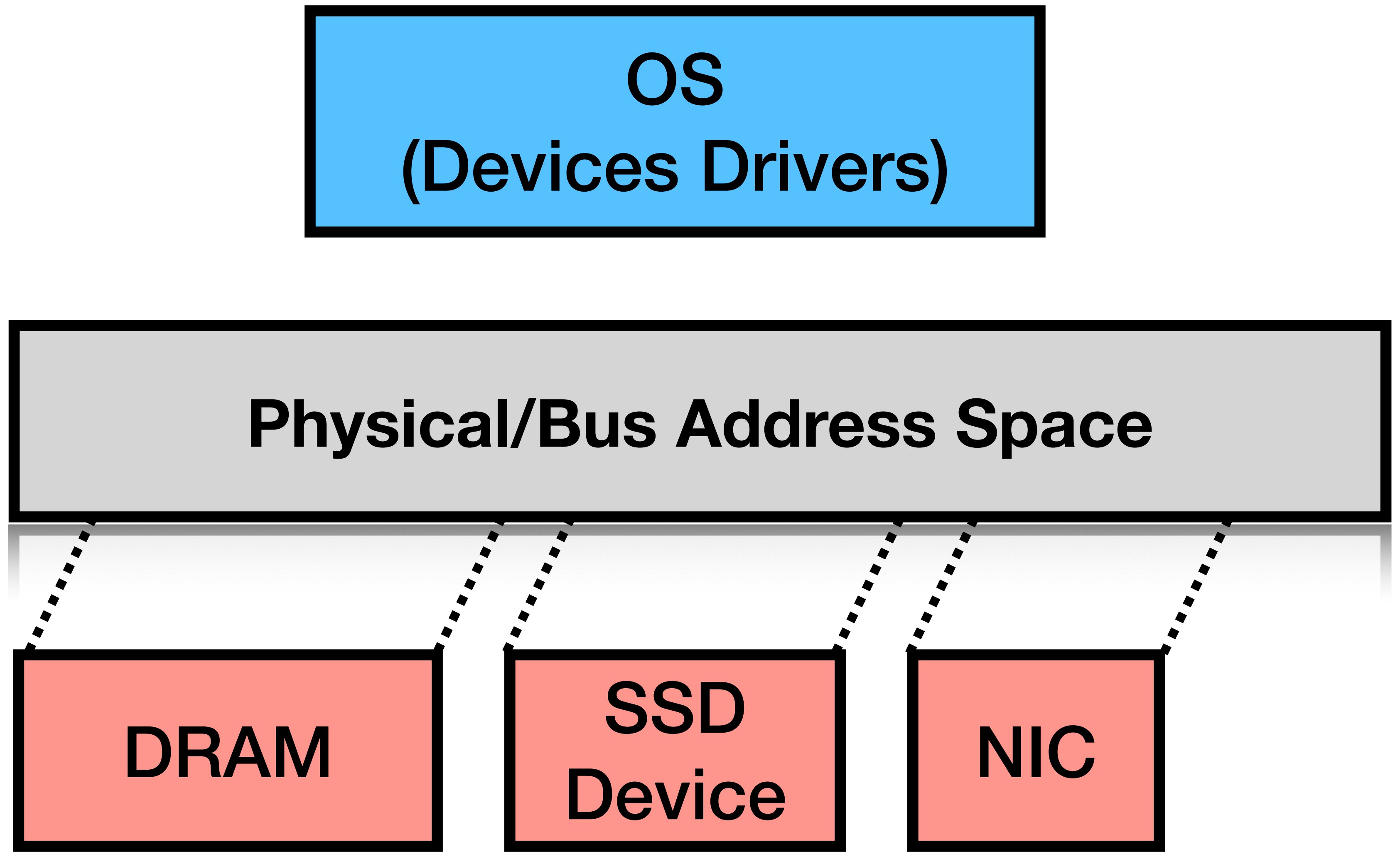
- Pure software simulation - VMware
- Para-virtualization Xen
- Hardware-supported virtualization - Intel/AMD VT-d
- Specialized virtualization cards - AWS Nitro and Microsoft FPGA
- Bare-metal virtualization - WIP

(Pretend there is a nice timeline figure here)

Virtualization Essense

- Essense
    - Emulate Address Space and devices behind it
    - Catch special Instructions
- Quote from QEMU developers

    And at the end of the day, all virtualization really means is running a particular set of assembly instructions (the guest OS) to manipulate locations within a giant memory map for causing a particular set of side effects, where QEMU is just a user-space application providing a memory map and mimicking the same side effects you would get when executing those guest instructions on the appropriate bare metal hardware



In a bare-metal server,
the physical address space directly maps to devices

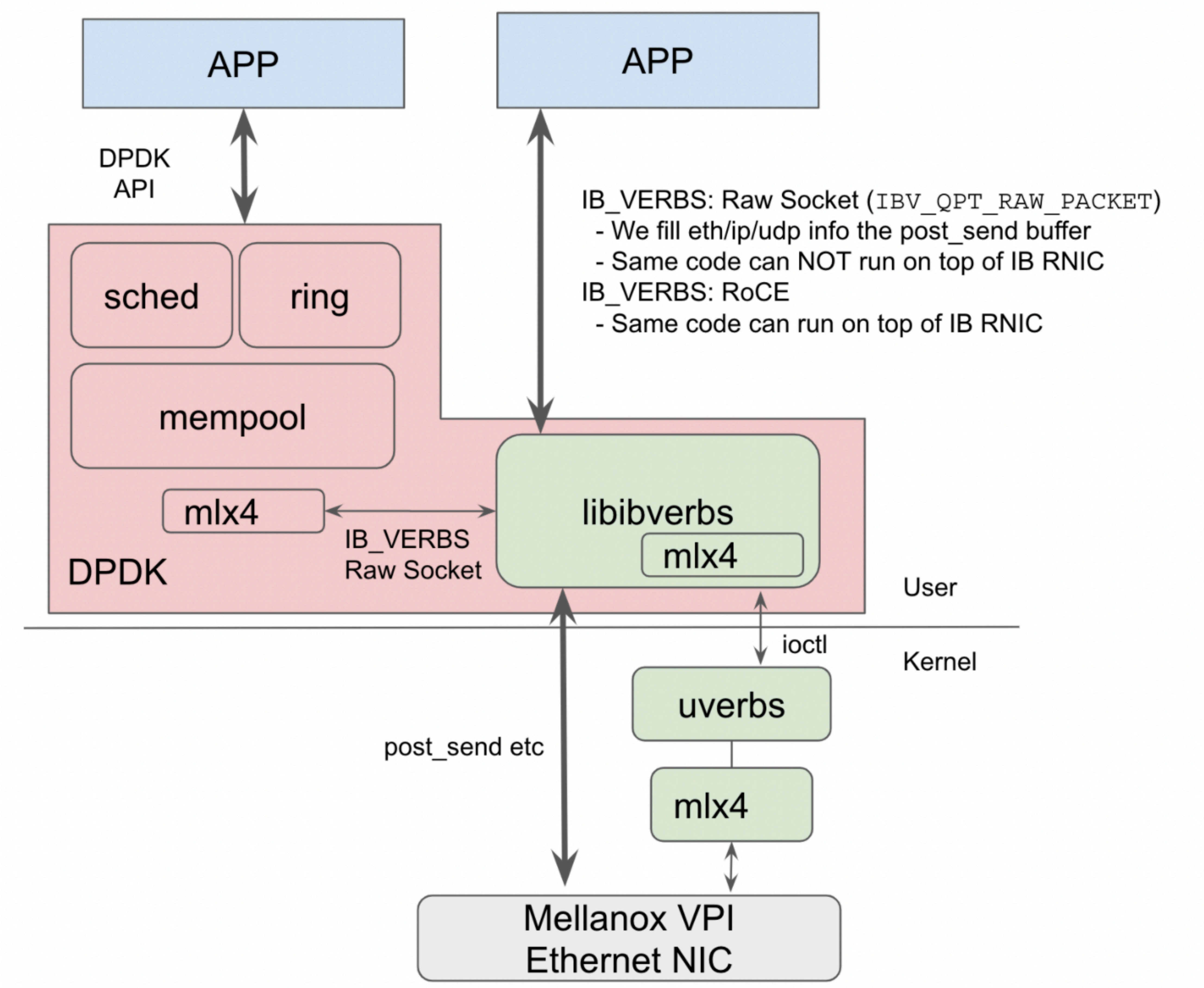
Device drivers do actions by writing or reading corresponding device’s PCIe address range.

