Evaluations of Network Performance Enhancement on Cloud-native Network Function

Yong-Xuan Huang, Jerry Chou

The Fourth International Workshop on
Systems and Network Telemetry and Analytics (SNTA 2021)

國立清華大學
NATIONAL TSING HUA UNIVERSITY

Agenda

□ Motivation

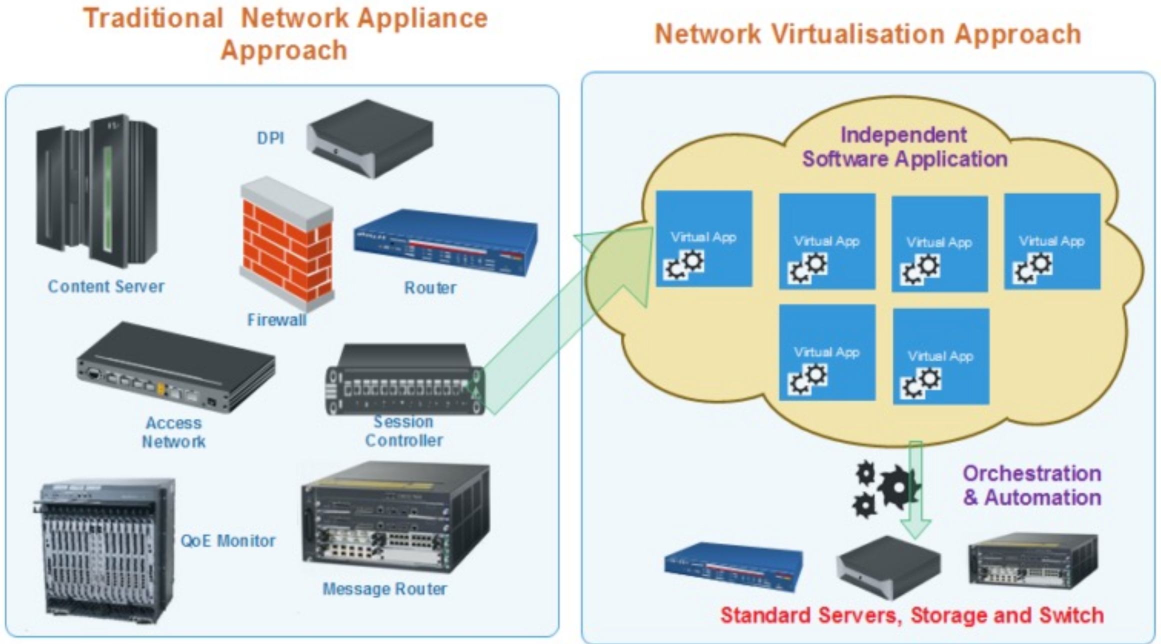
□ Method

□ Experiments

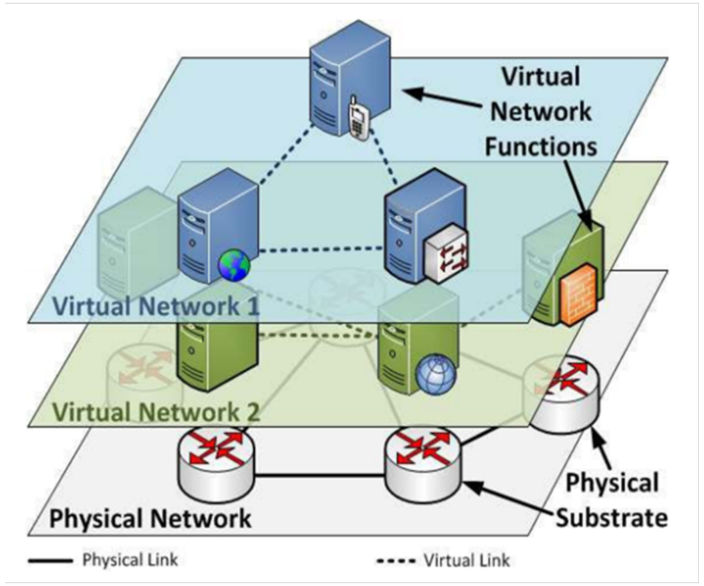
□ Conclusion

Network function virtualization (NFV)

