Extending KVM Models Toward High-Performance NFV

Jun Nakajima, James Tsai, Mesut Ergin, Yang Zhang, and Wei Wang

14 October 2014

Legal Disclaimer

Agenda

- The Challenge
- Architecture Proposals for NFV for KVM
- Current Status and Summary

NFV Vision from ETSI

Source:
http://portal.etsi.org/nfv/nfv_white_paper2.pdf

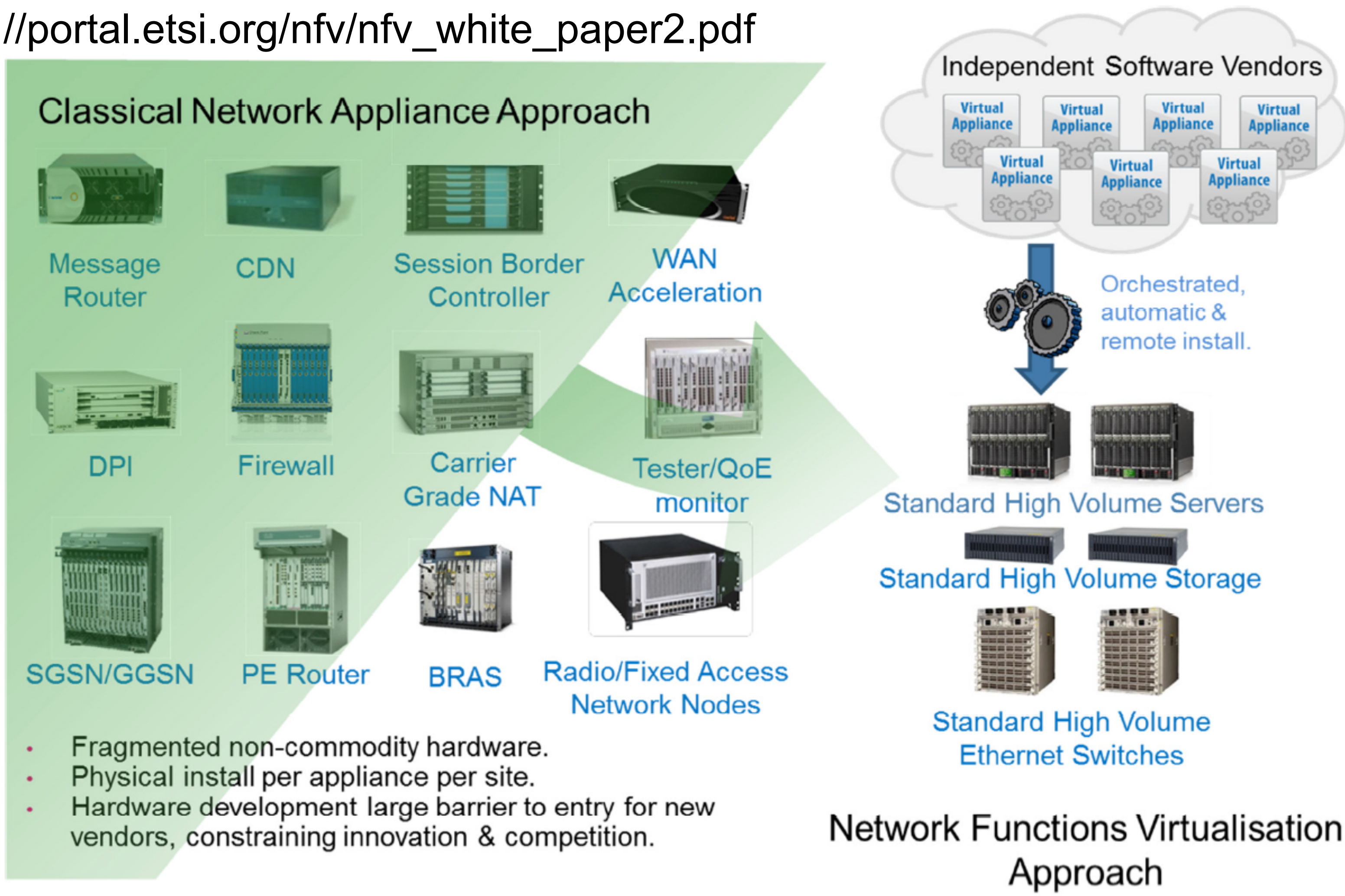