For example, ibv_post_send() allows a user-program directly writes into RDMA NIC’s memory-mapped PCIe range, hence causing the NIC to do some actions.

Same thing for NVMe driver (either kernel or SPDK)

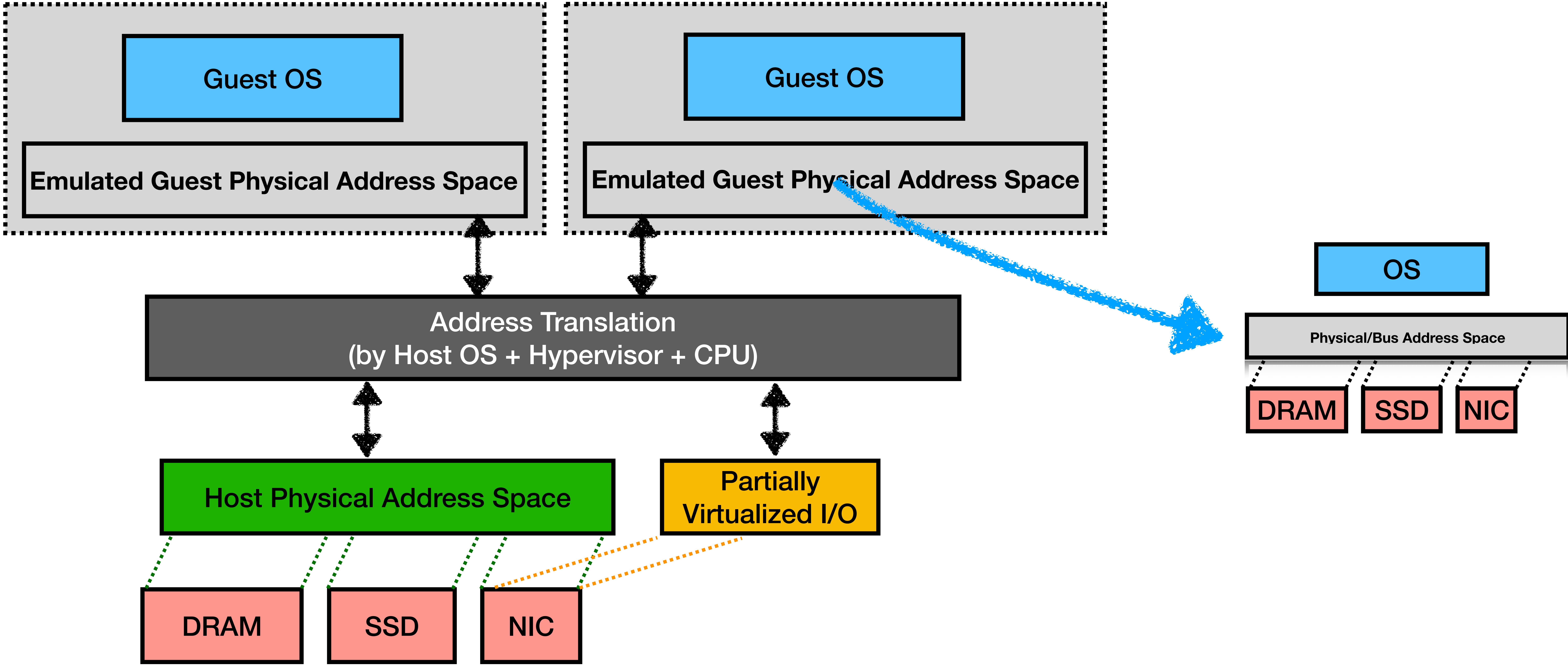
/proc/iomem

```
…
100000000-107fffffff : System RAM
  c05800000-c06600e80 : Kernel code
  c06600e81-c070581bf : Kernel data
  c07325000-c077fffff : Kernel bss
380000000000-380fffffffff : PCI Bus 0000:00
  380000000000-3800001fffff : PCI Bus 0000:01
385000000000-385fffffffff : PCI Bus 0000:85
386000000000-386fffffffff : PCI Bus 0000:ae
  386ffc000000-386fffffffff : PCI Bus 0000:af
    386ffc000000-386ffdffffff : 0000:af:00.1
      386ffc000000-386ffdffffff : mlx5_core
    386ffe000000-386fffffffff : 0000:af:00.0
      386ffe000000-386fffffffff : mlx5_core
…
```



