NetVM: High Performance and Flexible Networking using Virtualization on Commodity Platforms

Jinho Hwang†, K. K. Ramakrishnan*, Timothy Wood†

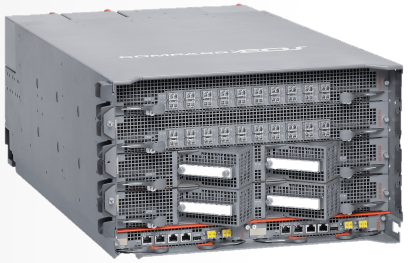
†The George Washington University

*Rutgers University

High Performance Networking

- Line Rate High Performance Machines

Special HW + SW
Very Expensive



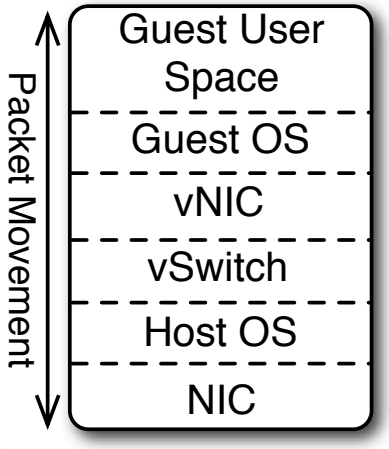
PacketShader
[Han:Sigcomm:2010]
GPU

NetMap
[Rizzo:ATC:2012]
Commodity
Servers

Intel DPDK
Commodity NICs

Data center is virtualized

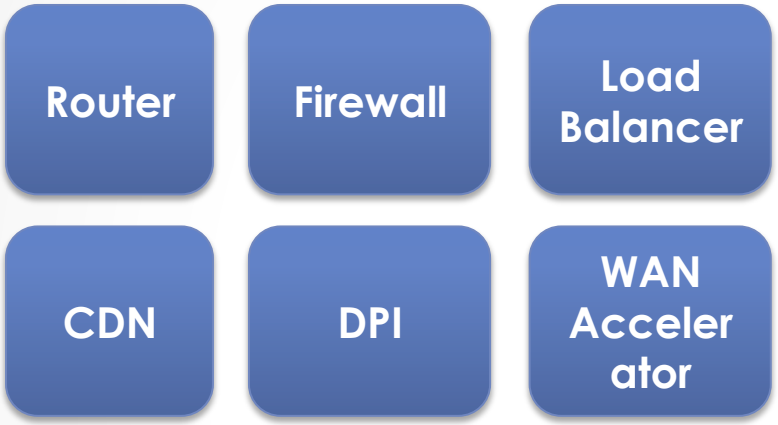
Performance in Virtualized Platforms?



Need to overcome virtualization overheads

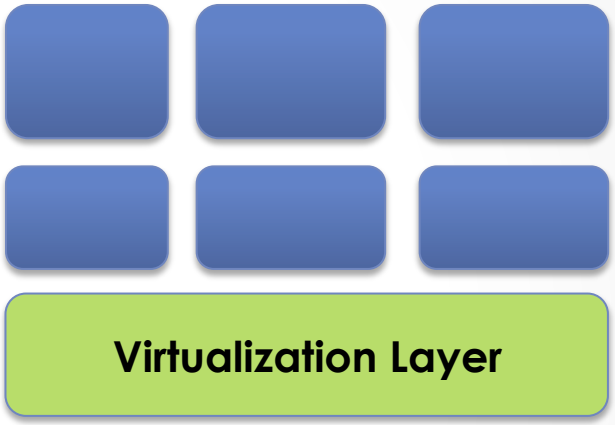
Network Functions Virtualization (NFV)

Existing Network Functions



6 Machines x Power x $HW x $SW

NFV-enabled Network Functions

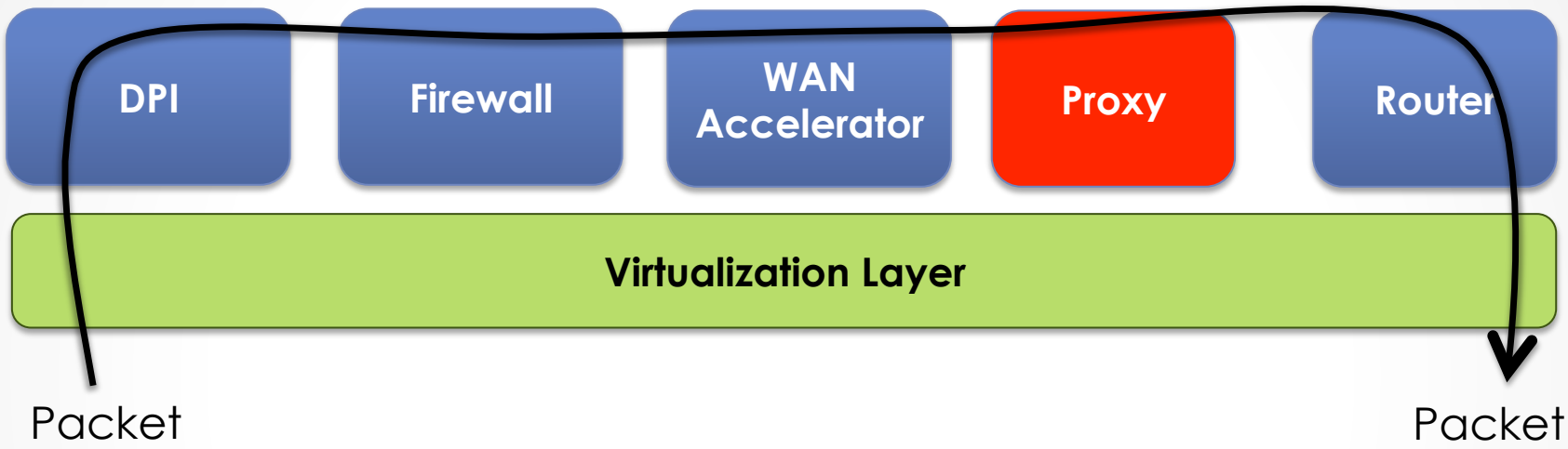


1 Machine x Power x $SW
+ Flexible deployment

Need to provide high speed and low latency

Chained Functionality

- Functions are often sequential



Need High Speed
Inter-Function (VM) Communication

So, What Has NetVM Done?