Figure 1: Vision for Network Functions Virtualisation

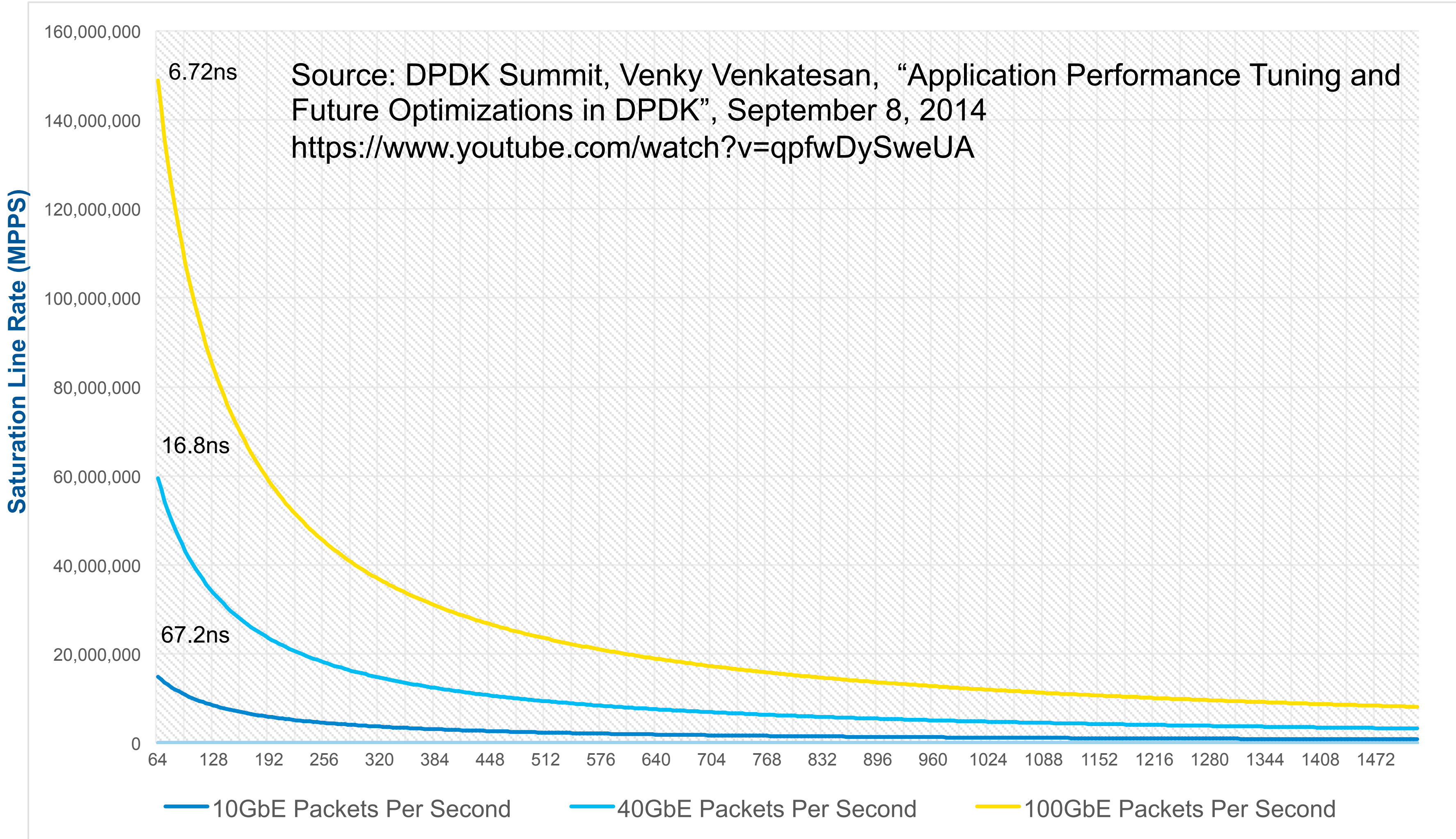
New/Different Requirements for NFV

Compared with Conventional Virtualization

- High performance across all packet sizes, including small packets (e.g. 64B)
- Real-time processing, including low latency and jitter
- RAS
- Security
- ...

