Scalable High-performance Userland Container Networking for NFV

Jianfeng Tan, Cunming Liang, Huawei Xie, Zhihong Wang, Yuanhan Liu, Heqing Zhu

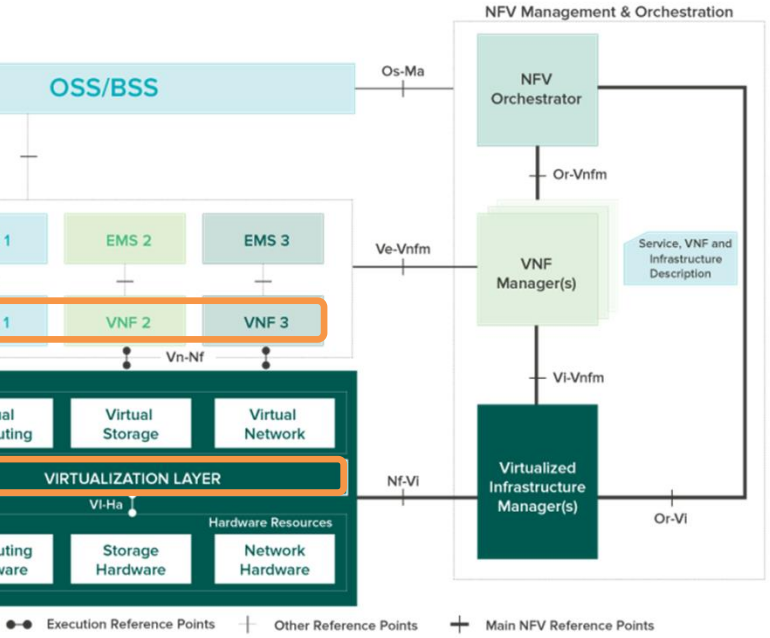
Agenda

- Background
- Userland container networking
- Impact on application development

NFV and challenges

Virtual Network function

Virtual machine, **Container**



VM vs Container based VNFs

- Challenges of VM-based VNFs
  - Provisioning time
  - Runtime performance overhead
- Challenges of container-based VNFs
  - **High performance networking**
    - For VNFs, like LB, FW, IDS/IPS, DPI, VPN, pktgen, Proxy, AppFilter, etc
  - Security

Challenges of high perf. network

NIC	Time budget for 64B	Time budget for 1518B
10Gb	67.2 ns	1,230 ns
40Gb	N/A	307 ns
100Gb	N/A	120 ns

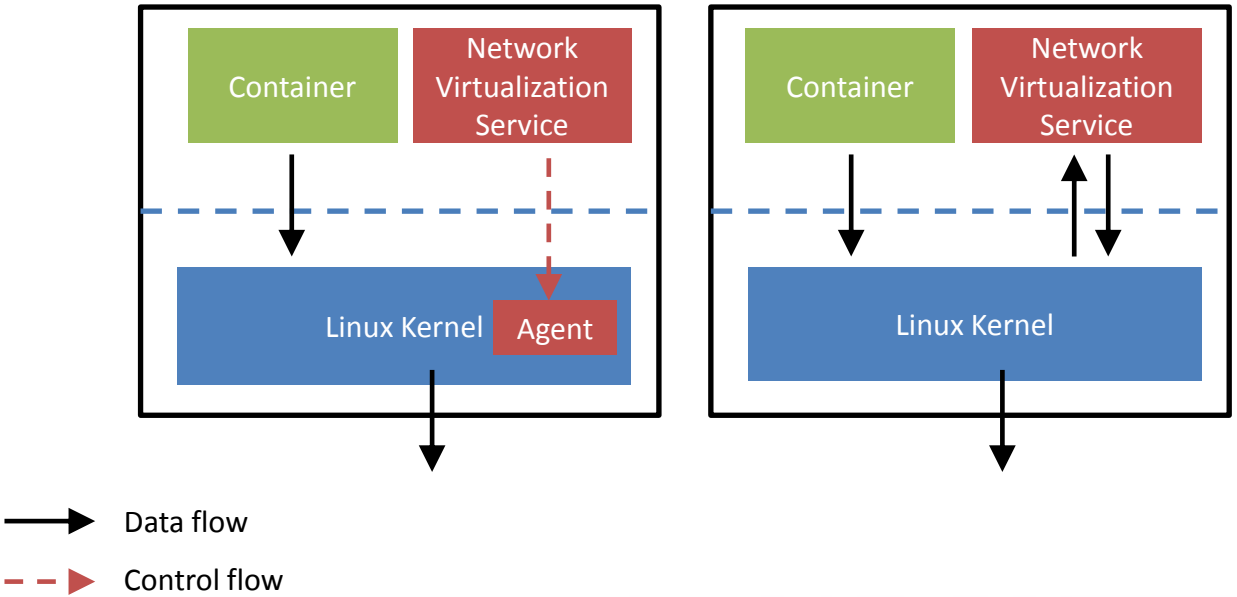
NIC	Time cost
System call	75 ns/42 ns
Atomic ops	8.25 ns
Spinlock lock/unlock	16+ ns
L3 miss*	~53 ns

- Forward 1~2 Mpps per core

* Tested on Intel i7-3770 (Ivy Bridge) by www.7-cpu.com
Other data from LWN article