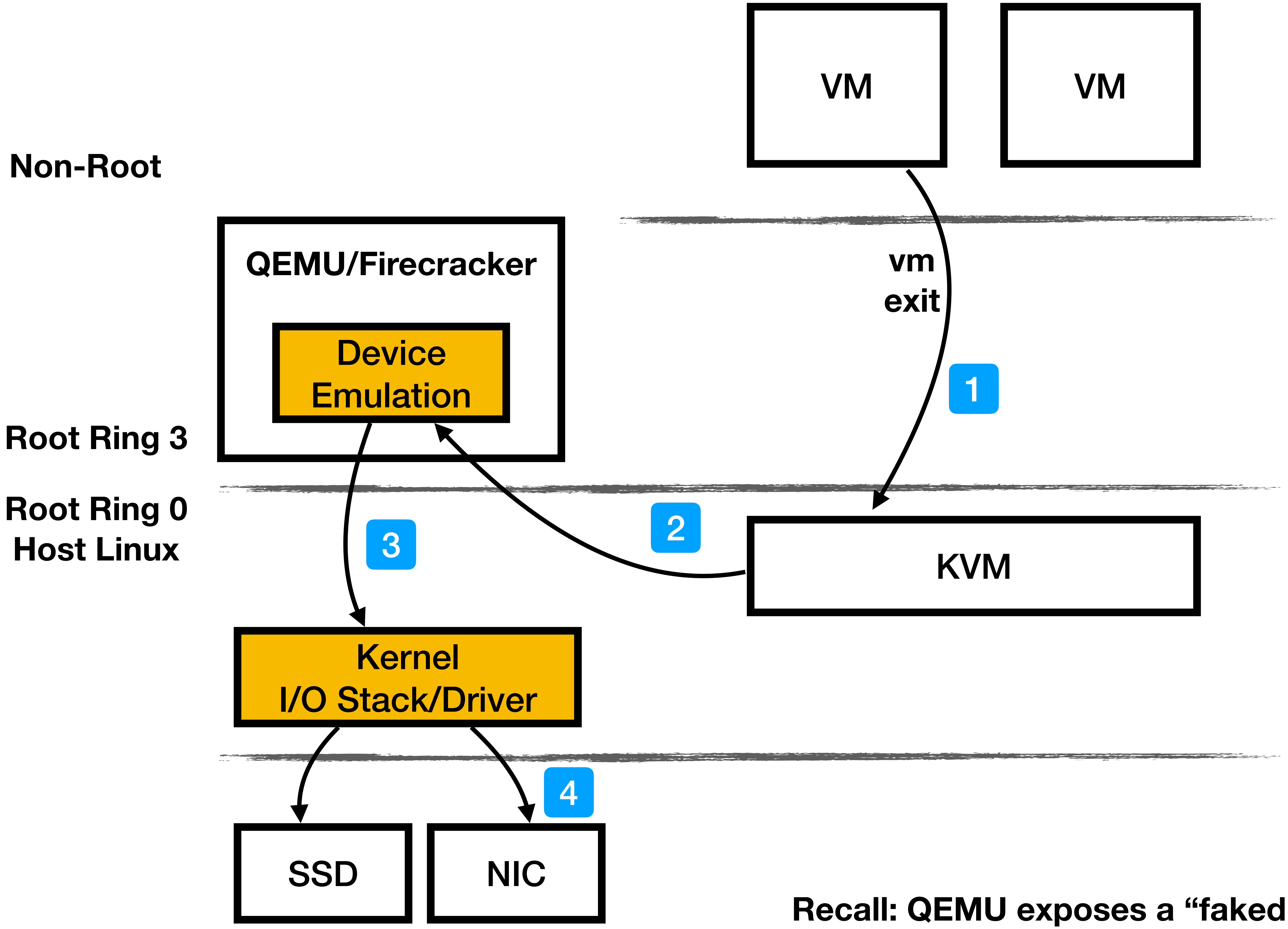
IB_VERBS: Raw Socket (IBV_QPT_RAW_PACKET)
 - We fill eth/ip/udp info the post_send buffer
 - Same code can NOT run on top of IB RNIC
IB_VERBS: RoCE
 - Same code can run on top of IB RNIC

Address Space Mapping in the Virtualized Environment


Guest OS
Emulated Guest Physical Address Space
Guest OS
Emulated Guest Physical Address Space
OS
Physical/Bus Address Space
DRAM
SSD
NIC
Address Translation
(by Host OS + Hypervisor + CPU)
Host Physical Address Space
Partially
Virtualized I/O
DRAM
SSD
NIC

Outline

- A short history on virtualization
- Virtualization Essense
- **Case Studies**
  - **QEMU + KVM**
  - **Xen**
- Virtualization in the Cloud
- Virtualization cards



Device Emulation

1. Pure Emulated Devices (e.g., serials)
2. Normal emulation (NIC card, HDD, SSD)

Essence of the Device Emulation Layer

1. Multiplex, Protection
2. Enforce extra functionalities (e.g., packet encap/decap)

Recall: QEMU exposes a “faked” address space to VM (by using EPT).
From VM’s perspective, its driver sees the same address range (PCIe range).
QEMU mimics device behavior.

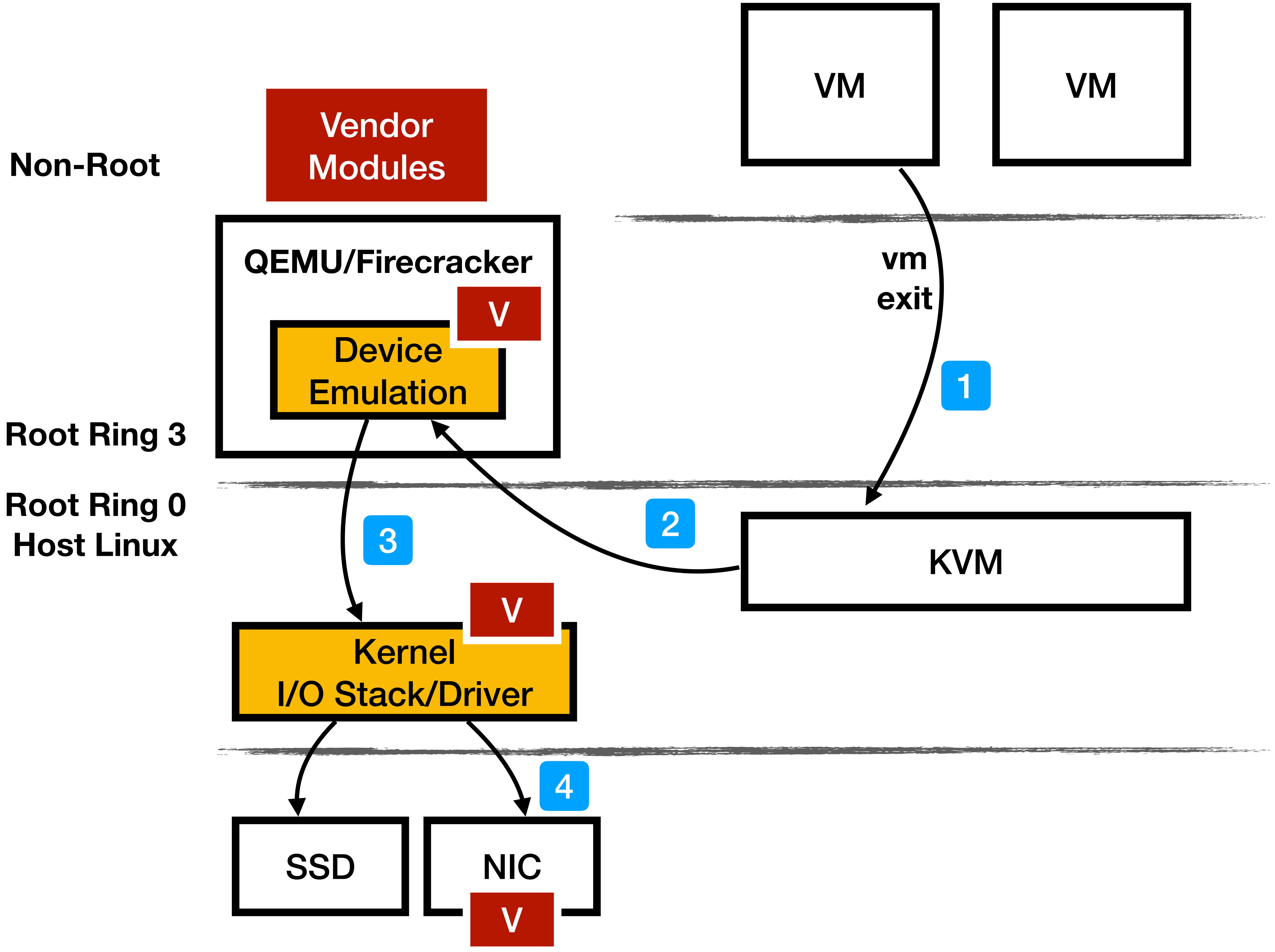
Suppose to have Xen here

Outline

- A short history on virtualization
- Virtualization Essense
- Case Studies
  - QEMU + KVM
  - Xen
- Virtualization in the Cloud
- Virtualization cards

Virtualization in the Cloud

- Status
  - AWS: Xen -> KVM
  - Azure: Hyper-V + something
  - Alibaba & Huawei: KVM ?
- They need more to have a complete virt solution
  - Network virtualization (e.g., OpenVSwitch + NFs)
  - Customized storage stack
  - Security checking



What are vendor modules?