Focus on Performance Topics Today

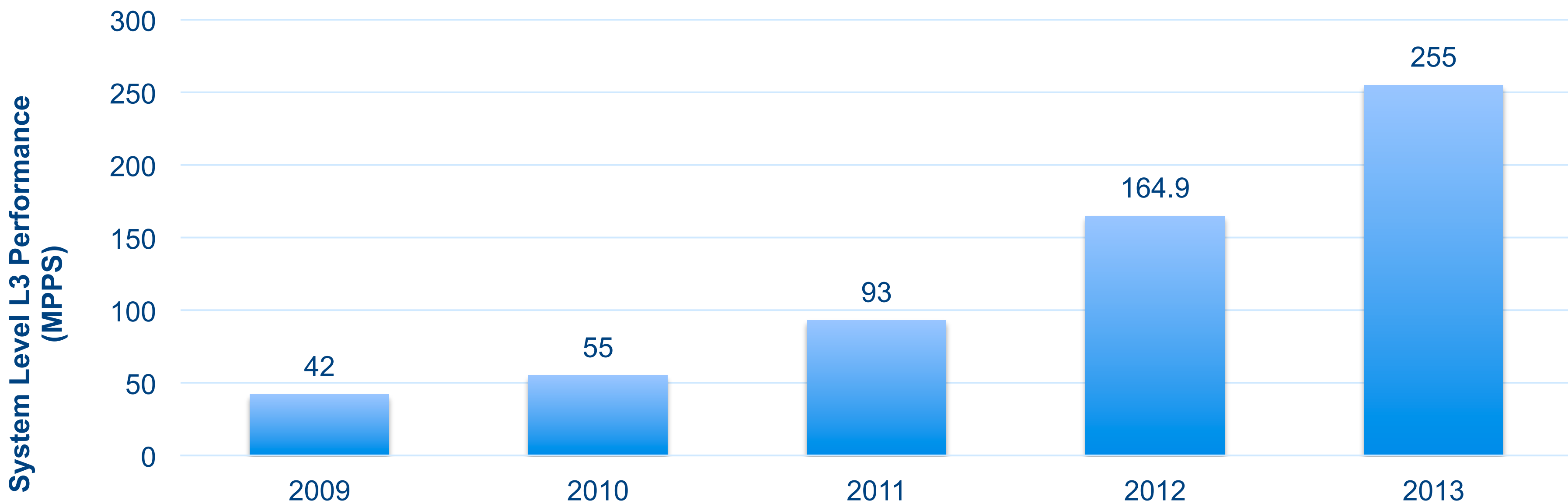
The Challenge



Intel® DPDK Performance

A snapshot of on different architectures

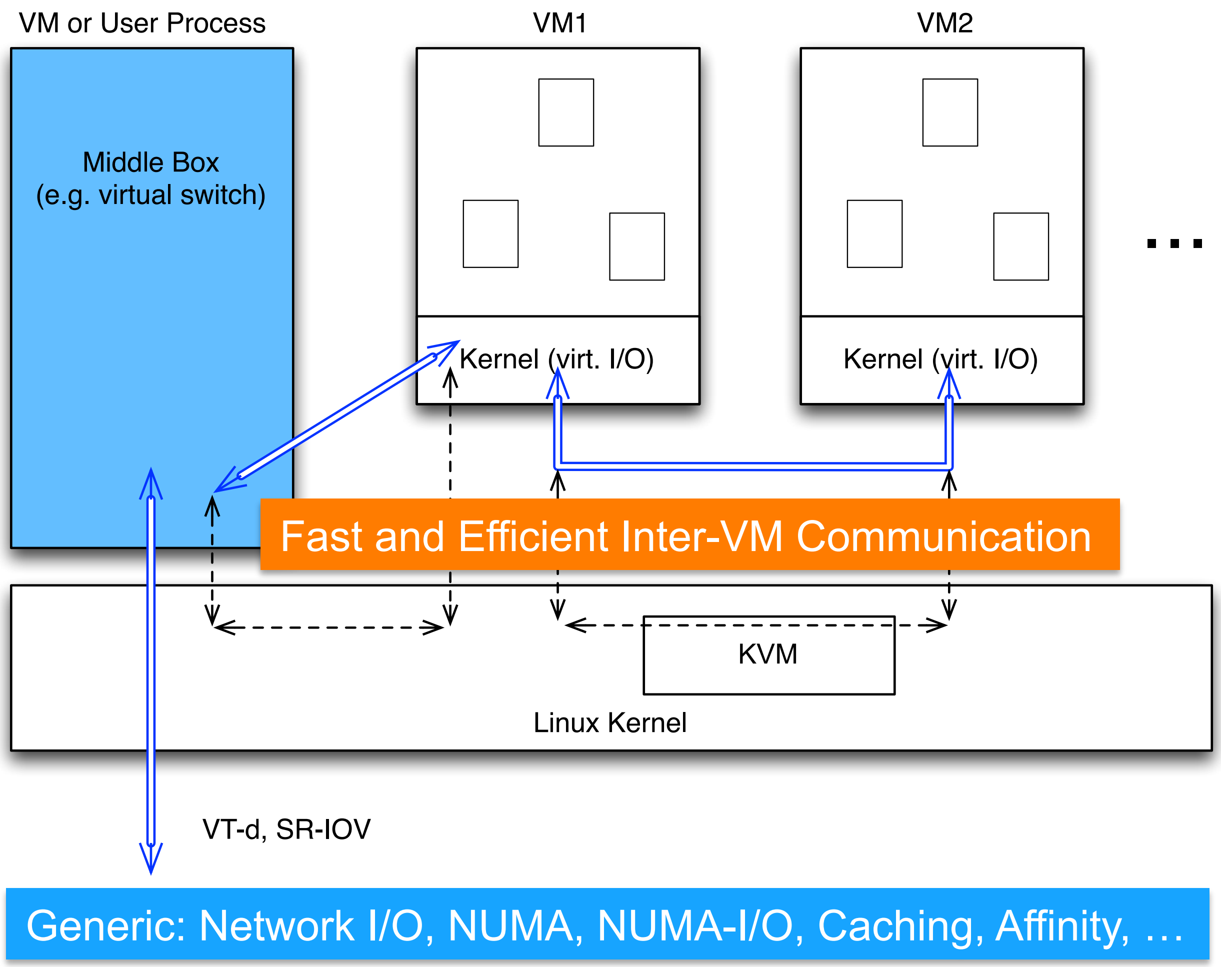
Platform Features			
	Integrated Memory Controller PCI-E Gen2	Data Direct I/O Integrated PCI-E Gen3 AVX (integer, 128-bit)	4x10 GbE NICs



Source: DPDK Summit, Venky Venkatesan, "Application Performance Tuning and Future Optimizations in DPDK", September 8, 2014
https://www.youtube.com/watch?v=qpfwDySweUA

Focus Areas for NFV Performance on KVM

Recall 67.2ns, 16.8ns, …



Why Inter-VM Communication?

- More cores
  - More middle boxes per socket, per server
  - Service chaining on server
- Lower latency
  - Inter-VM (i.e. intra-node) vs. Inter-node
- Higher Bandwidth
  - Memory (or cache) vs. PCIe bus