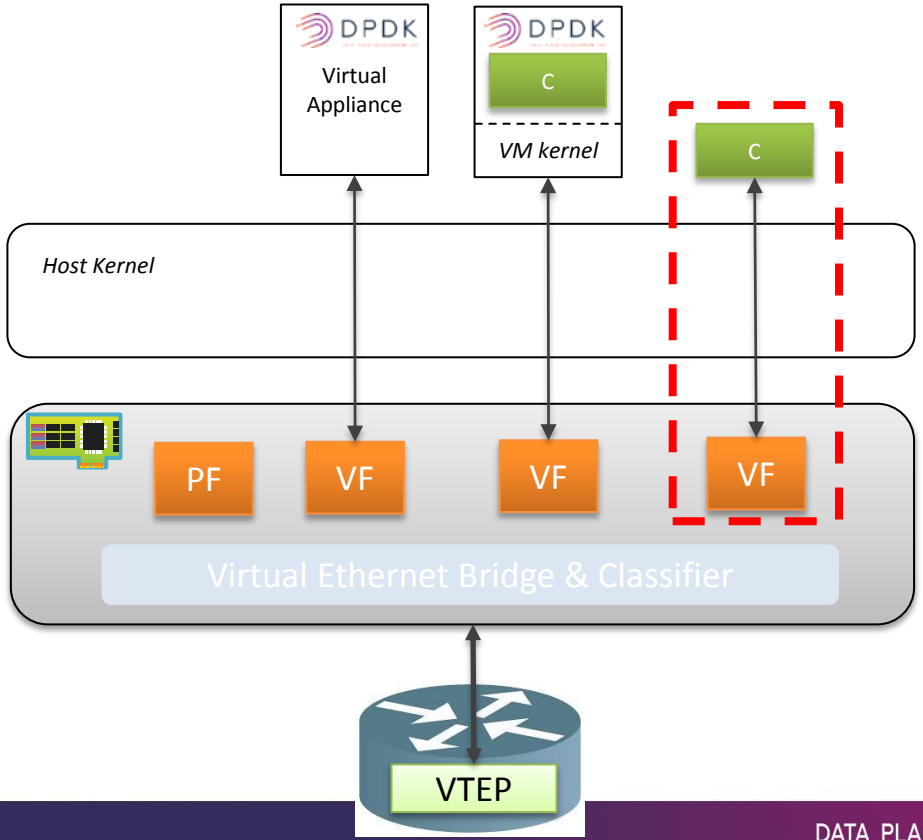
Container networking status quo

- Multi-host networking

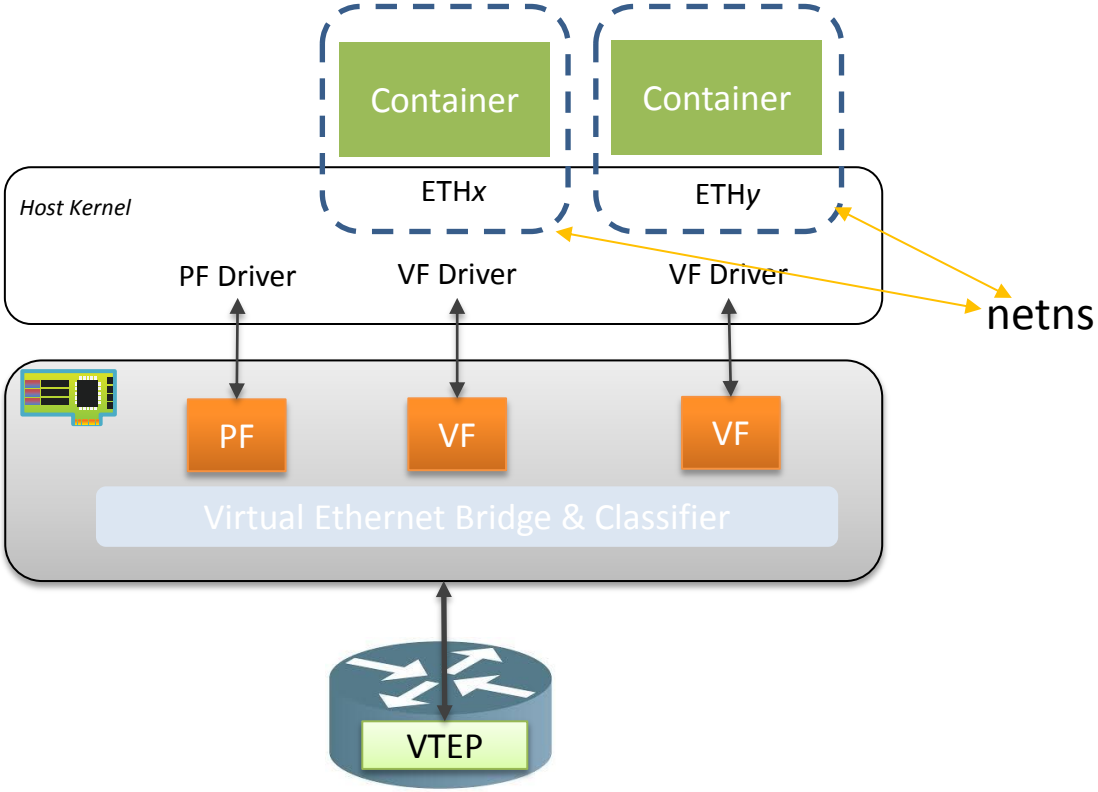


Userland Container Networking

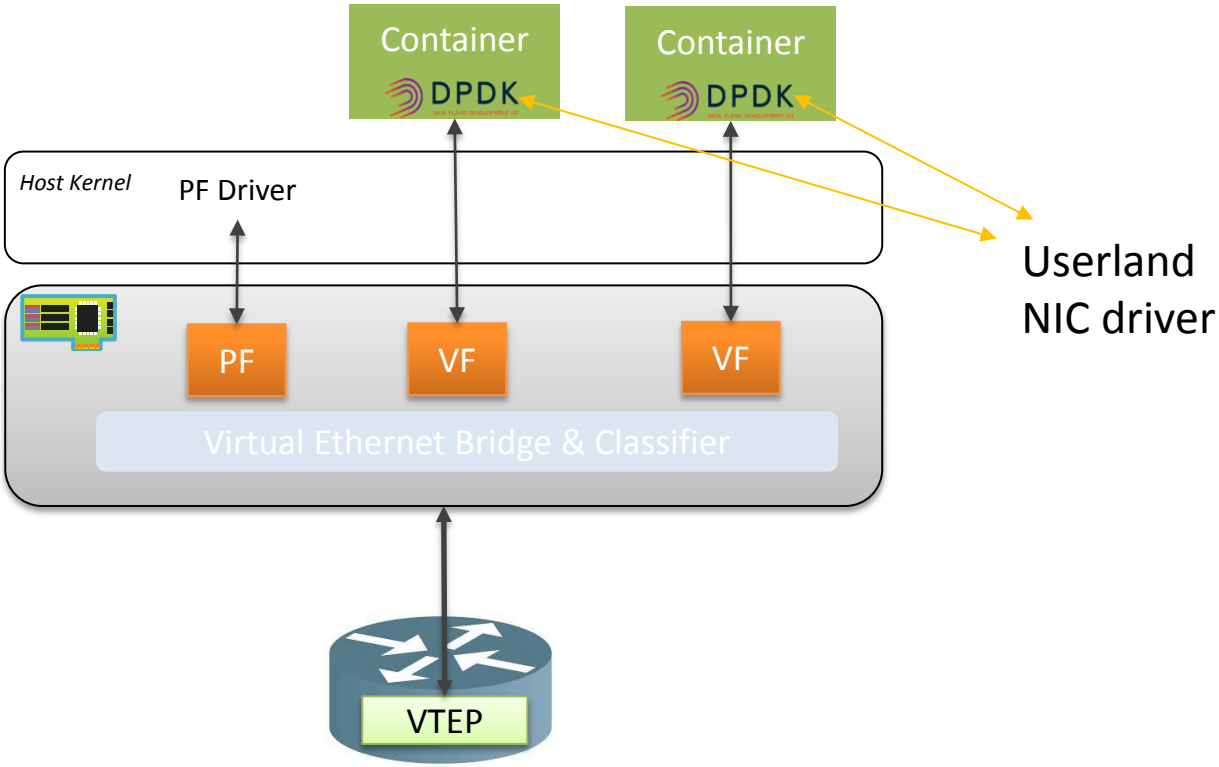
SR-IOV in Virtualization Technologies



SR-IOV in Kernel

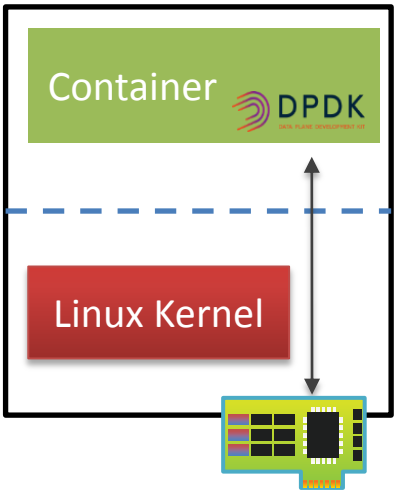


SR-IOV in Userland



SR-IOV in Userland

- Pros:
  - Line rate even with small packets
  - Low latency
  - HW-based QoS
- Cons:
  - # of VFs is limited (64 or 128)
  - Not flexible (in need of router or switch with support of VTEP)