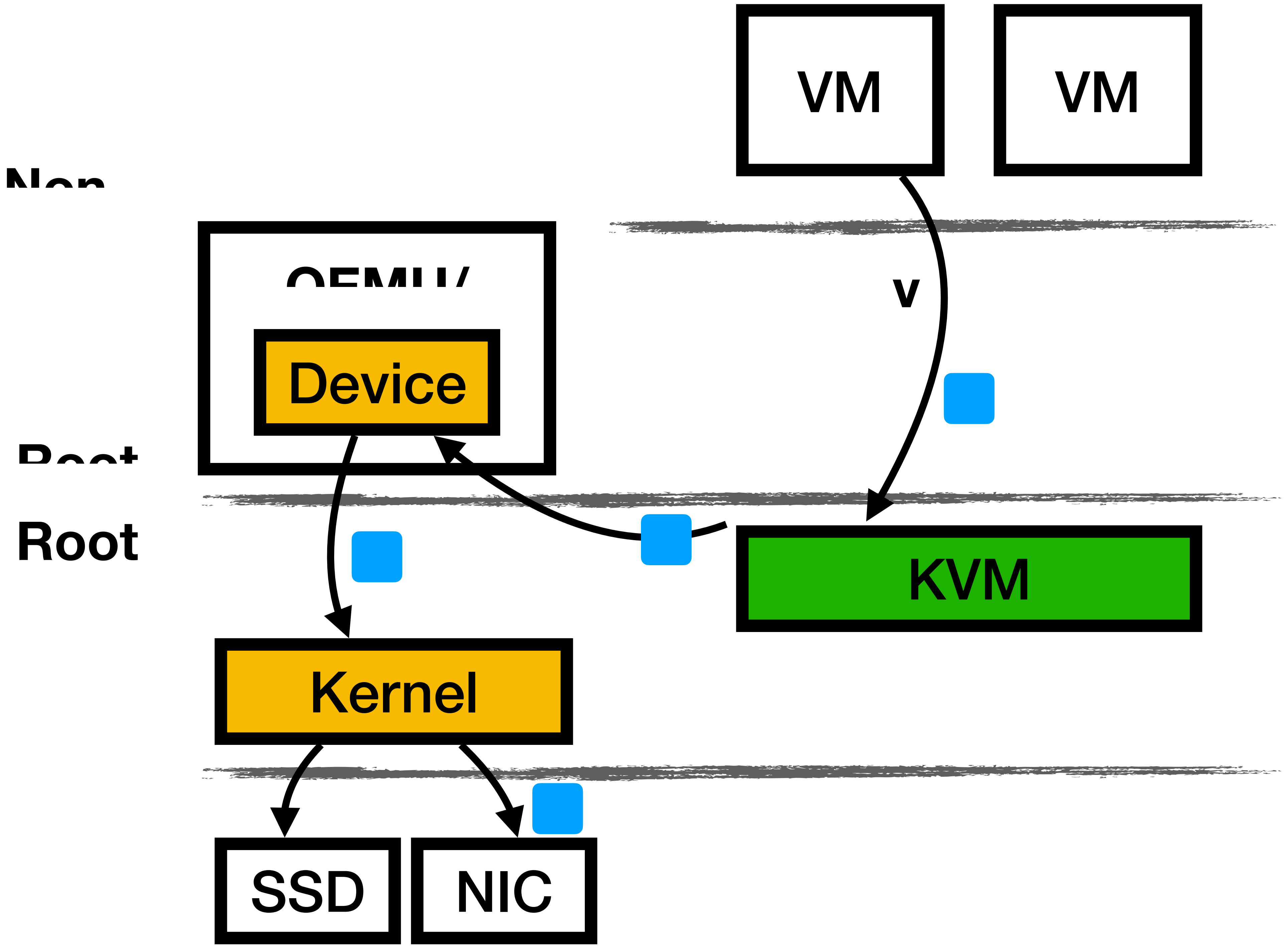
- Enforce vendor policies
- e.g., OpenVSwitch, NF Rules etc
  Storage Encryption

Where to add vendor modules?

- In a separate user space program
- In QEMU
- In host OS
- in hardware

(Check out the AccelNet, NSDI’18 paper)