- □ NFV replaces dedicated hardware with software instances that can be deployed, scaled, and migrated dynamically
- □ Reconstruct network services by launching dedicated functions in a single application on general-purpose hardware


Traditional Network Appliance Approach
DPI
Content Server
Firewall
Router
Access Network
Session Controller
QoE Monitor
Message Router
Network Virtualisation Approach
Independent Software Application
Virtual App
Virtual App
Virtual App
Virtual App
Virtual App
Virtual App
Orchestration & Automation
Standard Servers, Storage and Switch

Virtualized Network Function



A virtualized network function (VNF) is an NF designed to run in a virtualized environment

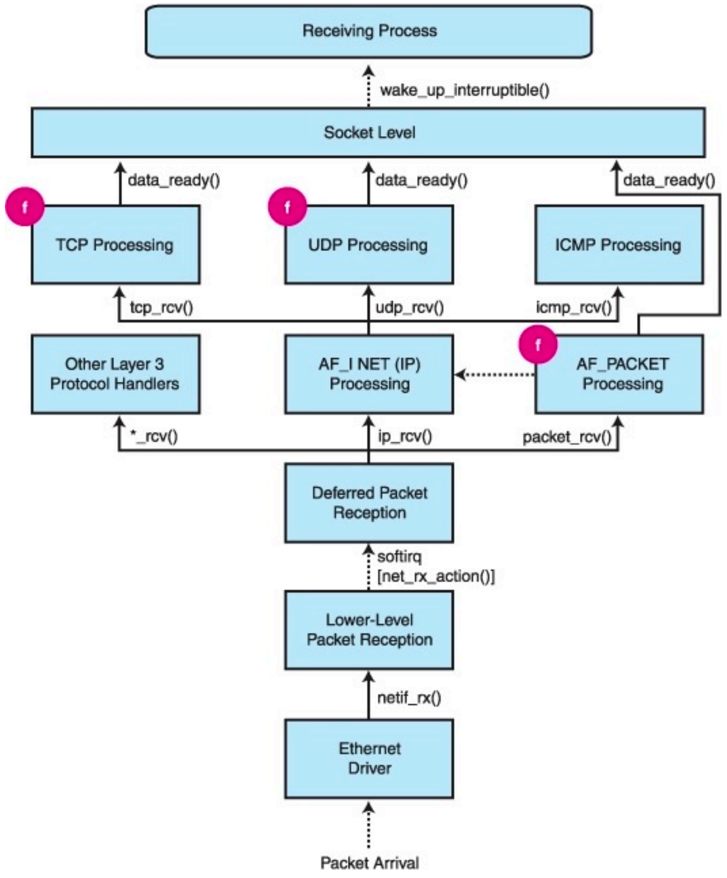
A virtual machine (VM) is a software-only version of a physical server machine.
（ a complete instance of the application code, a guest OS/kernel, and hypervisor that coordinates VM resource management ）

Network functions (NF) are physical devices that process packets supporting a network and/or application service

- □ The core of network functions is performing packets processing
- □ Actually, migrating network functions is implementing the same flow of processing packets onto different platform

New Challenges of NFV

- The network performance on NFV refers to the efficiency of packet processing
- Network performance impacts by migrating special-purpose application-specific integrated circuit (ASIC) to common-off-the-shelf (COTS) hardware
  - Key performance indicator such as throughput and latency, affecting the overall end-to-end application performance