Setup: SR-IOV in Userland

- Prepare VFs

```
$ echo 1 > /sys/bus/pci/devices/0000\:81\:00.0/sriov_numvfs
$ ./tools/dpdk_nic_bind.py --status
…
0000:81:00.0 '82599ES 10-Gigabit SFI/SFP+ Network Connection' if=eth1
drv=ixgbe unused=
0000:81:10.0 '82599 Ethernet Controller Virtual Function' if=eth5
drv=ixgbevf unused=
…
```

- Bind to vfio driver

```
$ modprobe vfio-pci
$ ./tools/dpdk_nic_bind.py -b vfio-pci 0000:81:10.0
```

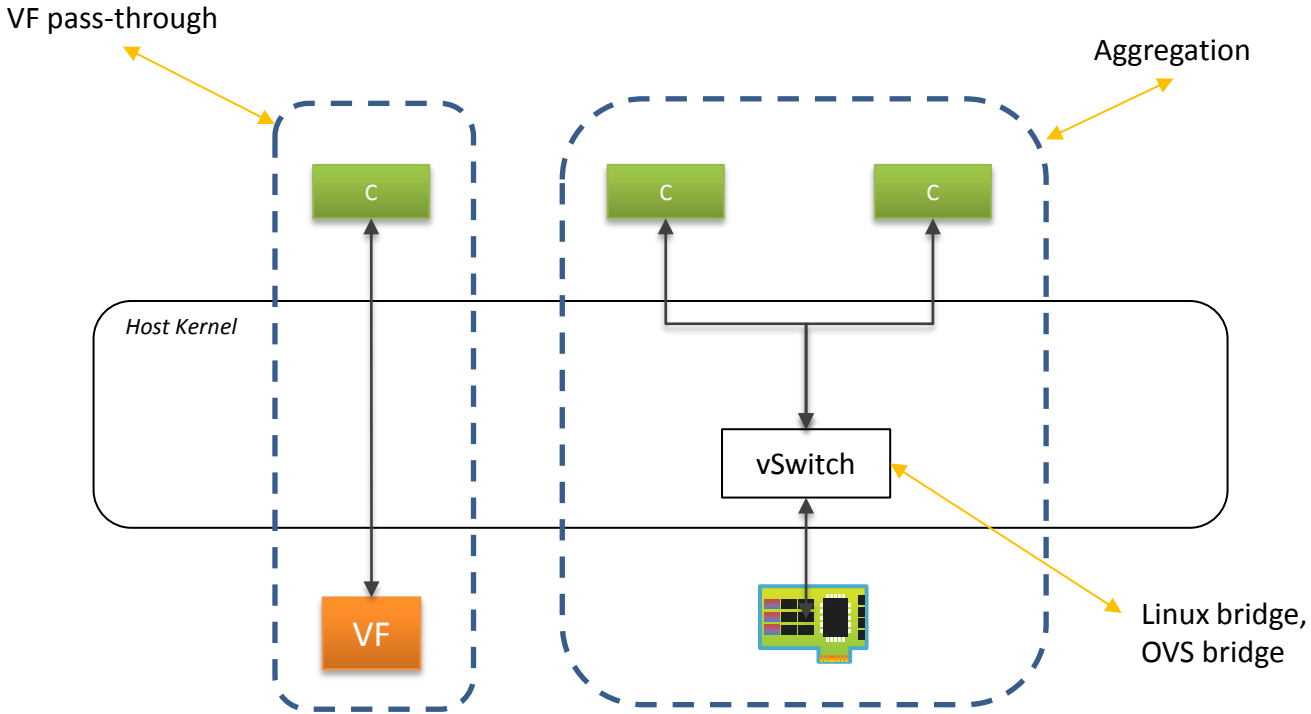
- Prepare hugetlbfs

```
$ mount -t hugetlbfs -o pagesize=2M,size=1024M none /mnt/huge_c0/
```