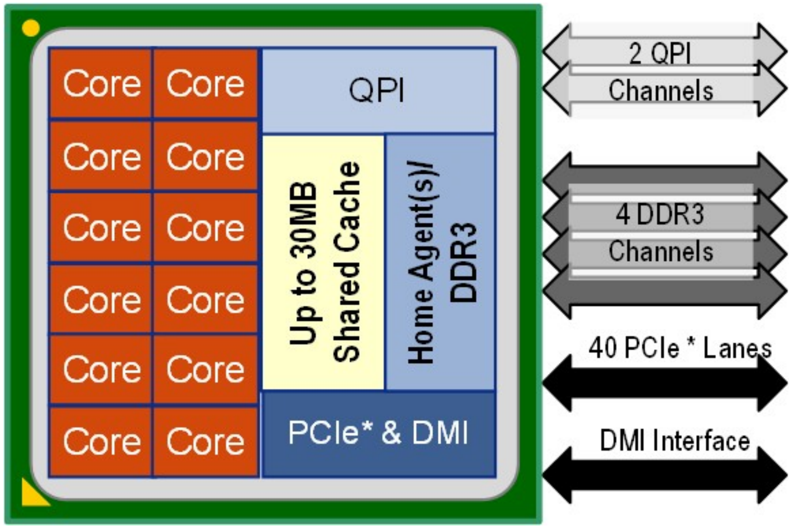
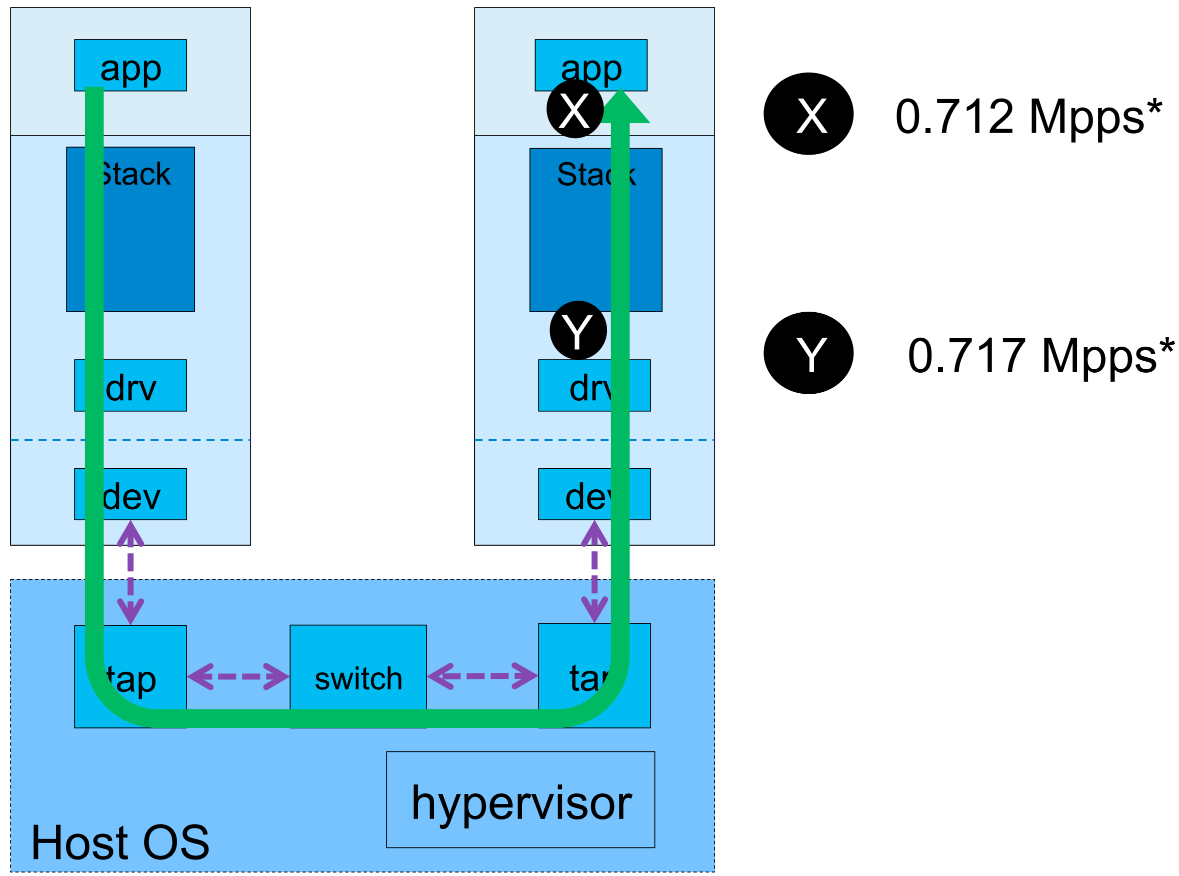


Figure 1. The Intel® Xeon® processor E5-2600 V2 product family Microarchitecture

Source (Figure 1.): https://software.intel.com/en-us/articles/intel-xeon-processor-e5-2600-v2-product-family-technical-overview

Inter-VM Communication on KVM

- Notifications for queue control
  - Kick, Door Bell
- Virtual Switch
- Packet Transmission
  - Copy, etc.
- Transitions
  - User-Kernel
  - Guest-Host



*Intel internal measurements

64B packets, virtio-net + vhost-net

Switching path can be a big performance bottleneck

Cost of Transitions/Isolation

Perspective of CPU Cycles

TSC Cycles (Haswell 3.2GHz), Round Trip*:

- User<->Kernel (System Call) in VM (on KVM)
  - E.g. getppid(): 1300 (≈ 400ns)
- Guest<->Host (Hyper Call)
  - E.g. Null Hypercall: 1500-1600 (≈ 500ns)

To reach Saturation Line Rate (10GbE):

- If system call/Hyper call is used for each 64B packet transmission, we would need:
  - > 6-7 Cores**
- 40GbE:
  - > 24-28 Cores?

Practically, those are rather lower bounds because batching is limited and actual packet processing in hypercalls overturns gain of batching.

*Intel internal measurements

**:400/67.2 = 5.9, 500/67.2 = 7.4

Agenda

- The Challenge
- Architecture Proposals for NFV for KVM
- Current Status and Summary

Solutions: Empower Guests in a Safe Way

Avoid hypervisor interventions

1. Move knowledge and control for inter-VM communication to VMs
2. Allow VMs to access other VMs to share or access memory in a safe way
   - Provide VMs with “Protected Memory View”
     - Mapping itself is provided by the hypervisor
3. Allow VMs to use low-latency notification mechanisms w/o VM exits or interrupts
     - E.g. MONITOR/MWAIT, Posted Interrupt