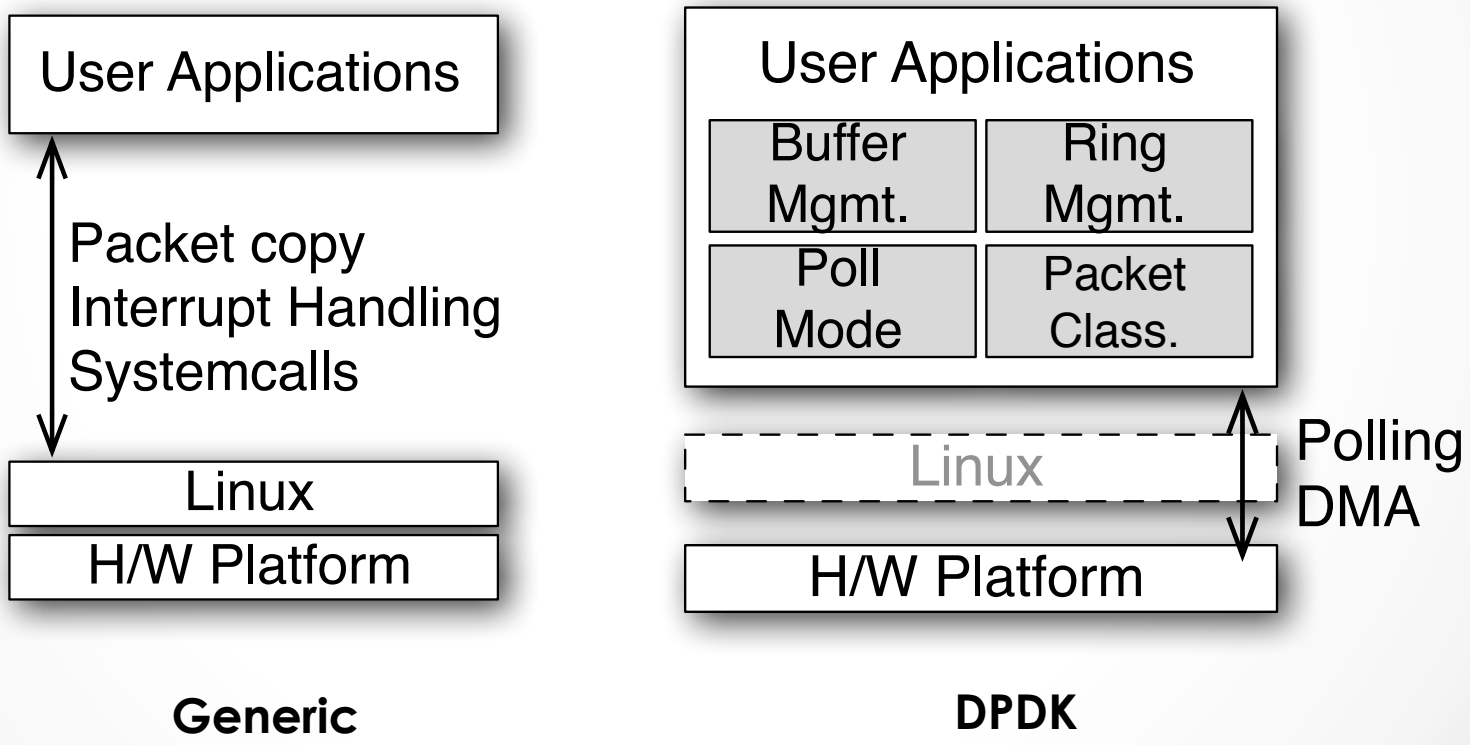
- NetVM is a platform for running complex network functionality at line-speed (10Gbps~) using commodity hardware
  - ✓ A virtualization-based high-speed packet delivery
  - ✓ Memory sharing framework for network data
  - ✓ A hypervisor-based switching
  - ✓ High speed inter-VM communication
  - ✓ Security domains

Outline

- Motivations & Contribution
- Background
- System Design
- Evaluation
- Conclusion & Future Work

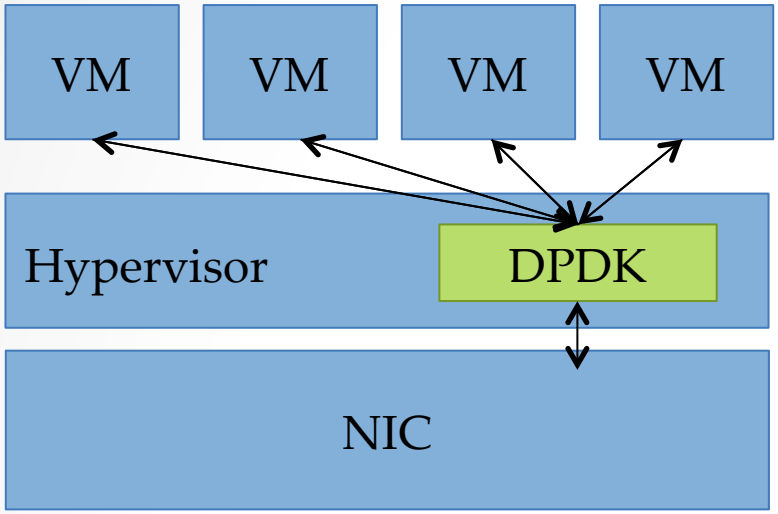
COTS and DPDK (Data Plane Development Kit)

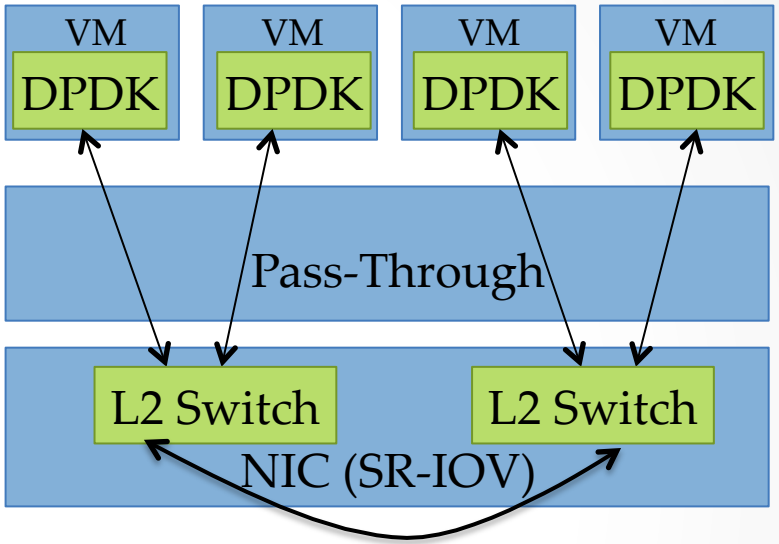
- COTS = commercial, off-the-shelf servers
- DPDK is a set of optimized "***user space***" software libraries and drivers that can be used to accelerate packet processing on Intel architecture



DPDK with Virtualization

- Two Architectural Variations





SR-IOV = Single Root I/O Virtualization

- Flexible(dynamic) Configuration
- Control over packet switching
- Control over load-balancing
- Has more overhead