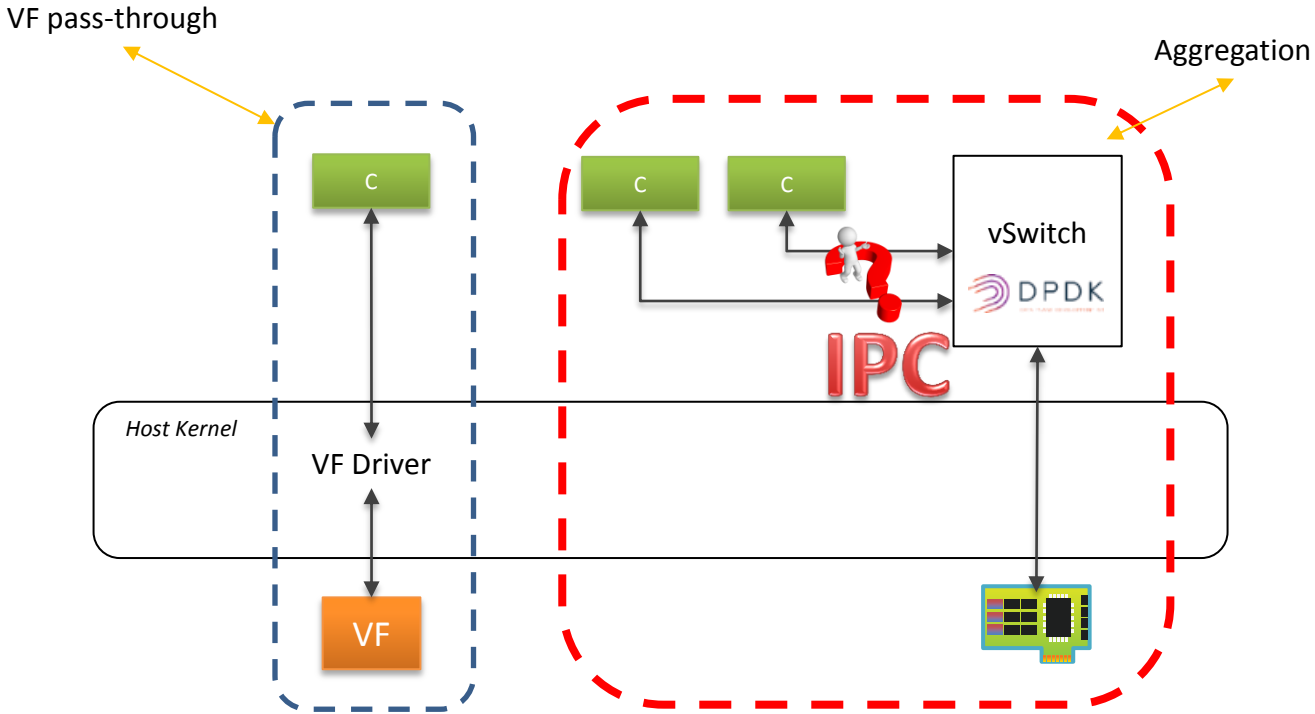
- Start container

```
$ docker run … -v /dev/vfio/vfio0:/dev/vfio/vfio0 -v
/mnt/huge_c0/:/dev/hugepages/ …
```