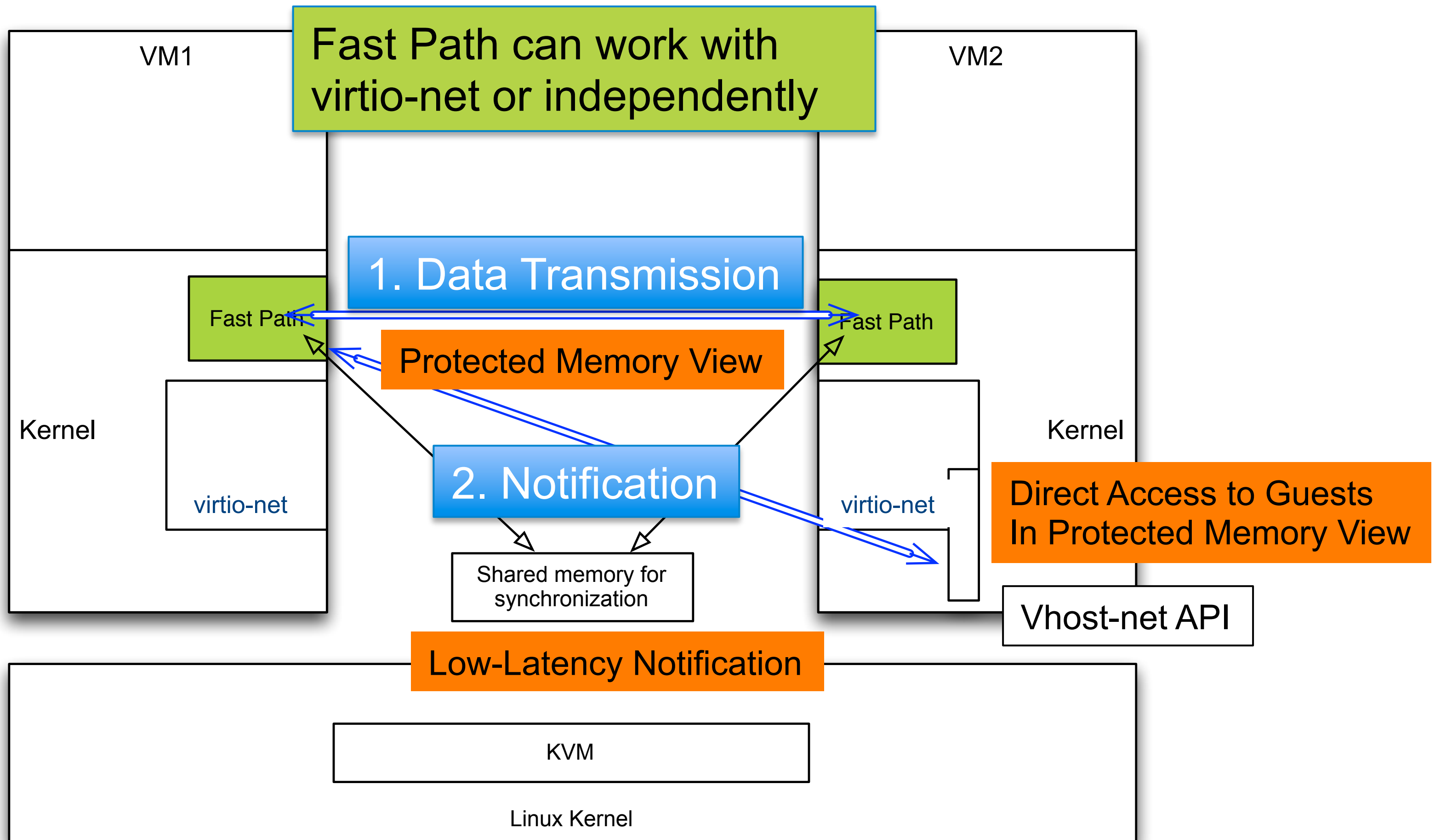
Example: vhost-net Functionality in Guests

vhost-user is already there

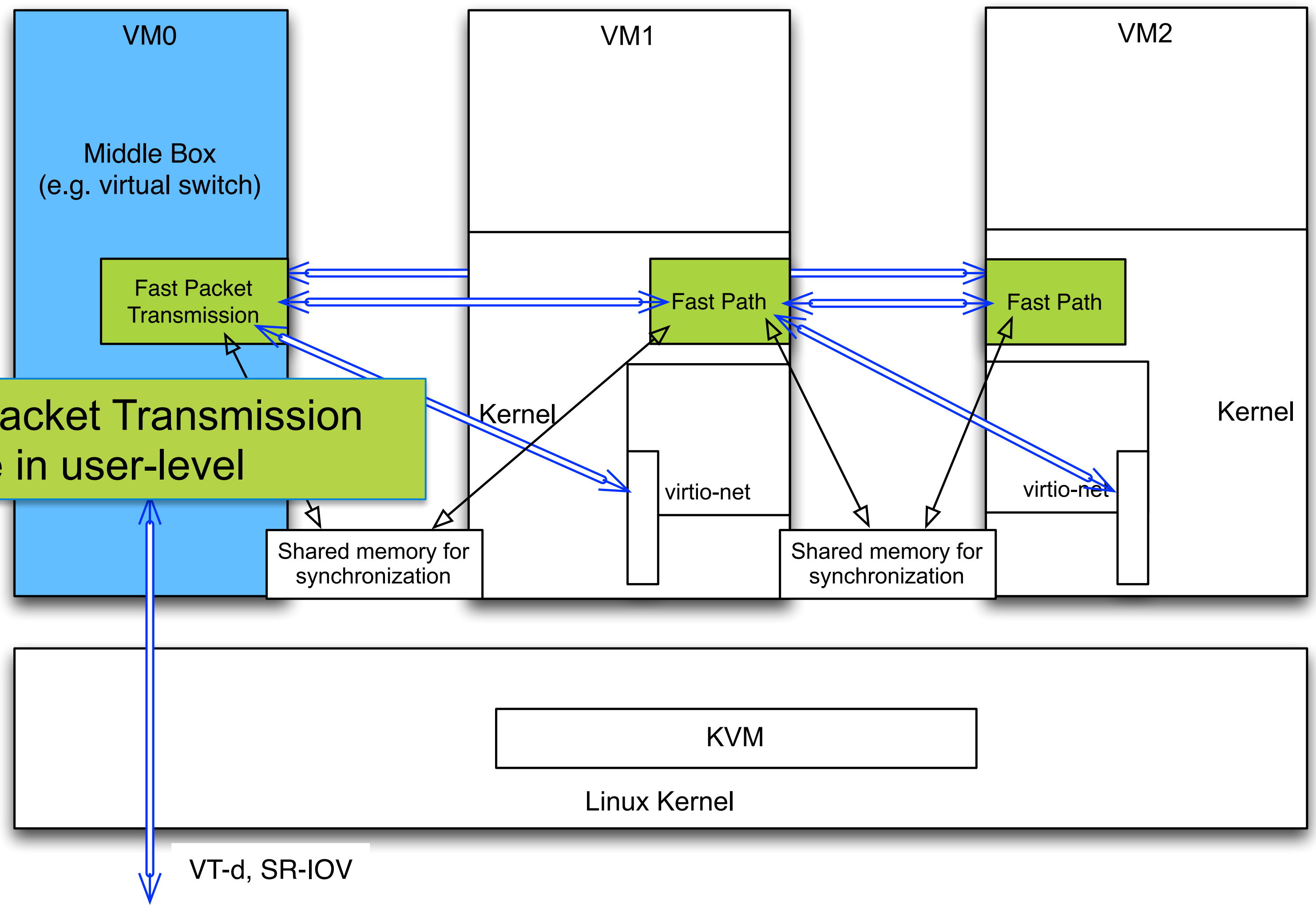
Motivation:

- Why does a kernel module need to know about data structures for PV drivers in guests?
  - Because we trust kernel or kernel modules only.
- What if we trust specific (part of) guests…
- Vhost-net in guest can avoid hypercalls if it can directly access destination guests (virtqueue, etc.)