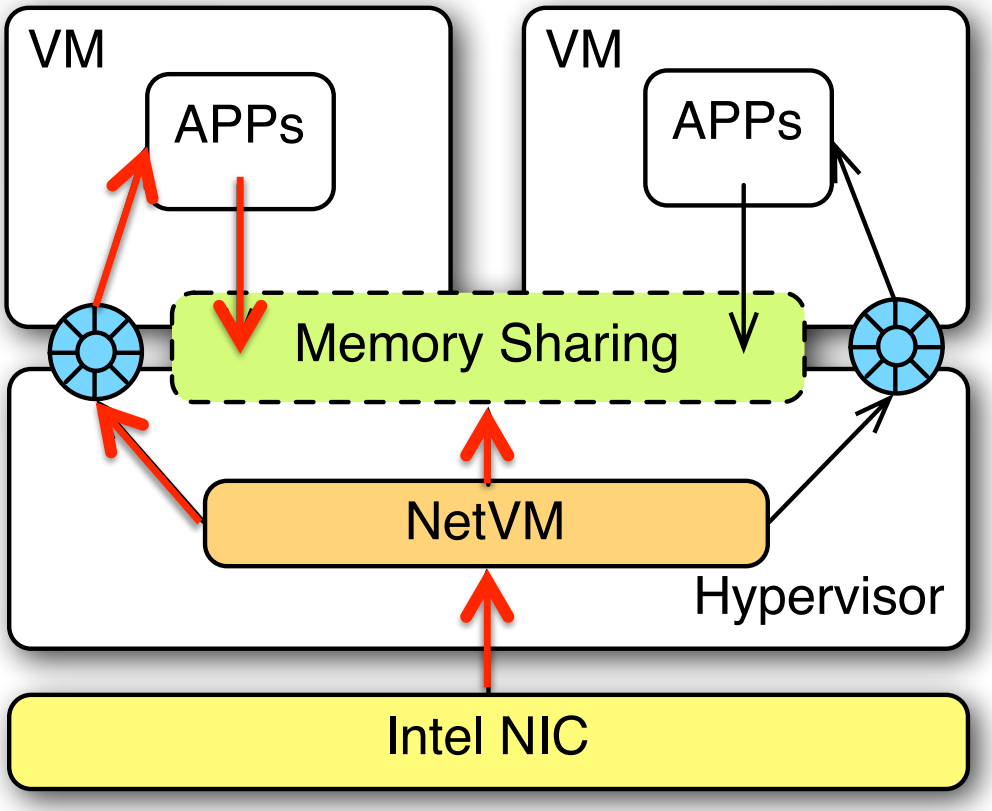
- Max 63 virtual functions (tx/rx)
- Static configuration
- Inter-VM switch is limited per port
- No control over packet switching
- No control over load-balancing

Neither can achieve full line-rate network speed in VMs

Outline

- Motivation & Contribution
- Background
- System Design
- Evaluation
- Conclusion & Future Work

NetVM System Overview



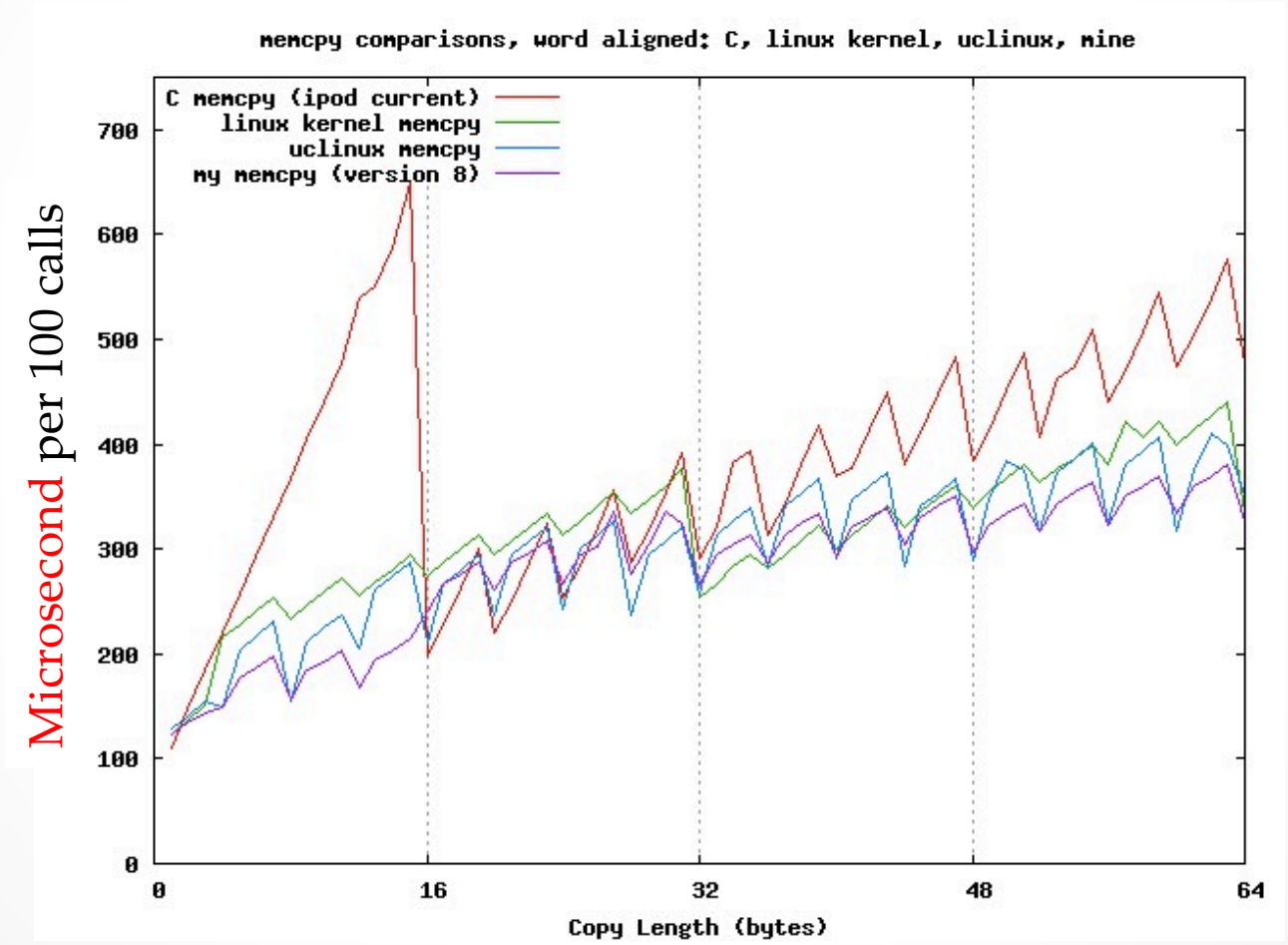
- NetVM (with DPDK) runs in hypervisor User Space
- For zero packet copy, memory is shared for network data
- Each VM has its own ring to receive/transmit a packet descriptor

System Design Challenges

1. Zero-copy
2. A huge-page sharing
3. Lockless & non-uniform memory architecture (NUMA) aware design
4. Security domains