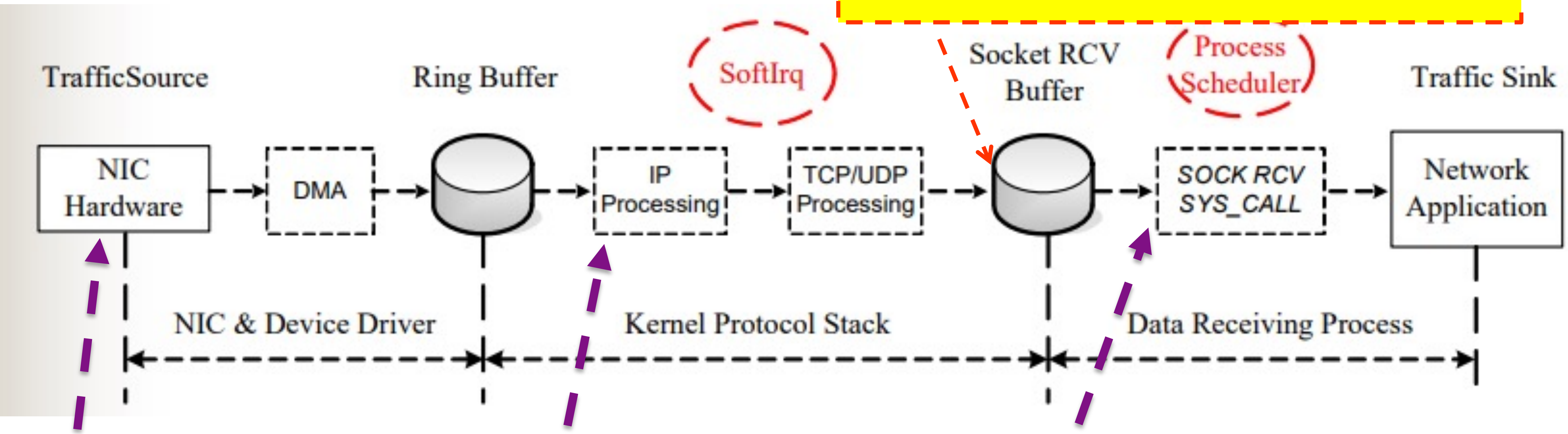


Linux Packet Processing Flow

Root cause 1 - Linux packets processing

- Using ping as example：

Root Cause 1：
Packet Context Switch



1. Receiving packet from hardware

2. Packet is transferred from ring buffer to a socket receive buffer

3. Packet is copied from the socket receive buffer to the application

Root cause 2 – Overlay Network

- Why overlay network ?
  - **Avoid ARP and routing table crash**
  - **Migration services seamless**

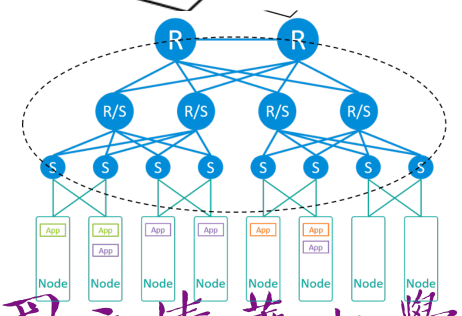
Root Cause 2：
Overlay network overhead

If we have 200 rack, each rack has 10 host, once the working instances grows to 100x, the routing table will crash

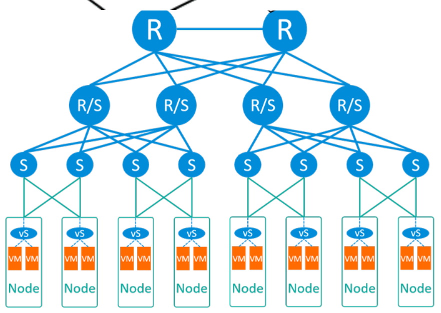
FIB TCAM entries: O(N)
- N: total nodes
- P: ~16K*70% = ~11K effective entries
- N ≪ P (~2k vs. ~11k)

TCAM entries: O(M)
- N: number of nodes
- M: number of total instances
- P: ~64K*70% = ~44K entries
- M ≈ P (~30K vs. ~44K)