High-Level Architecture for Fast Inter-VM Communication (w/o VT-d, SR-IOV)



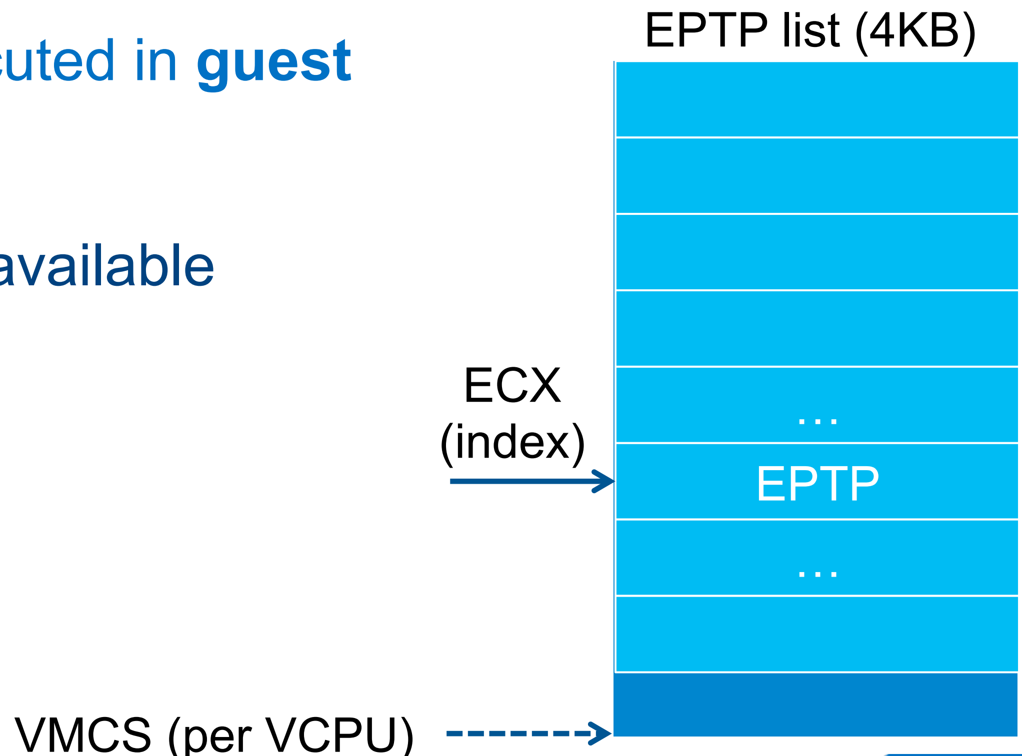
High-Level Architecture for Fast Inter-VM Communication (with VT-d, SR-IOV)



Fast Packet Transmission can be in user-level

Introducing VM Function 0: EPTP* Switching

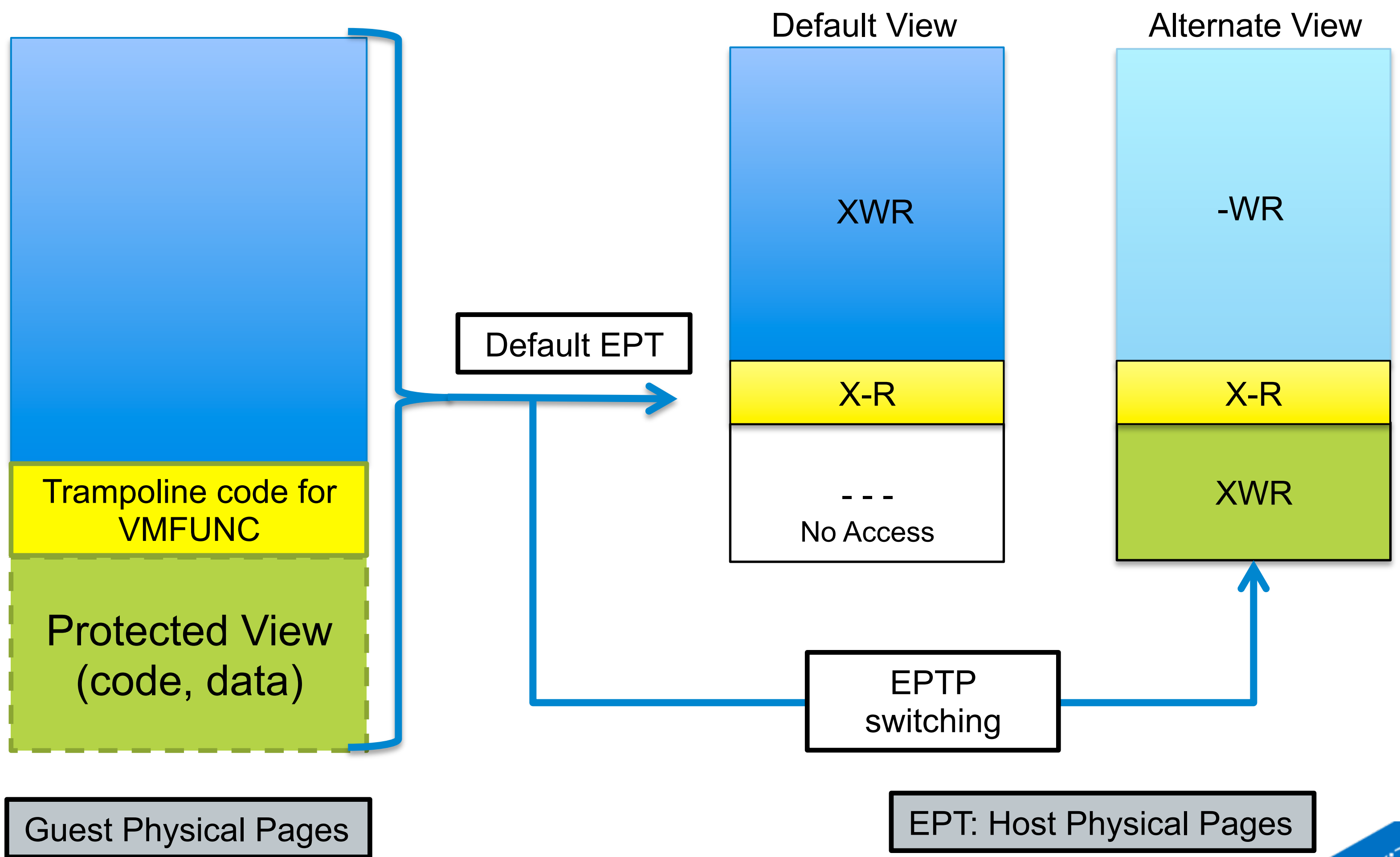
- VMFUNC instruction with EAX = 0
- Value in ECX selects an entry from the EPTP (Extended-Page-Table Pointer) list
- Available in Ring 0-3, executed in **guest**
  - No VM exit
  - Can be virtualized if not available