Memory Copy Hazard

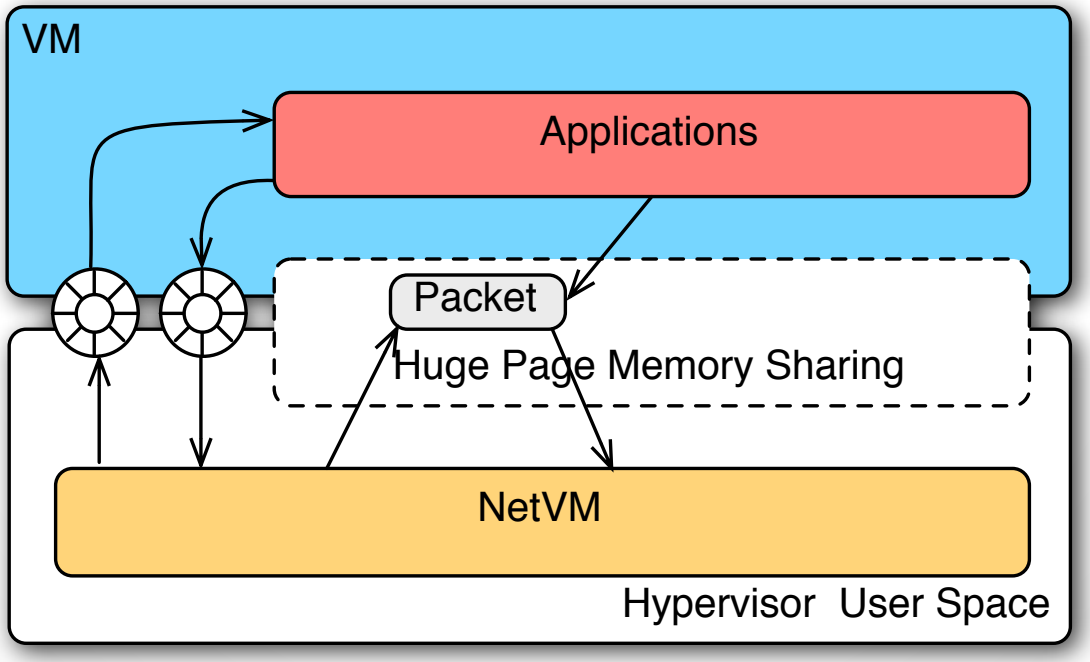
- 10Gbps = 14,880,952 packets/s (64 byte)
- Each packet should be processed in 67 ns



http://diffenbach.org/rockbox/memcpy/memcpy.comparison.0.64.html

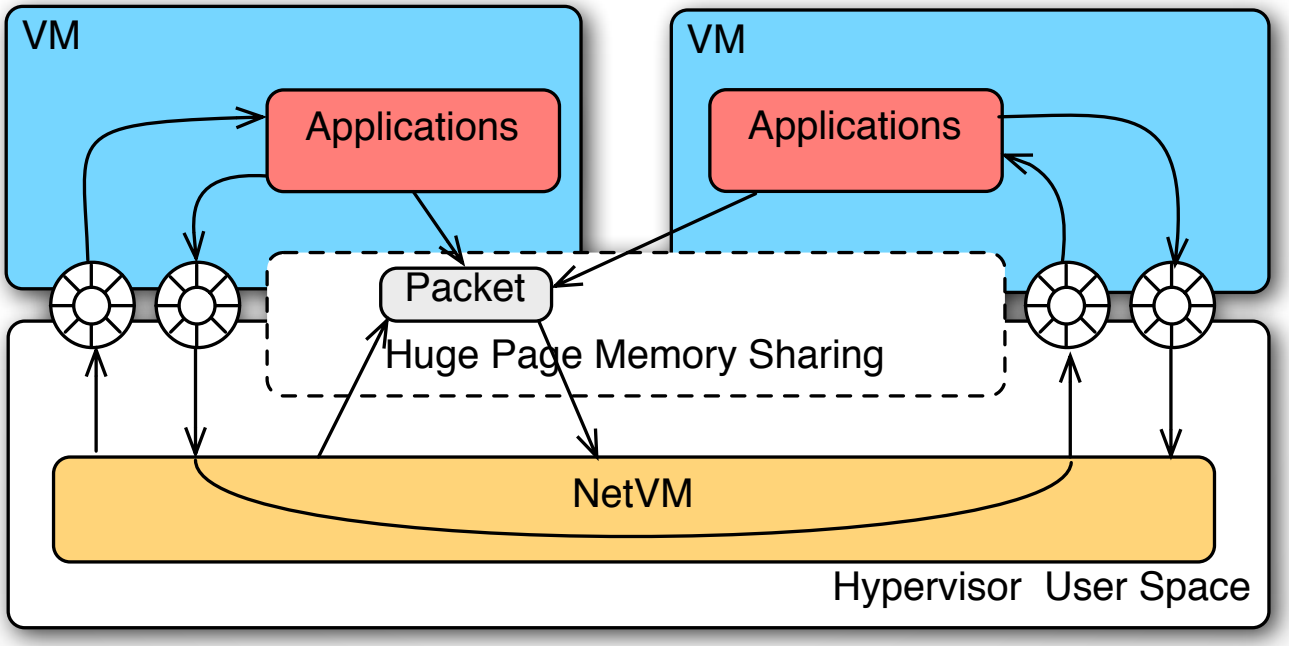
Zero-Copy Packet Delivery

- Packet directly DMAs into huge page memory
- Applications in VM receive references (location) via shared ring buffers
- Packet contents can be manipulated
- Applications decide an action: chain to another VM, send out, discard



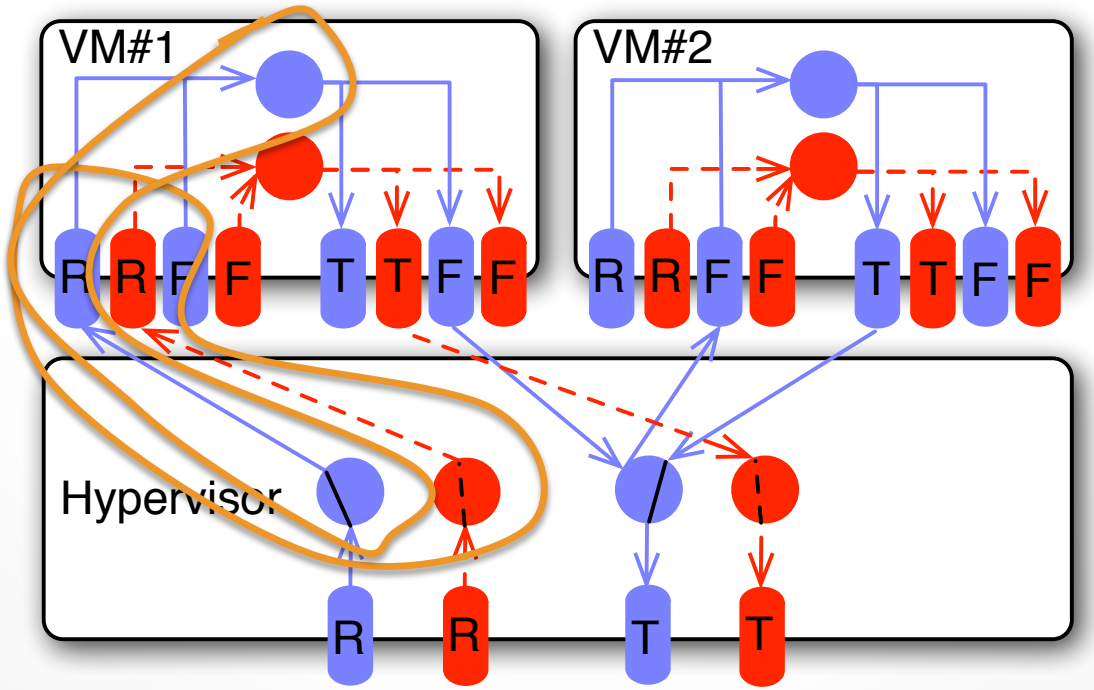
Chained Packet Delivery

- Packets are processed in a sequence/parallel
- Applications in VMs pass packet references to other VM
- Only one application can process a packet at a time (read/write)