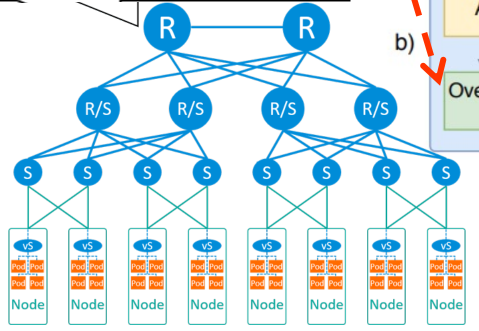
TCAM entries: O(M)
- N: number of nodes
- M: number of total instances
- P: ~64K * 70% = ~44K entries
- M ≫ P (~150k vs. ~44k)



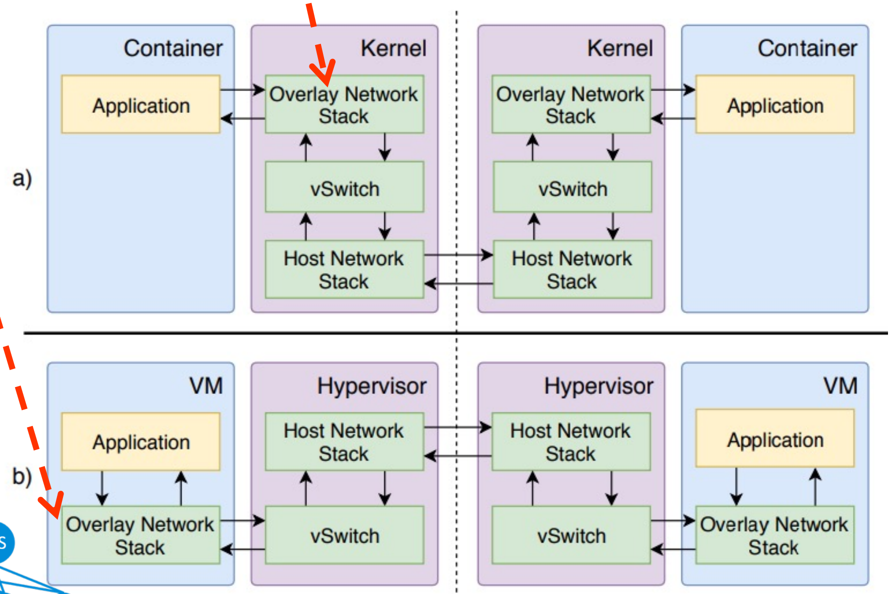
Network Service Scale: 1x



Network Service Scale: 10x



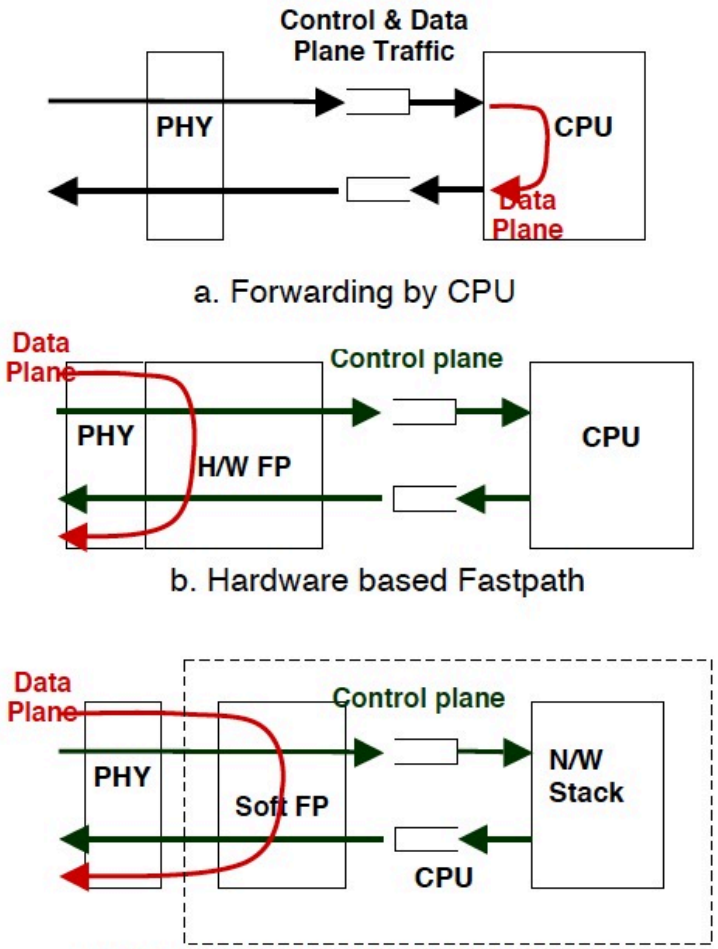
Network Service Scale: 100x



Network Accelerating Solutions

- **Hardware**
  - network processor units (NPUs)
  - graphics processing units (GPUs)
  - field programmable gate arrays (FPGAs)
  - smart NICs (sNICs)
- **Software**
  - CPU pinning
  - Zero-copy
  - Batch processing
  - NUMA-aware
  - Lockless Parallelism
  - eBPF

Most of the software tuning are functioning in user space



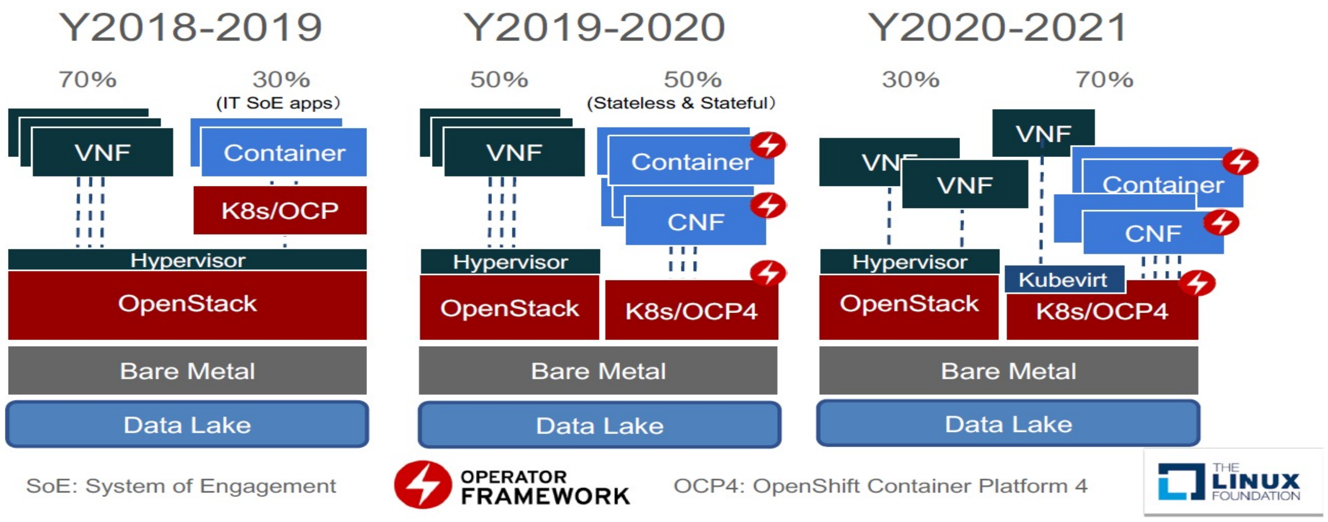
a. Forwarding by CPU

b. Hardware based Fastpath

c. Software based Application Specific Fastpath

Motivation

- □ The platform of NFV is migrating from virtual machine to container due to :
  - □ Slower provision time
  - □ CPU and memory
  - □ Poor network speed



SoE: System of Engagement

OCP4: OpenShift Container Platform 4

- □ How the existing packet processing solutions performs on container-based network function ?

Contribution

- □ To describe existing packet processing framework using in NFV with container
- □ To evaluate the network performance after packet processing framework applied in NFV with container
- □ To discuss and tuning the packet processing performance impacting by network architecture.