Threading Model

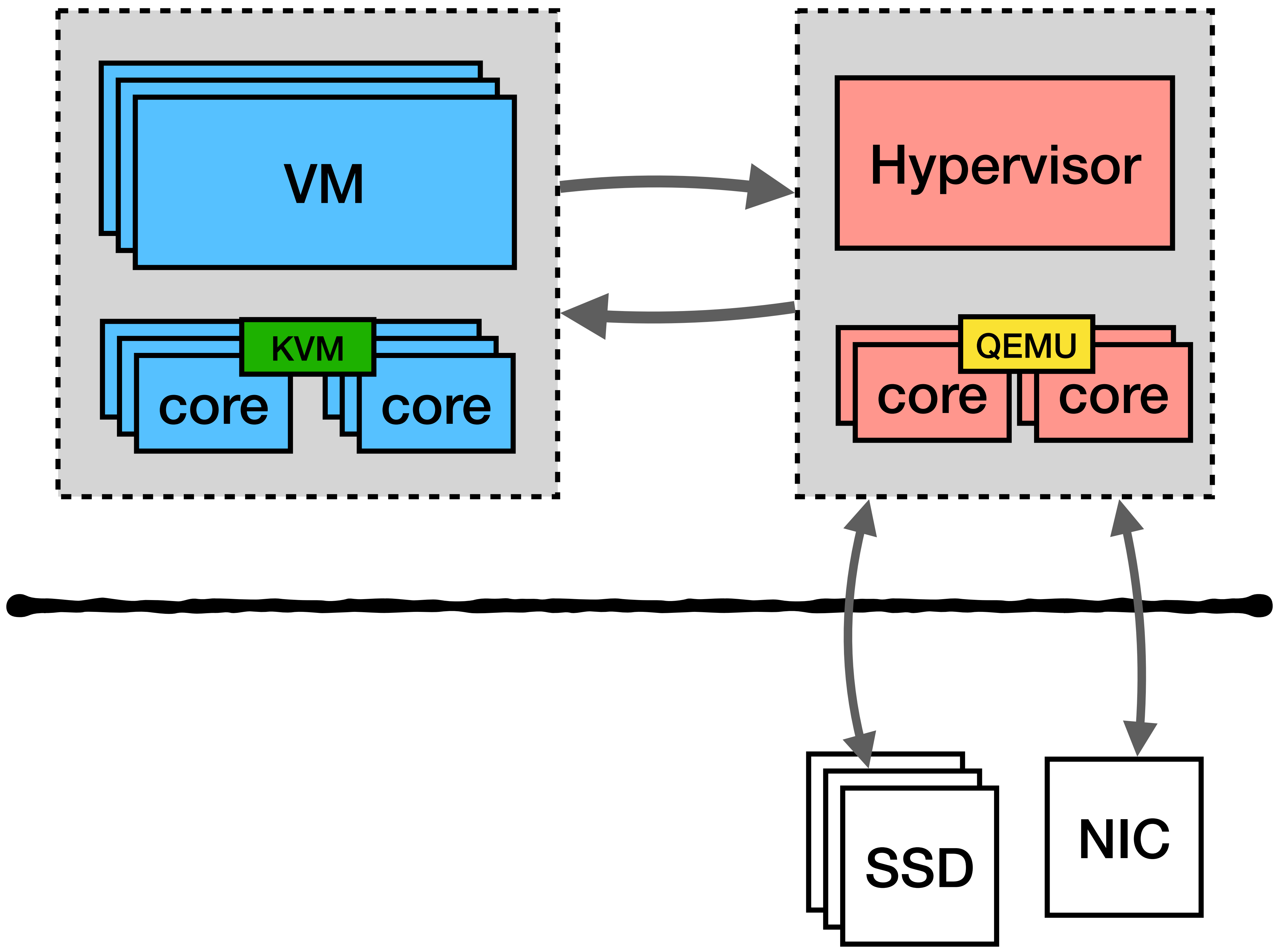


Are all these ops running on the same core??

We shouldn't - bad for provisioning & scaling.

QEMU has separate I/O threads for dev emu

Cloud vendors reserve cores to run hypervisor



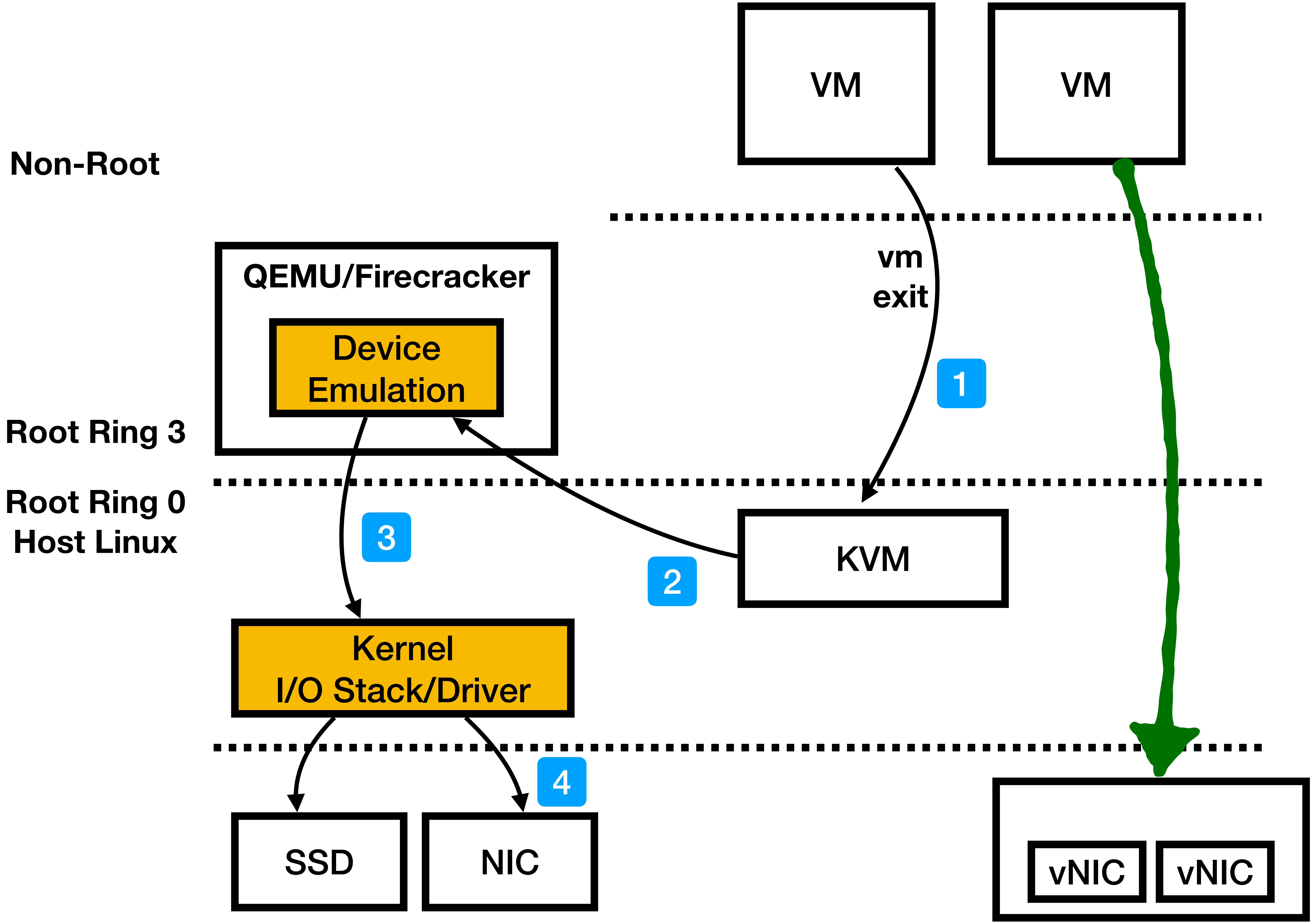
Outline

- A short history on virtualization
- Virtualization Essense
- Case Studies
  - QEMU + KVM
  - Xen
- Virtualization in the Cloud
- Virtualization cards

Virtualization is no free lunch

- **High perf cost!**
  - High-speed I/O
  - 2-level VM (EPT) implicit overhead
  - VM exit/enter (e.g., periodical timer interrupt)
- **Optimizations?**
  - Para IO virtualization - VIO between QEMU and guest; but this model cost a lot of CPU cycles, especially for high-performance IO stack (not our focus today)
  - SR-IOV
  - Specialized Cards

SR-IOV enabled Passthrough



Normal SR-IOV enabled devices presents itself as multiple virtual devices. Internally, it has a *multiplexing* layer for all virtual devices.

Problem?