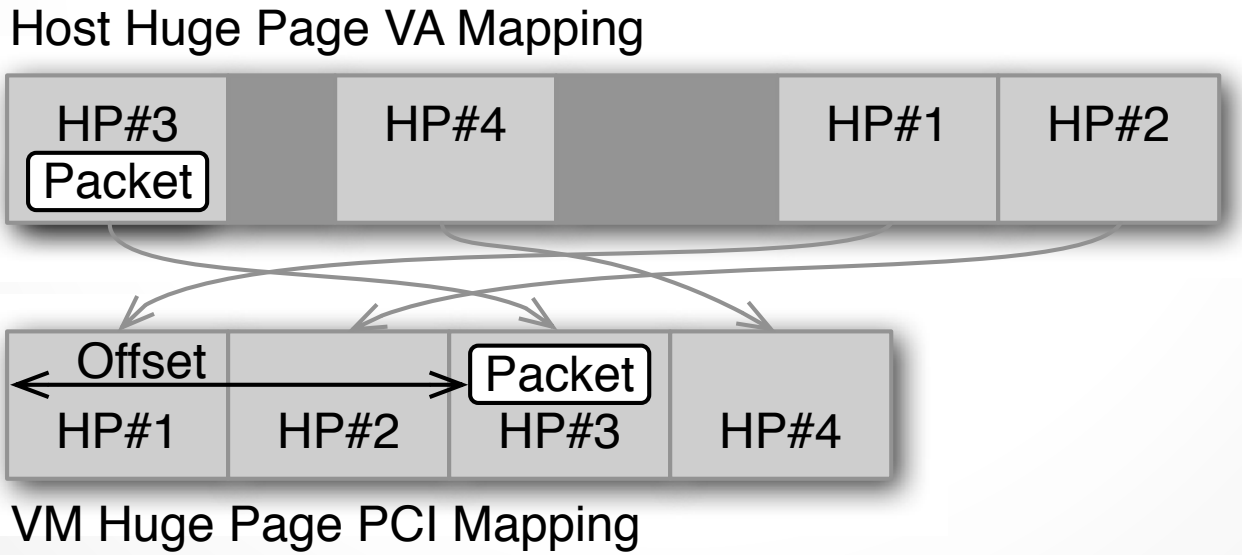
Lockless & NUMA-Aware Design

- **Locks** significantly degrade packet transmission performance
  - Core-queue matching (between horizontal cores) and data structure separation (between vertical cores)
- With **NUMA**, reading local and remote memory alternatively invalidates local cache → cache misses
  - This is fraught with Intel Direct I/O (Sandy-bridge)



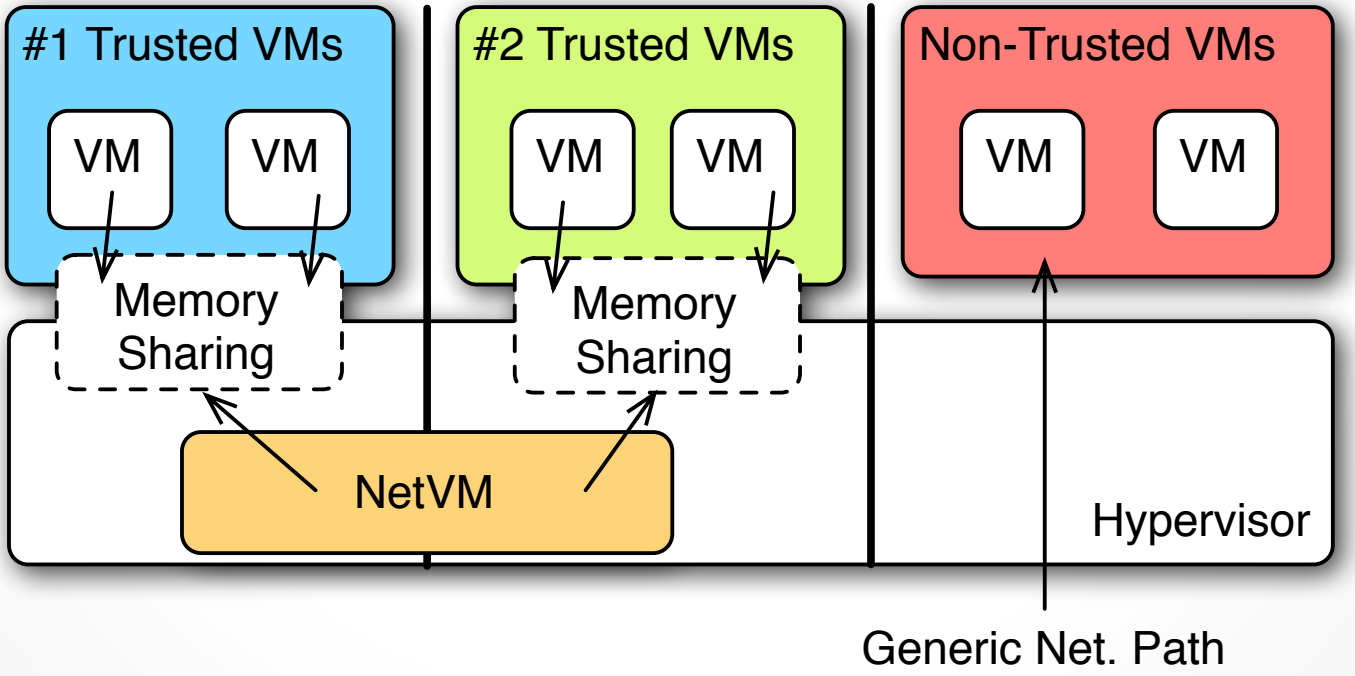
Huge Page VA Mapping

- Hypervisor is often unable to allocate consecutive physical huge pages and map to consecutive virtual addresses
- But, VM can see a mapped huge memory, which makes huge pages consecutive
- Offset Calculation (no looping)
  - ✓ Pre-calculate locations of each huge page
  - ✓ When a packet is received, we can calculate an offset in one programming line without looping (only bit operation and array index)