Agenda

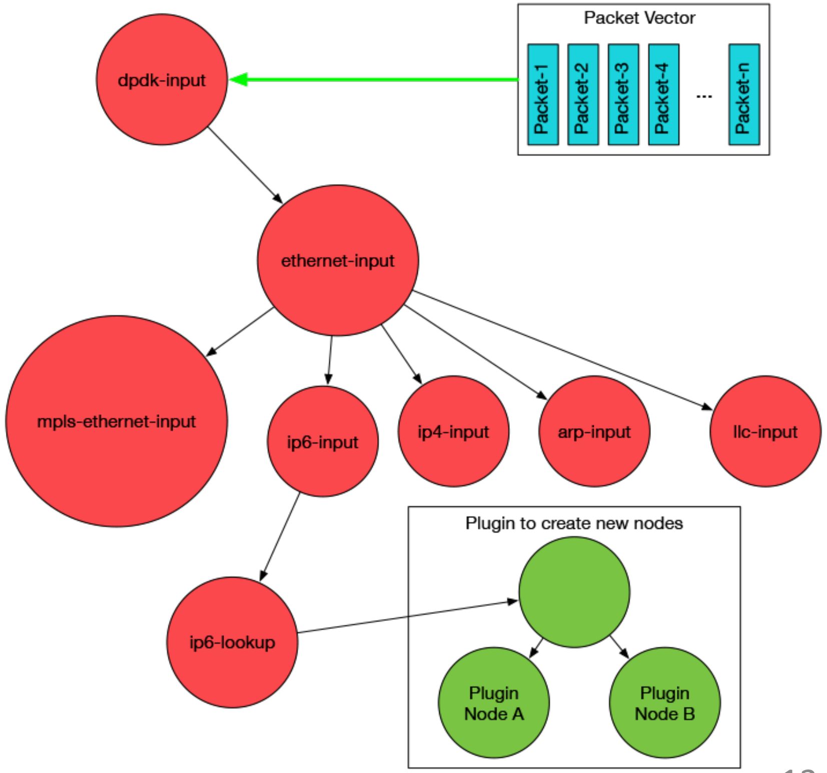
- □ Introduction
- □ Method
- □ Experiments
- □ Conclusion

Kubernetes Networking

- We use Kubernetes to be our evaluation platform, since Kubernetes is a very mature container manager
- Kubernetes does not provide any solution for handling containers networking
  - It offloads networking to third-party certified plugins called CNI plugins
- Most CNI plugins use overlay network as default
  - the acceleration is still needed