Container Networking Scenarios



Aggregation Scenario

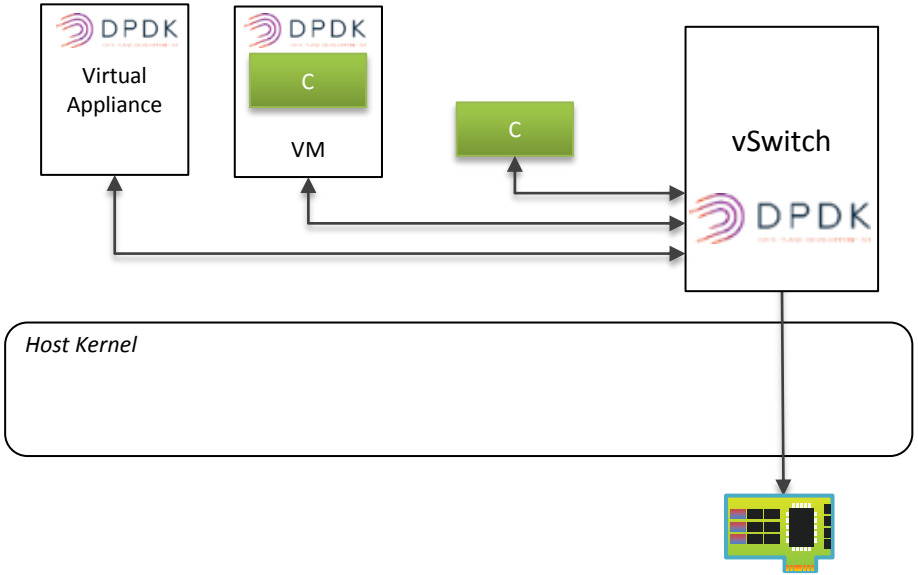


VIRTIO for IPC

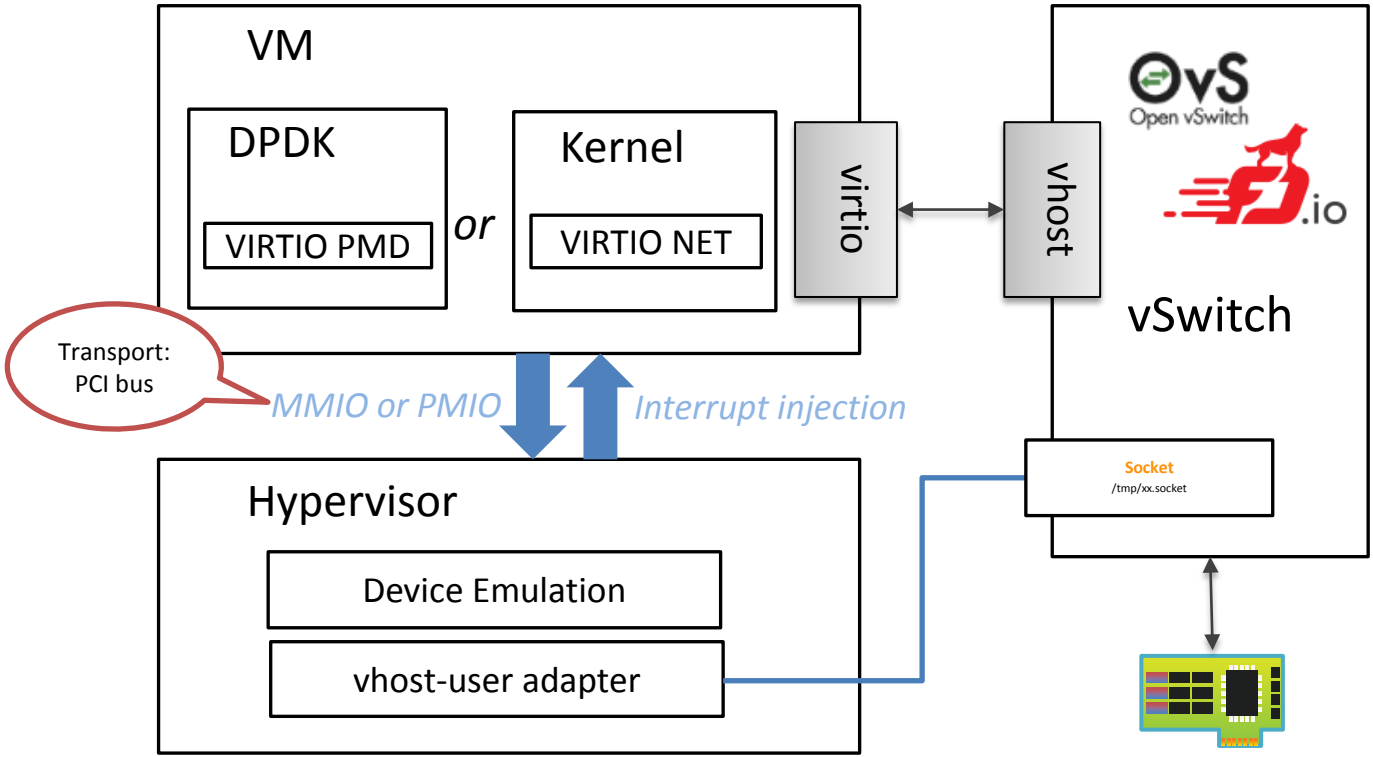
Why virtio for IPC?

- Performance
  - Bypass kernel (shm-based)
  - Smarter notification
  - Cache friendly
- Make best use of what already we have in DPDK