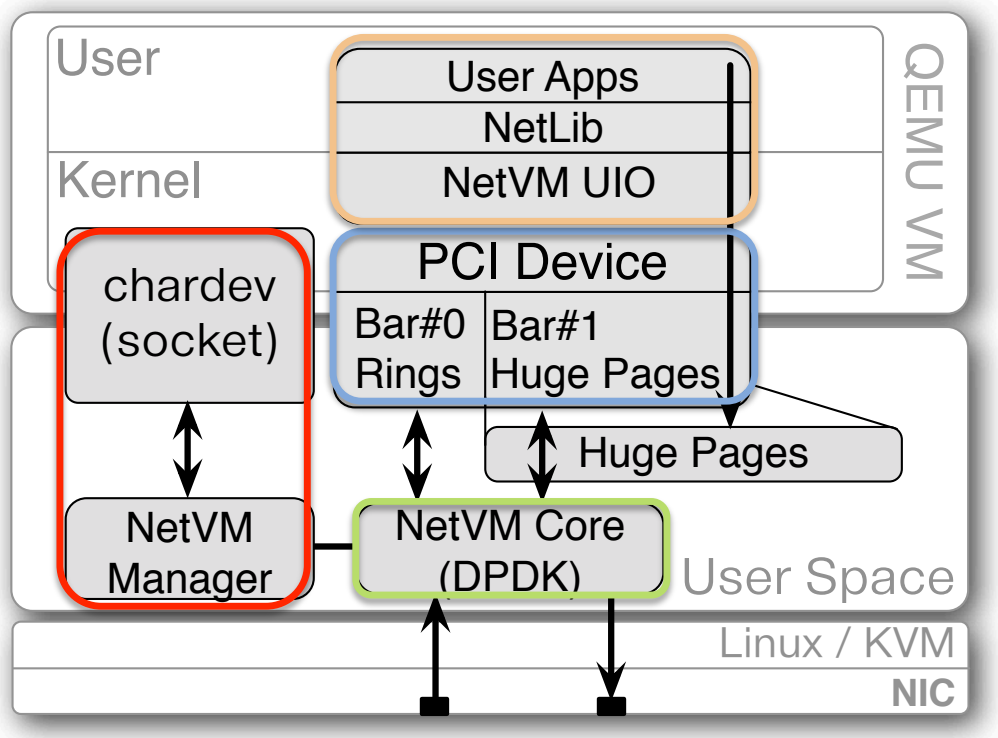
Trusted and Non-trusted Domains

- Virtualization should provide security guarantees among VMs
- Provide a security boundary between trusted VMs and non-trusted VMs
- Non-trusted VMs can not see packets from NetVM
- Groups of trusted VMs via huge page separation



NetVM Implementation

- KVM + DPDK



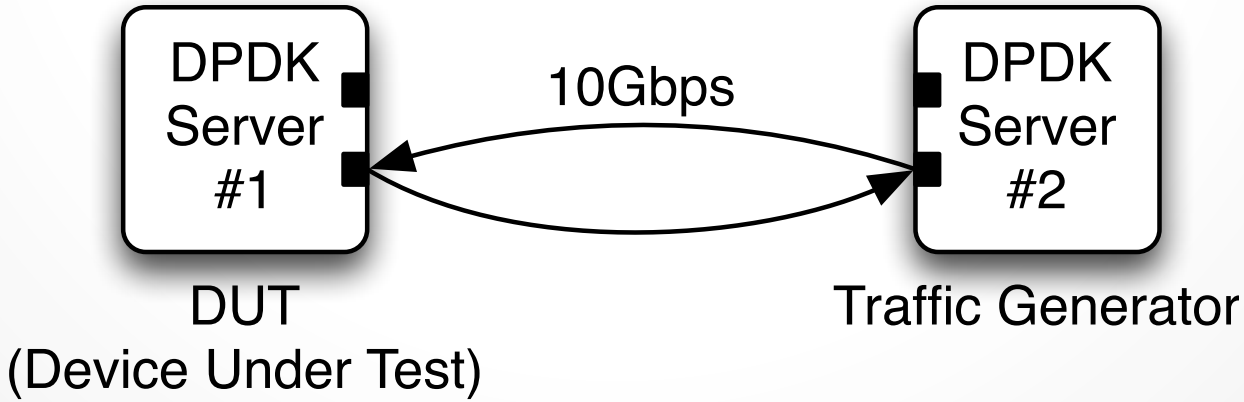
- **NetVM Manager**
  - Hypervisor user space application + QEMU chardev interface
- **NetVM Core**
  - Hypervisor user space application + DPDK
- **Emulated PCI (QEMU)**
  - KVM hardware emulation
- **NetLib**
  - VM user space library + Kernel module

Outline

- Motivation & Contribution
- Background
- System Design
- Evaluation
- Conclusion & Future Work