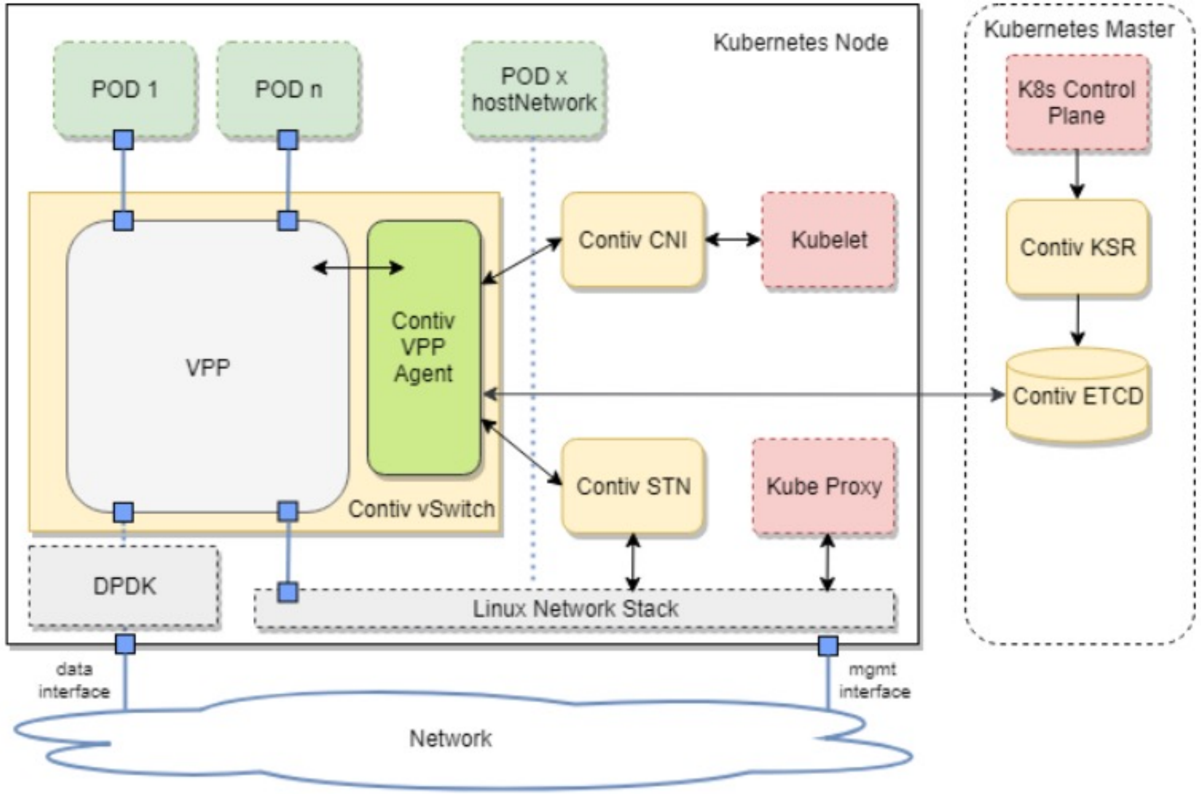
User Space CNF packet processing

- Why user space?
  - It is easier to design a new algorithm if the packets are processed in the user space
- User space accelerating Solutions
  - Vector Packet Processing (FD.io VPP)
    - Lockless Parallelism
  - Data Plane Development Kit (DPDK)
- Kubernetes Solutions
  - Contiv



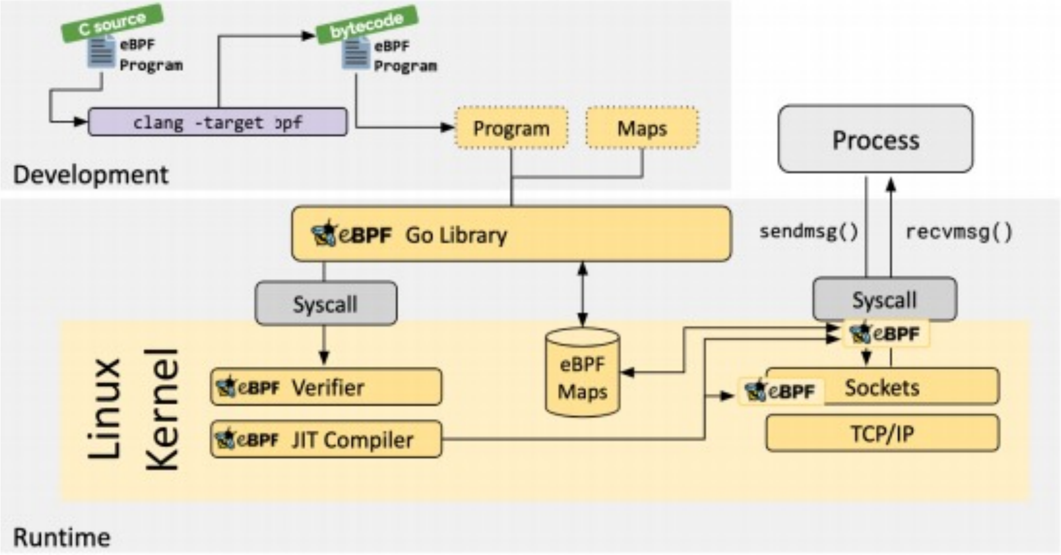
Contiv Network Architecture

□ Contiv/VPP is a Kubernetes network plugin that uses FD.io VPP with DPDK as the dataplane for packet forwarding between PODs in a Kubernetes