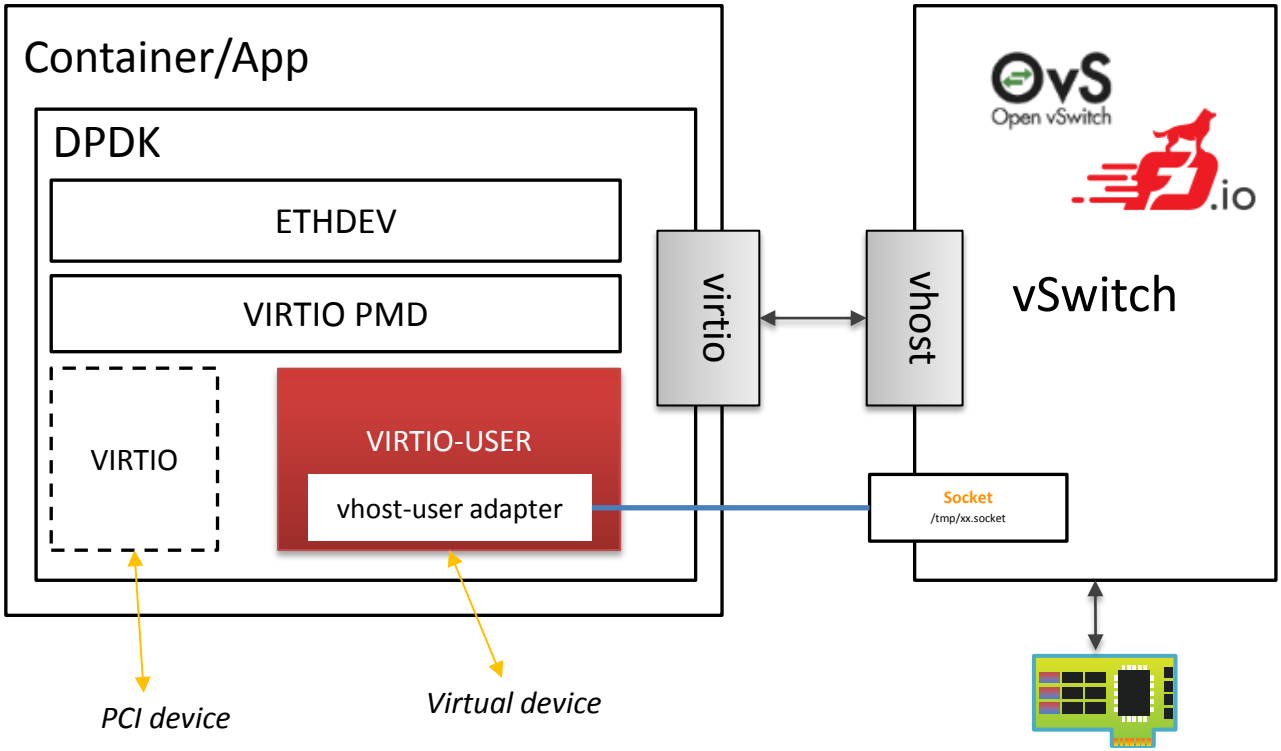
VIRTIO as Unified Interface



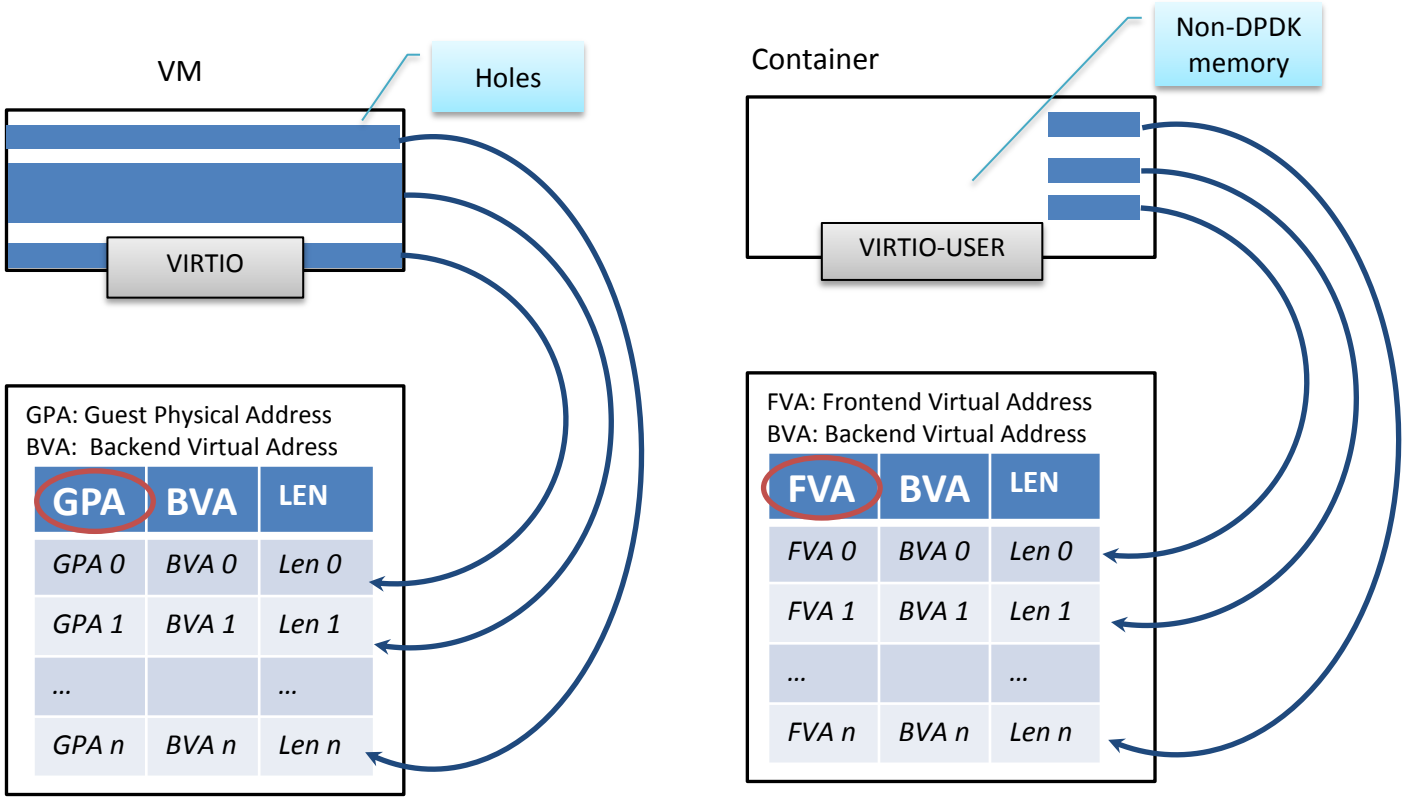
VIRTIO in VM


VM
DPDK
VIRTIO PMD
or
Kernel
VIRTIO NET
virtio
vhost
OvS
Open vSwitch
.io
vSwitch
Transport:
PCI bus
MMIO or PMIO
Interrupt injection
Socket
/tmp/xx.socket
Hypervisor
Device Emulation
vhost-user adapter

VIRTIO in Container



Address translation



Setup: VIRTIO in container

- Add a bridge and a vhost-user port in ovs-dpdk

```
$ ovs-vsctl add-br br0 -- set bridge br0 datapath_type=netdev
$ ovs-vsctl add-port br0 vhost-user-1 -- set Interface vhost-user-1
type=dpdkvhostuser
```

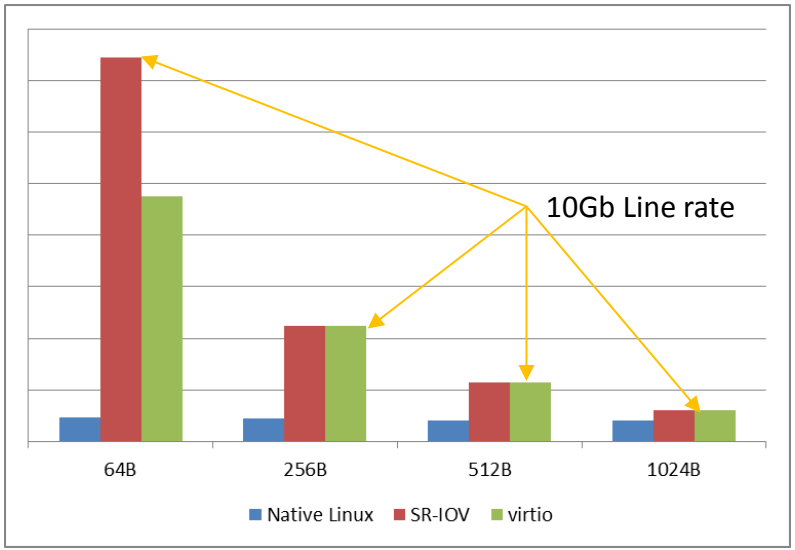
- Prepare hugetlbfs

```
$ mount -t hugetlbfs -o pagesize=2M,size=1024M none /mnt/huge_c0/
```