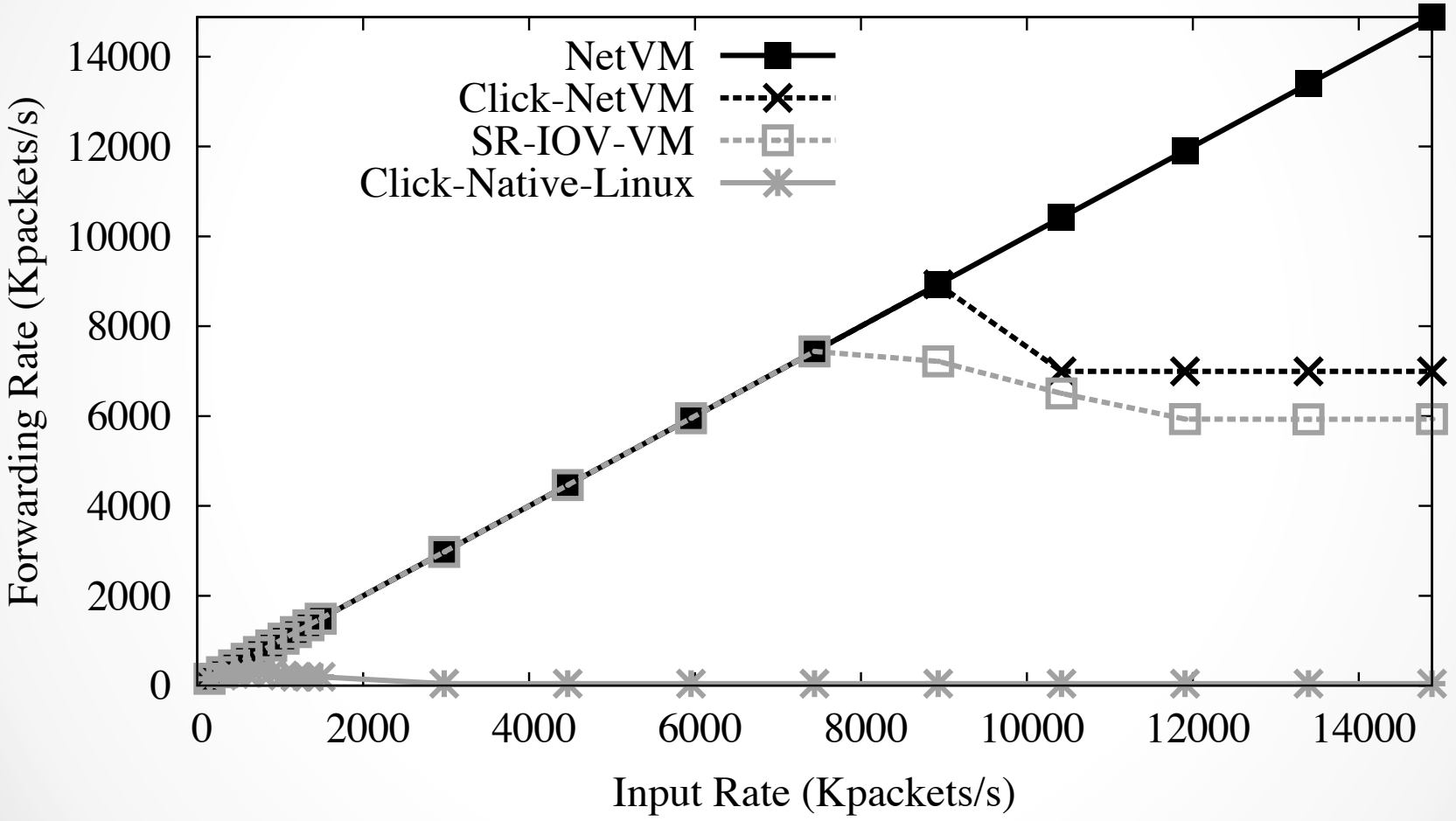
Evaluation Settings

- 2 x Nehalem-architecture (old architecture) machines
- 82599 Intel NIC
- 2 processors * 6 Cores (no hyperthreading)
  - ✓ 2 cores for receiving packets
  - ✓ 4 cores for transmitting/forwarding packets
  - ✓ Rest of cores for VM
- Total 8GB Hugepages (4GB per each processor)
- Only 1 port(out of 2 ports) is used
- Apps: L2/L3 Forwarder, Click Userspace Router, Firewall
- Packet Generator: DPDK-Pktgen (Wind River Systems, Inc.)