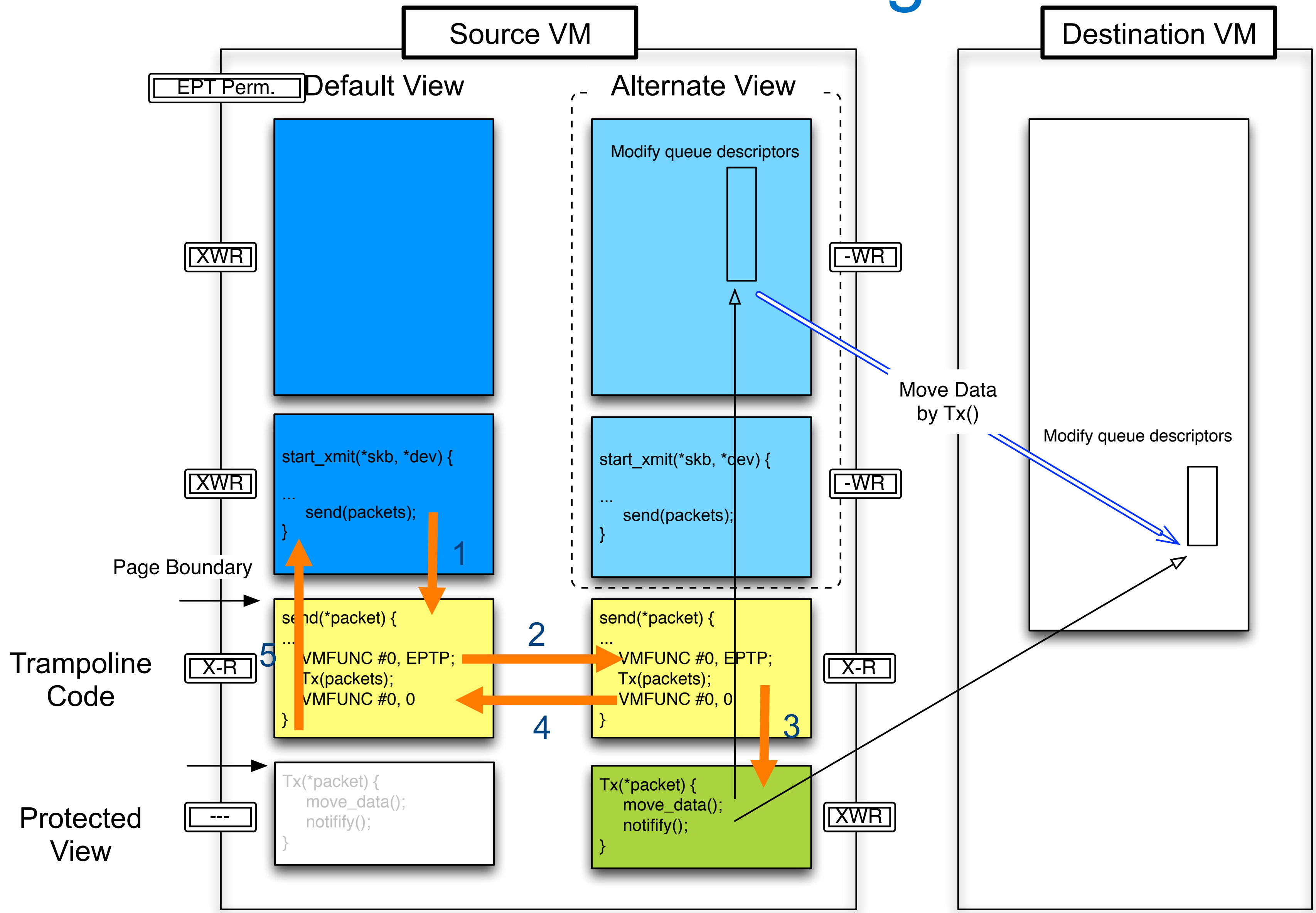
*:Extended-Page-Table Pointer

EPTP Switching and Trampoline Code

- VMFUNC executed outside Trampoline Code will cause EPT violation at next instruction
- Hypervisor needs to restore Default EPT to deliver virtual interrupts