SR-IOV is ALL or NOTHING.
It bypasses cloud vendor modules

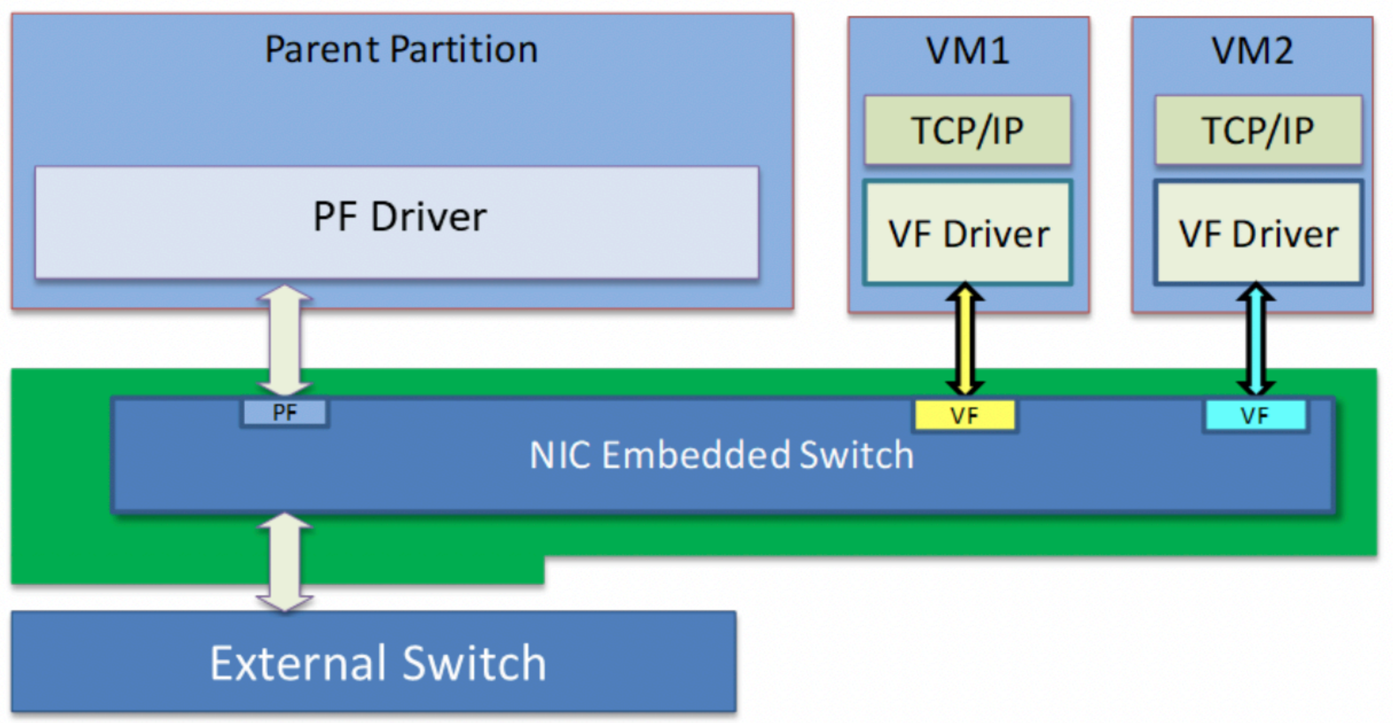
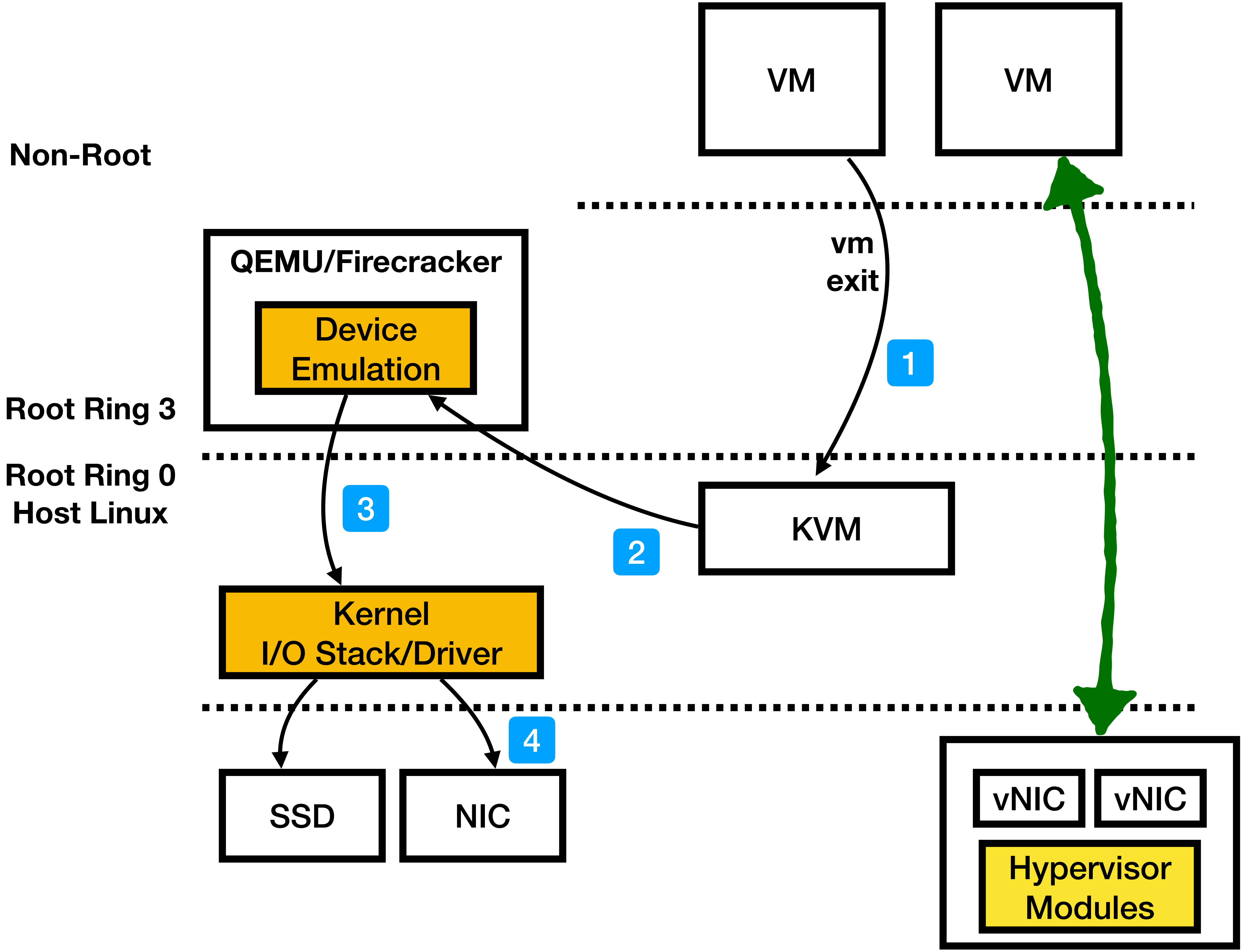
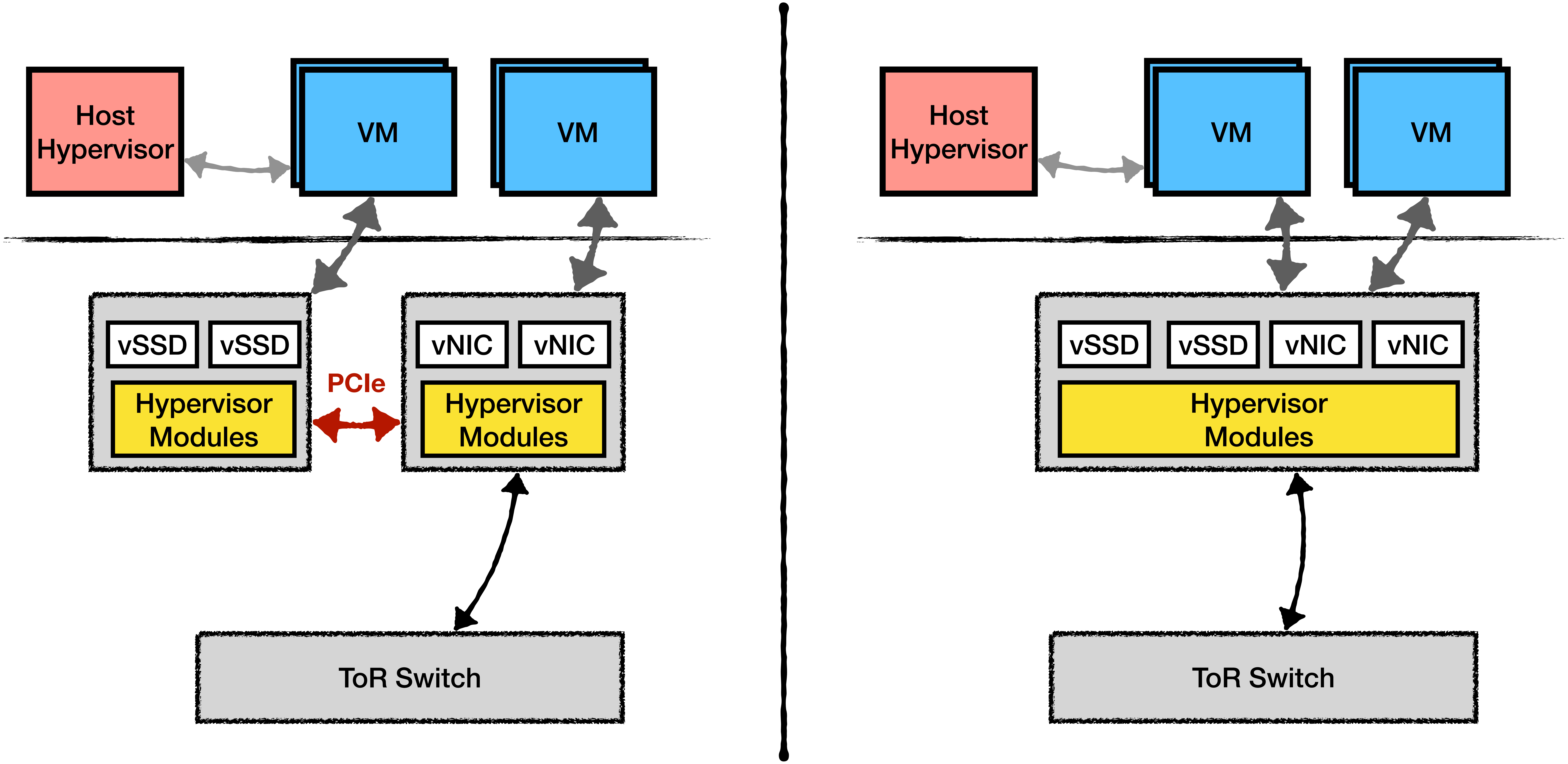