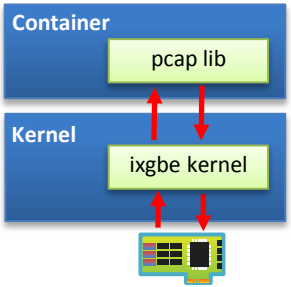
- Run container

```
$ docker run … \
-v /usr/local/var/run/openvswitch/vhost-user-1:/var/run/usvhost \
-v /mnt/huge_c0/:/dev/hugepages/ \
…
-c 0x4 -n 4 --no-pci  --vdev=virtio-user0,path=/var/run/usvhost \
…
```

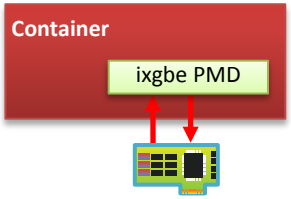
Performance Evaluation - throughput



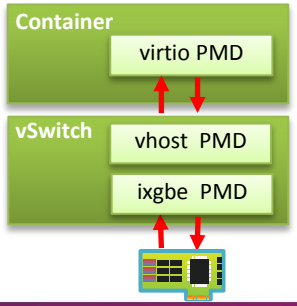
Case I:
Native Linux



Case II:
SR-IOV



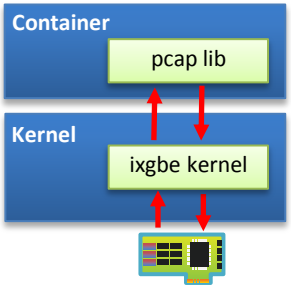
Case III:
VIRTIO



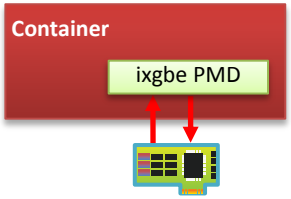
Performance Evaluation - latency

- For native Linux, ms level
- For the other two, us level
  - Polling mode
  - Batching
  - SIMD

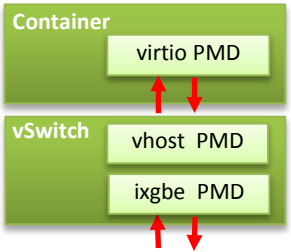
Case I:
Native Linux



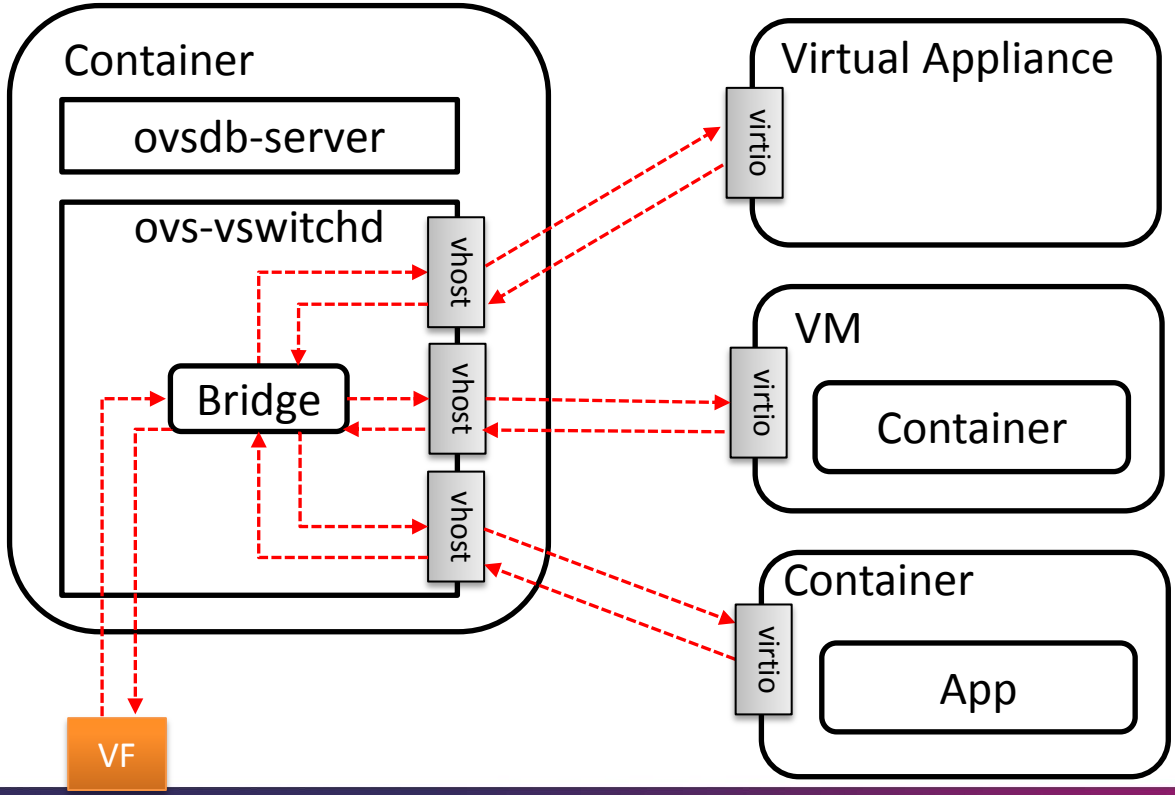
Case II:
SR-IOV



Case III:
VIRTIO



Use case


Container
ovsdb-server
ovs-vswitchd
vhost
vhost
vhost
Bridge
VF
Virtual Appliance
virtio
VM
virtio
Container
Container
virtio
App

Is everything OK to run DPDK in container?