More Details: Transmitting Packets



Low-Latency Notification

Known methods

- Posted Interrupt
  - Deliver virtual interrupts on destination guests w/o VM exits.
  - Already supported by KVM
    - Still requires VM exit on source guest
- MONITOR/MWAIT (Energy-Efficient Polling) between guests
  - The feature is not advertised on KVM today
  - Use variables on shared memory between source and destination
- PAUSE Loop (Polling) between guests
  - Lowest latency, but not energy efficient

In practice, combine Interrupt and Polling (like NAPI)

Practices for Performance
General

Minimize impact of TLB misses, cache misses:

- Large pages (both guest, EPT, VT-d), NUMA, IO-NUMA, Data Direct I/O
  - E.g. LIFO memory pool
- Zero-copy
  - E.g. Add source buffers mapping to EPT of destination
    - If EPT PTEs were not valid, no INVEPT is required

Practice for Performance

EPTP Switching

getppid() in VM: 1300 (≈ 400ns)

Null Hypercall: 1500-1600 (≈ 500ns)

Frequency of VMFUNC operation:

- Cost of VMFUNC is about 150 TSC cycles (Haswell, 3.2 GHz)*
  - Around 50ns, and sensitive to TLB, caches
  - Recall 67.2ns, 16.8ns, …

To reach Saturation Line Rate (10GbE):

- If VMFUNC is called for each 64B packet transmission, we
  - > 1-2 Cores (100ns for round-trip)
- 40GbE:
  - > 4-8 Cores?

Practically, those are rather lower bounds because batching is limited and actual packet processing overturns gain of batching.

- The cost of VMFUNC would be relatively small, and it would provide scalable performance

*Intel internal measurements

Security Consideration

- Trampoline Code is loaded by the guest, but the EPT permission (X-R) is set by KVM
  - Should be signed together with the code in the Protected View in advance
- The set of pages (in Destination VM) accessed by code in Protected View need to be checked and added by KVM
  - In a way, code in Protected View is an extension of the KVM/ hypervisor running in controlled environment (still in VXM non-root mode)

Agenda

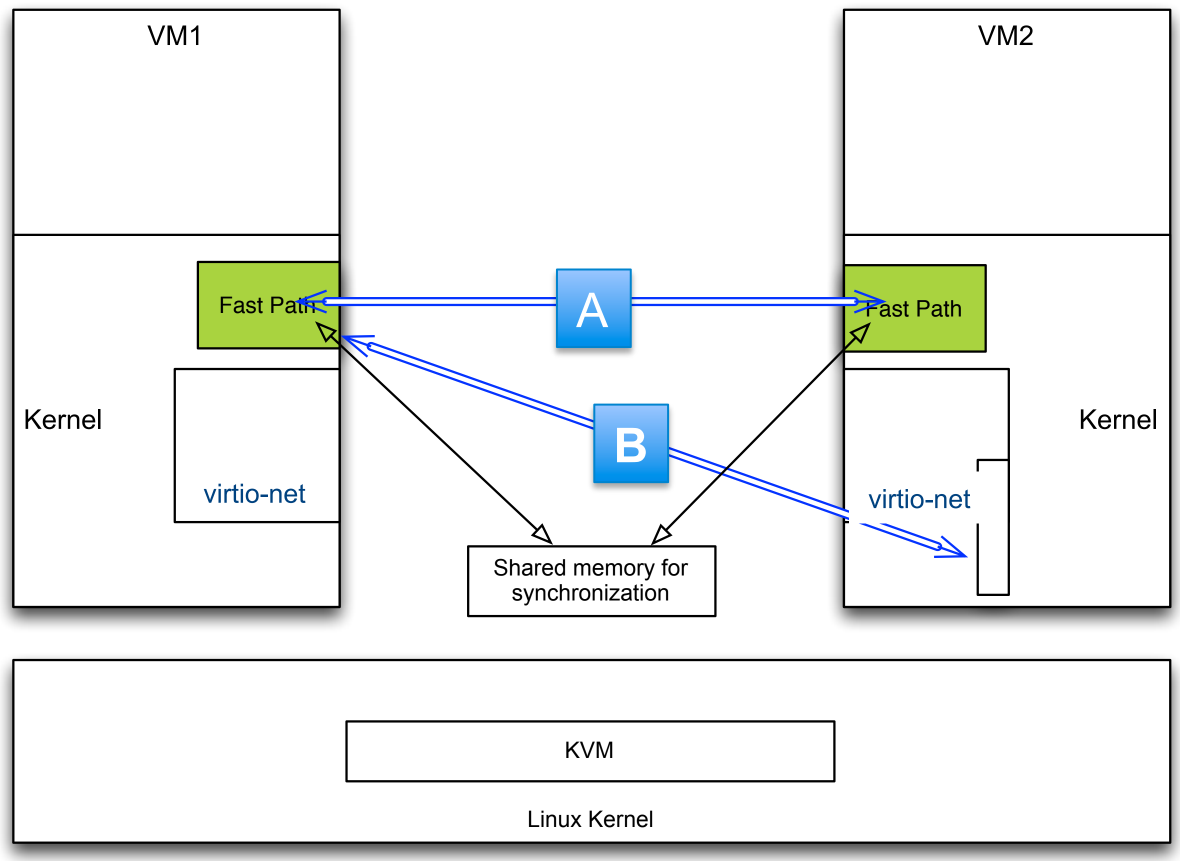
- The Challenge
- Architecture Proposals for NFV for KVM
- Current Status and Summary

Current Status

PoC

PoC in progress:

- Measured cost of VMFUNC, memory bandwidth
- Enabled and measured latency of MONITOR/MWAIT in guests
- Measuring path A
- Working on path B



Summary

Benefits of the Architecture:

- Contain knowledge and control for Inter-VM communication in guests
- Allow KVM to enable more optimization and customization for guests to handle high network loads efficiently
  - More efficient and scalable than existing ones
- Work with direct I/O assignment as well

Next Step:

- Complete PoC and get more data

Backup

#VE: Virtualization Exception

- Can occur only in guest (vector 20)
- Some EPT violations can generate #VE instead of VM exits (controlled by hypervisor)
- Can virtualized if not available