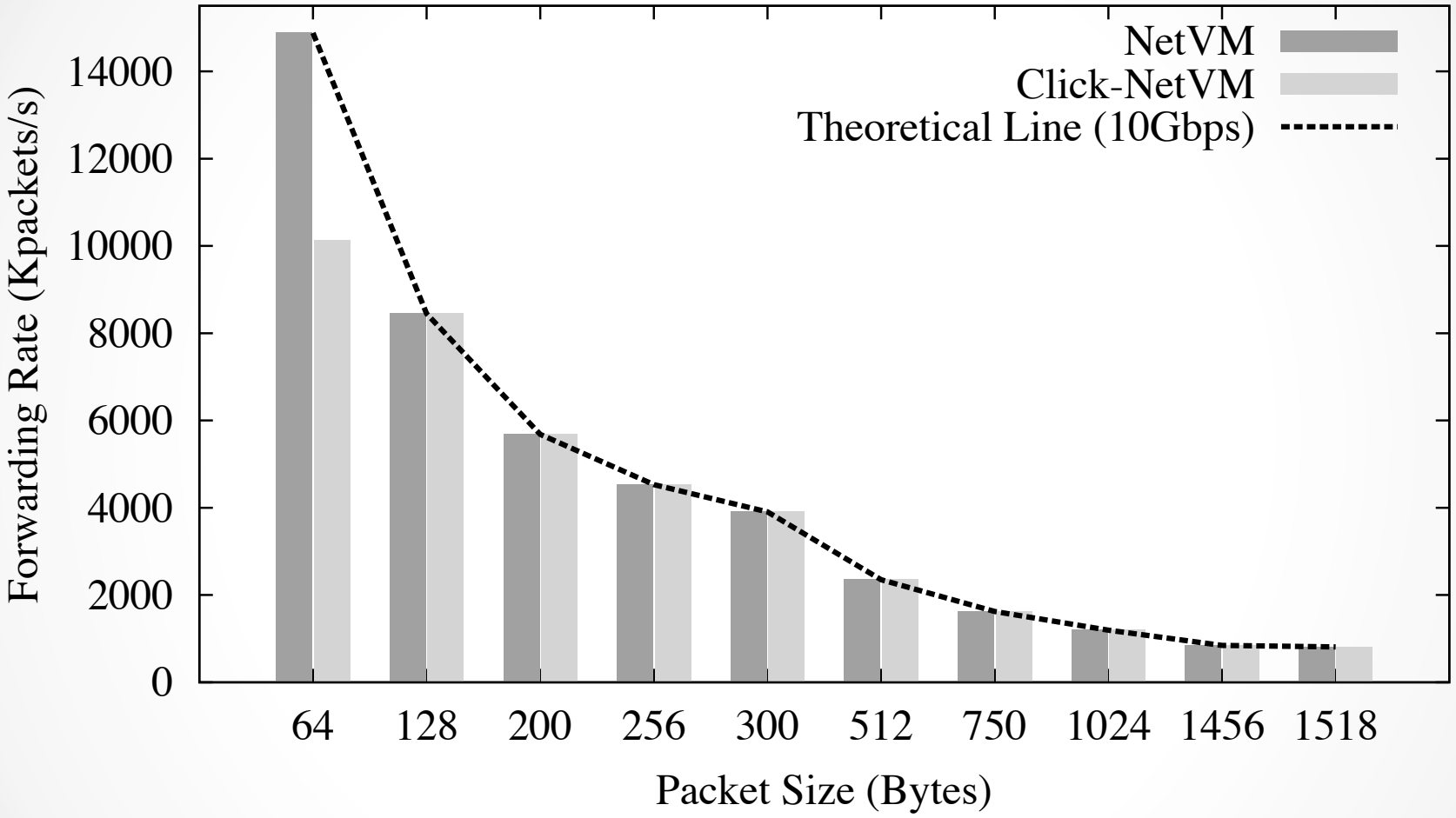


High Speed Packet Delivery

- 64-byte packets, 10Gbps = 14,880,952 packets/s

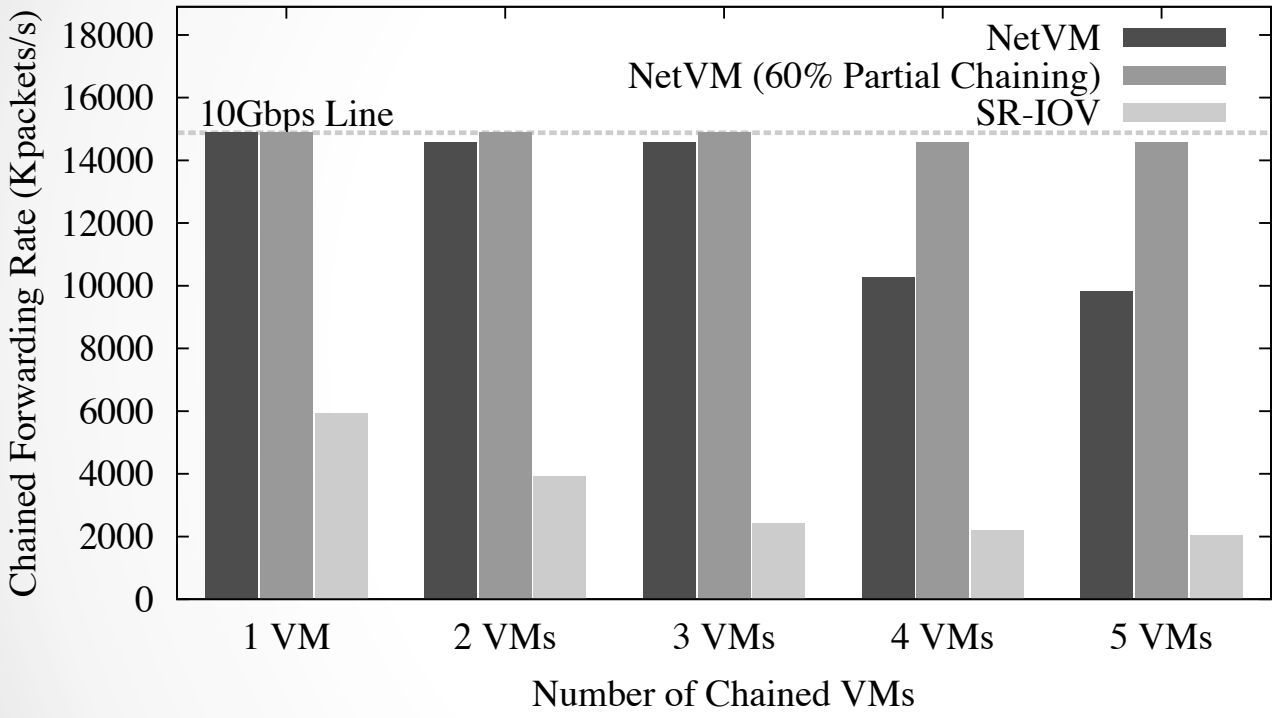


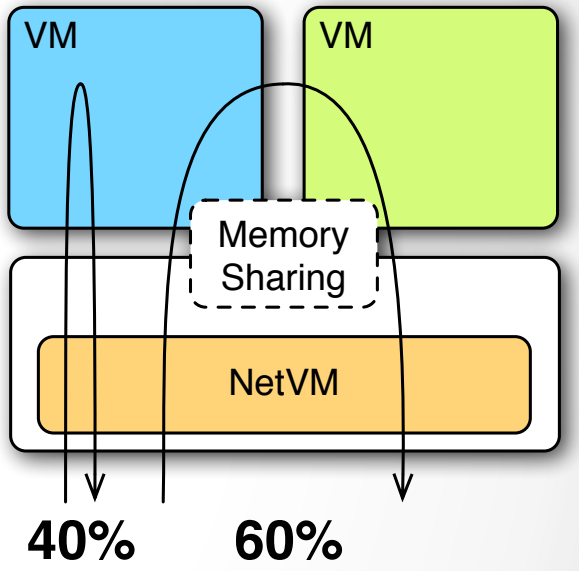
Packet Size vs. Forwarding Rate



Inter-VM Forwarding

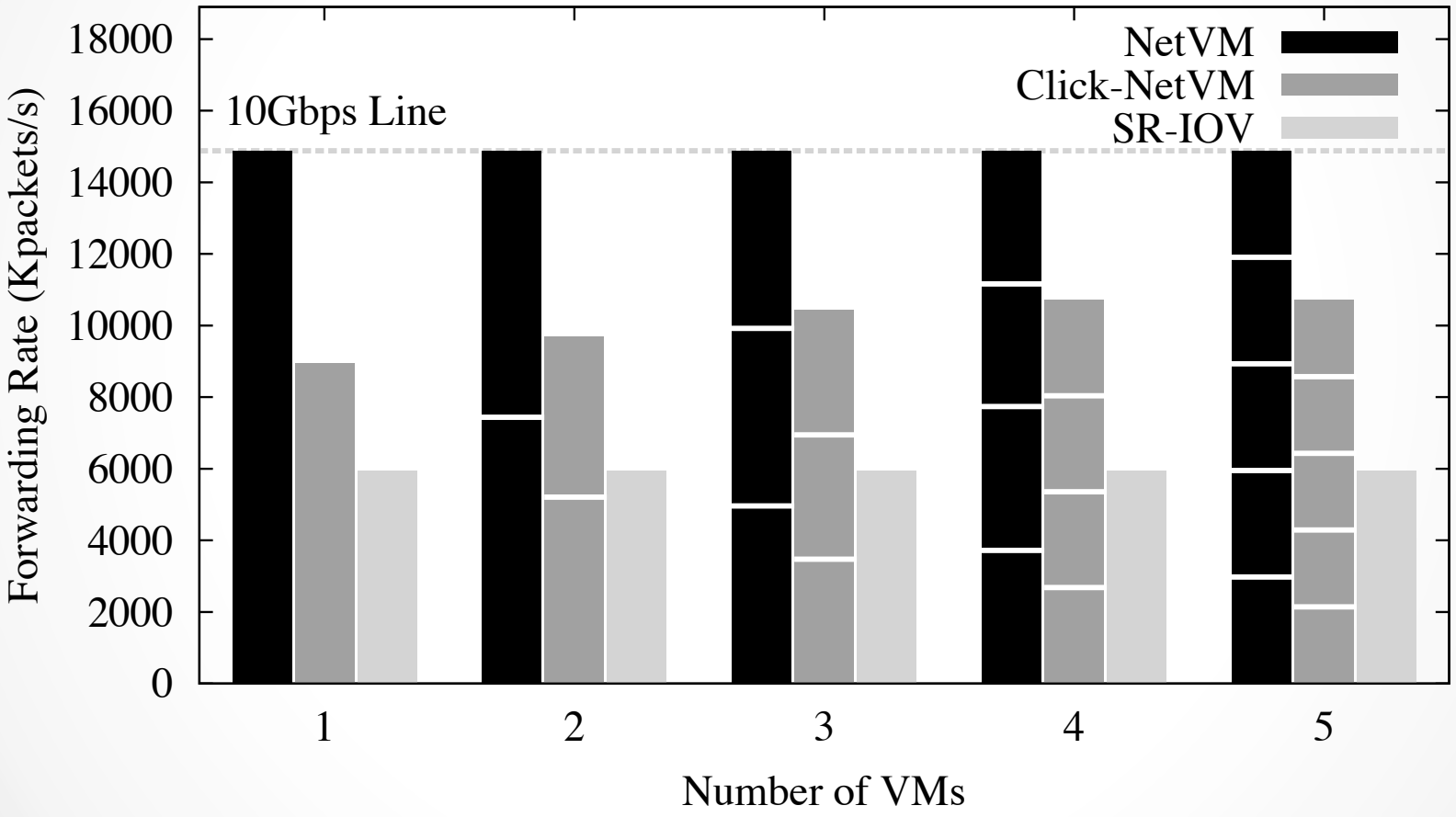
- A mix of L2/L3 forwarding and custom firewall (address filtering)





Switching Flexibility

- State-based load balancing (queue load)



Outline

- Motivation & Contribution
- Background
- System Design
- Evaluation
- Conclusion & Future Work

Conclusion

- NetVM: a high speed packet processing platform based on DPDK
- Advantages:
  - achieve high network speed for hypervisor-VM and VM-VM communication;
  - flexible flow control

- Disadvantages:
  - active polling causes high CPU overhead
  - low speed between trusted and untrusted virtual machines

Future work

- move network stack to user space based on DPDK to provide a compatible running environment for unmodified applications.

- provide more necessary functions such as QoS, bandwidth allocation, NAT, VXLAN, load balancing based on DPDK platform if applied in DC.

- examine DPDK’s application in EPC.