DPDK efforts towards container

- Hugetlb initialization process
  - sysfs is not containerized, and DPDK allocates all free pages
    - Addressed by here, avoid to use *-m* or *--socket-mem*
- Cores initialization
  - When/how to specify cores for DPDK?
    - Addressed by here, avoid to use *-c* or *-l* or *--lcores*
- Reduce boot time
  - Addressed by here and here

Run DPDK in Container securely

- Dedicated hugetlbfs for each container
- Run without --privileged
  - DPDK needs privilege to do VA2PA translation
    - Use VA instead of PA (see here)
- DMA attack
    - Use VA as the IOVA for IOMMU table

Deterministic Environment

- Deterministic CPU env
  - Boot-time: disable timer / task scheduler
    - … default_hugepagesz=1G isolcpus=16-19 …
    - Reduce scheduling-clock ticks: *adaptive-tick* mode
  - Run-time: core-thread affinity
    - cpuset tool: taskset / numactl
    - cgroup.cpuset: cset / docker run … --cpuset-cpus …
  - BIOS setting: if necessary, disable *Hyper-Threading*

Deterministic Environment

- Deterministic cache env
  - Data Direct I/O (DDIO) technology
  - Cache Allocation Technology (CAT)
    - An example from here

```
$ pqos -e "llc:2=0x00003"
$ pqos -a "llc:2=8,9,10"
```

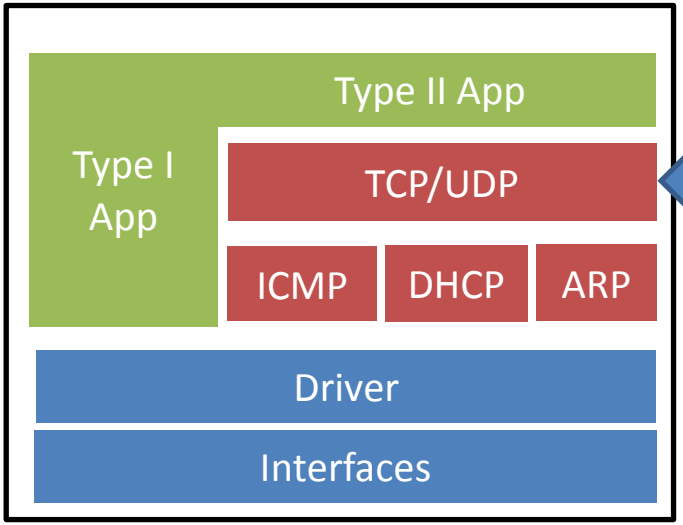
Noisy Neighbor Scenario*		
CAT	**Throughput (Mpps)**	LLC Occupancy (MB)
Not Present	**9.8**	4.5
Present	**15**	13.75

* DPDK IP Pipeline Application (Packet size = 64 Bytes, Flows = 16 Millions)

Impact on application development

How to Develop Apps?

- Type I: DPI, FW, LB, vSwitch/vRouter, …
- Type II: Applications in need of TCP/UDP stack



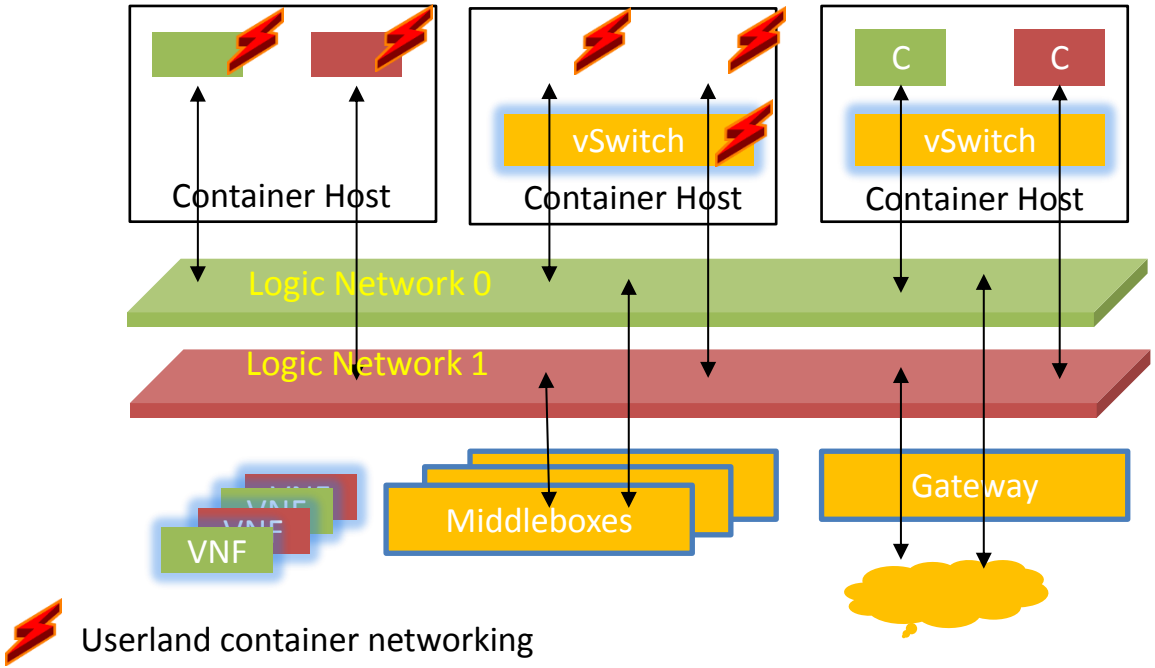
From scratch:

- mTCP
- LwIP
- TLDK

Ported:

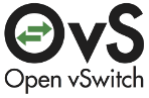
- Libuinet
- NUSE (libos)
- Linux Kernel Library

Transform Middleboxes with VNFs



Full Stack in Container

- Full stack
  - Hardware resource
  - Driver (network)
  - Under layer network facilities (optional)
  - Dependencies and application itself
- Benefits
  - Better isolation
  - Convenient for live migration



Snort-DPDK

TRex

Vortex

6WINDGate

Contrail

ScyllaDB

Future work

- Interrupt mode of virtio (to scale)
- Long path to handle VF interrupts in userland (low latency)
- Integrate with popular orchestrators

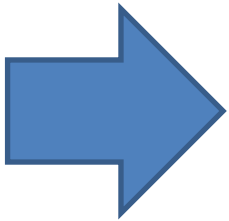
Summary

- Use DPDK to accelerate container networking
  - Userland SR-IOV
  - Userland virtio-user
- Compared to traditional ways, it provides
  - High throughput
  - Low latency
  - Deterministic networking

Q & A

Backup- How DPDK improves net?

- CPU affinity
- Hugepages
- **UIO**
- **Polling**
- Lockless
- **Batching**
- SSE/AVX



- High-throughput
- Low-latency
- Deterministic