Figure 1: An SR-IOV NIC with a PF and VFs.

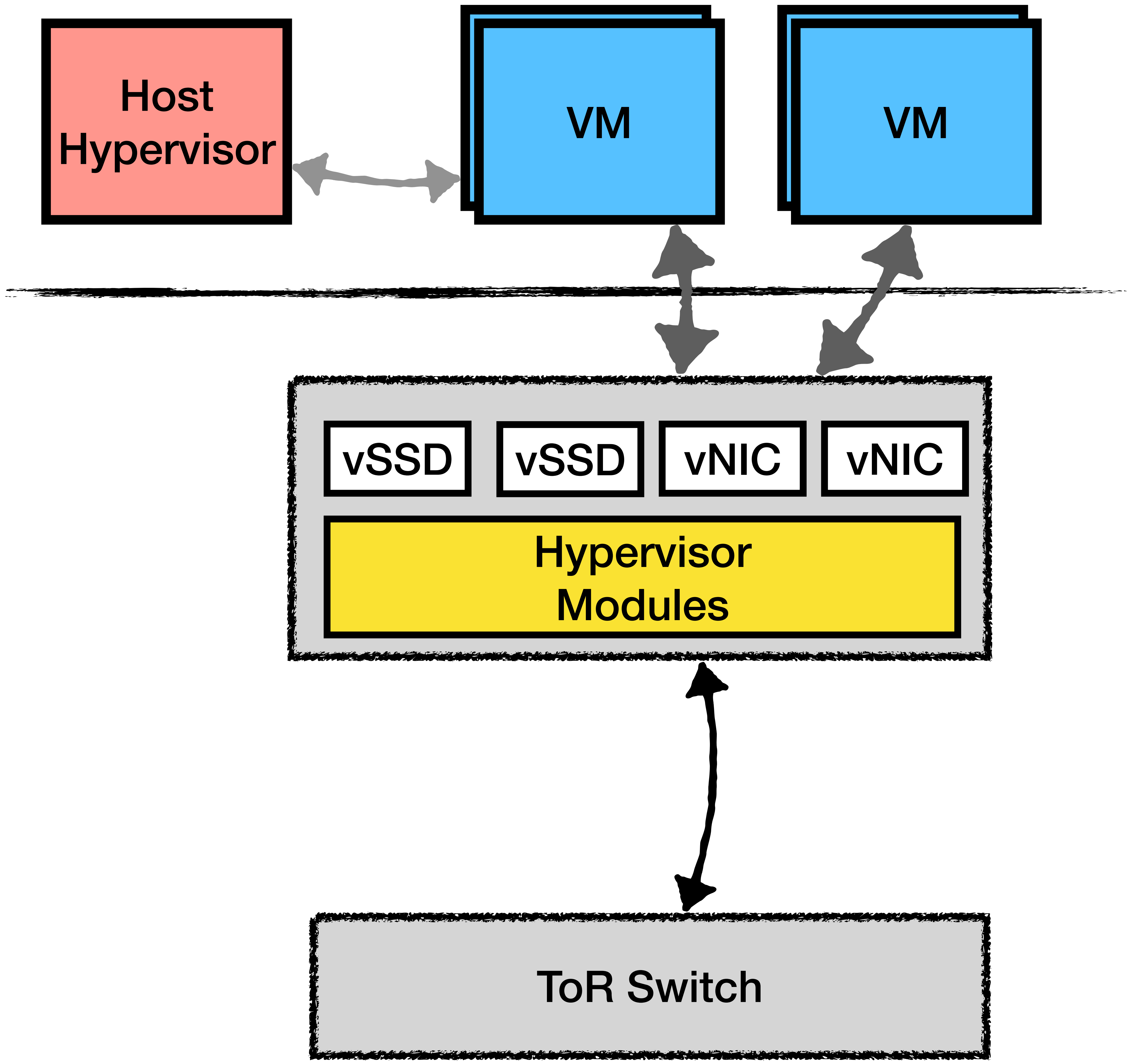
Virtualization Cards



Essentially
SR-IOV + Hypervisor modules!



NOTE: This is mostly for the data path.
Control path might be more complex - but not perf critical



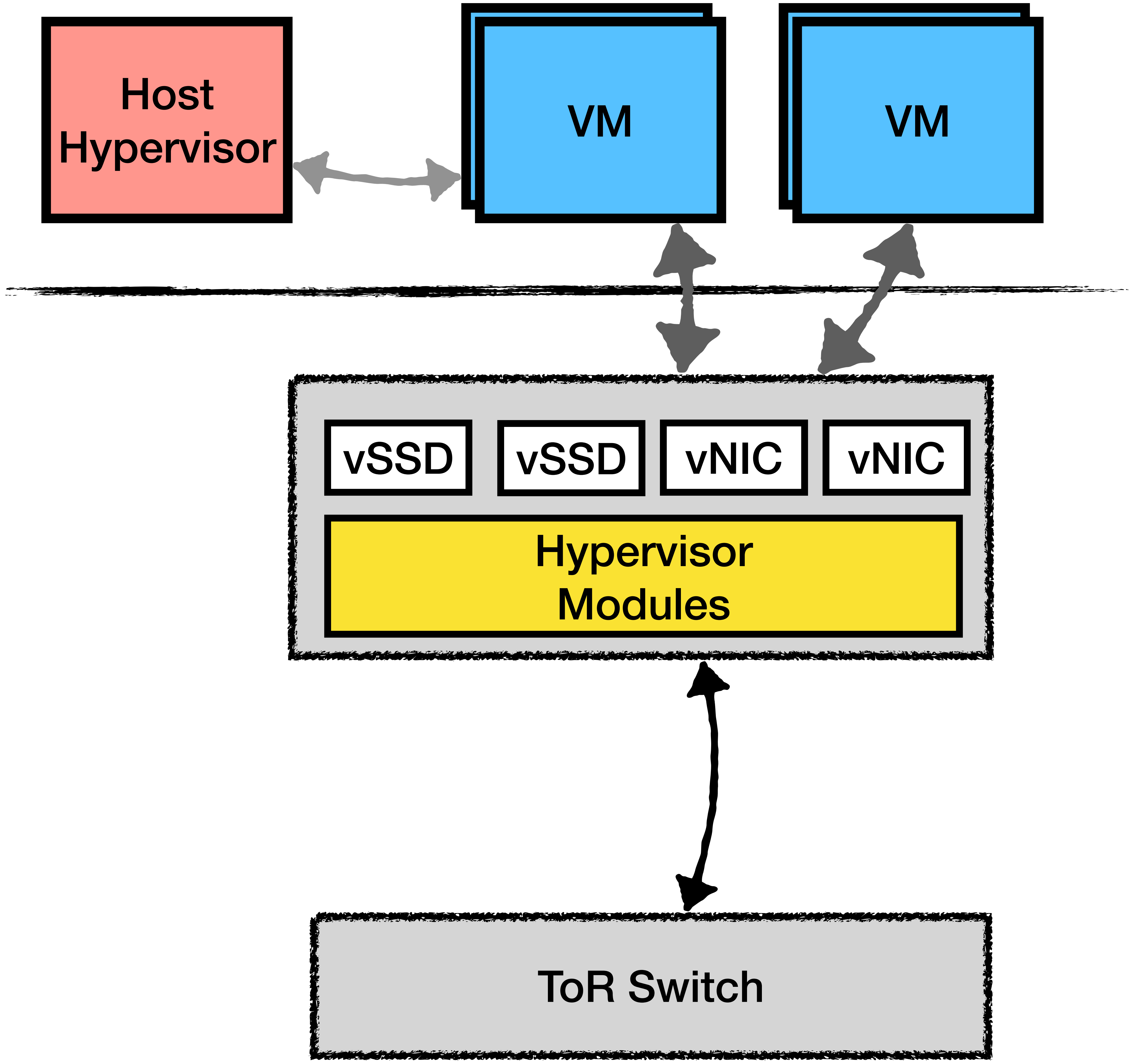
How can we implement those hypervisor modules?

1. ASIC
2. FPGA
3. SoC

Does it has to be one way or another?
No - they can be combined. Depends on vendor usages.

What's the benefit of SoC here?
Fast prototyping
Easier for non-FPGA/ASIC teams to deploy new stuff

Going Forward



So, are we done? Apparently no.

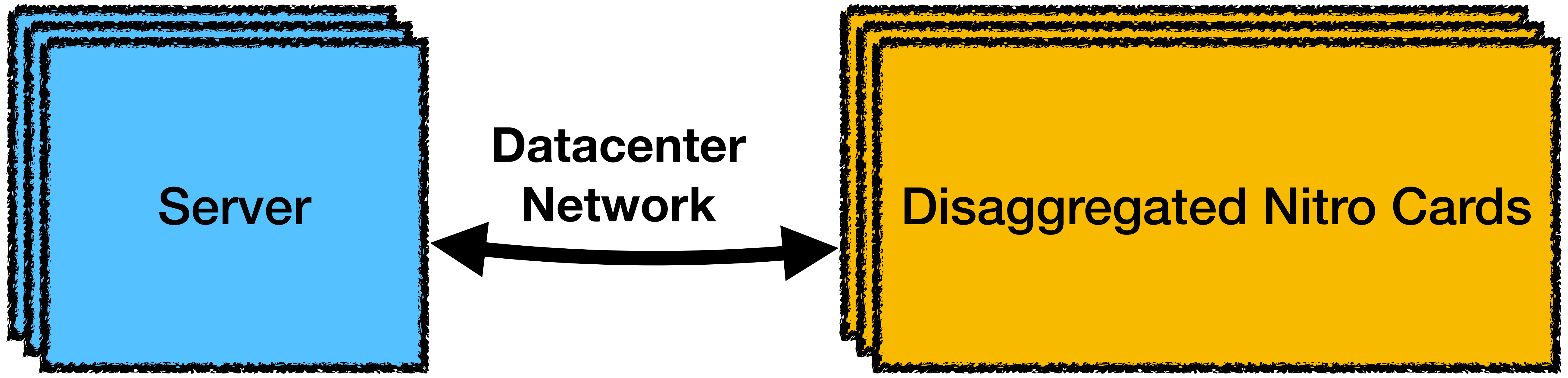
Two major issues

1. VMs are still running on VT-d CPU - EPT perf cost
2. Cards provisioning. Can one card support all usages on a server? Can they use remote cards?

==>
Bare-metal virtualization
Disaggregated virtualization cards

Disaggregated Hypervisor Pool

- A standalone hypervisor pool
- Benefits
  - Separate hypervisor processing power provisioning
  - Elastic, auto-scaling



Summary

- Virtualization cards are no myth

The Dark Side of Virtualization

- 2-level paging overhead if the customer uses no virt feature at all
  - 20-30% overhead
- Vendors are looking into bare-metal virtualization
  - ref ISCA’10 paper