Kernel Space CNF packet processing

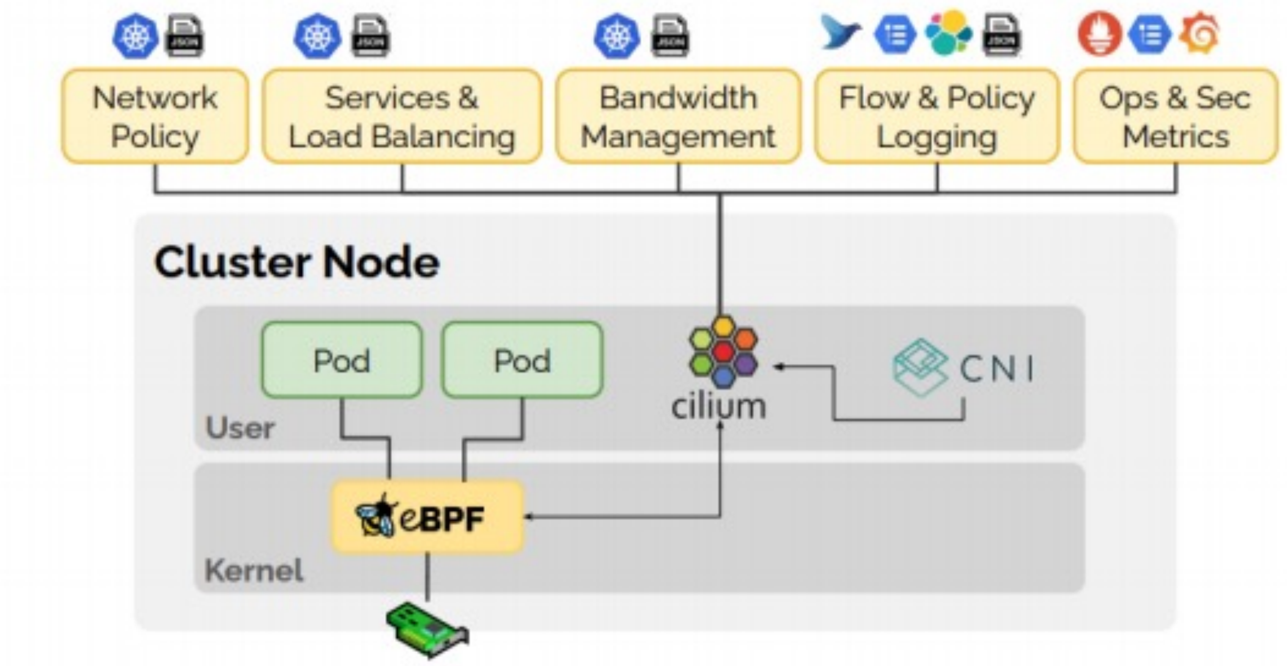
- New Chapter of Kernel：A new just-in-time (JIT) feature
  - Developers can extend functions efficiently under the governments to reduce failures
- Kernel space accelerating Solutions
  - eBPF/XDP
- Kubernetes Solutions
  - Cilium
  - Silm
    - https://github.com/danyangz/Slim



Cilium

- □ The foundation of Cilium is eBPF
- □ Because eBPF runs inside the Linux kernel, Cilium modules can apply and updated without any changes to the application code or container configuration
- □ It is possible to use routing since

the routing module has updated



Solutions Comparison

CNI	Space	Background	Network Model	Routing	Technical
Contiv	User	Using VPP and DPDK to increase packet processing on user space	Layer2, Layer 3, ACI	iptables	Vector Packet Processing, Data Plane Development Kit
Cilium	Kernel	Using eBPF to provide new Kernel module solution like routing and security	Layer 2 by (default), Layer 3 (optional)	BGP, Kubeproy	eBPF

Agenda

□ Introduction

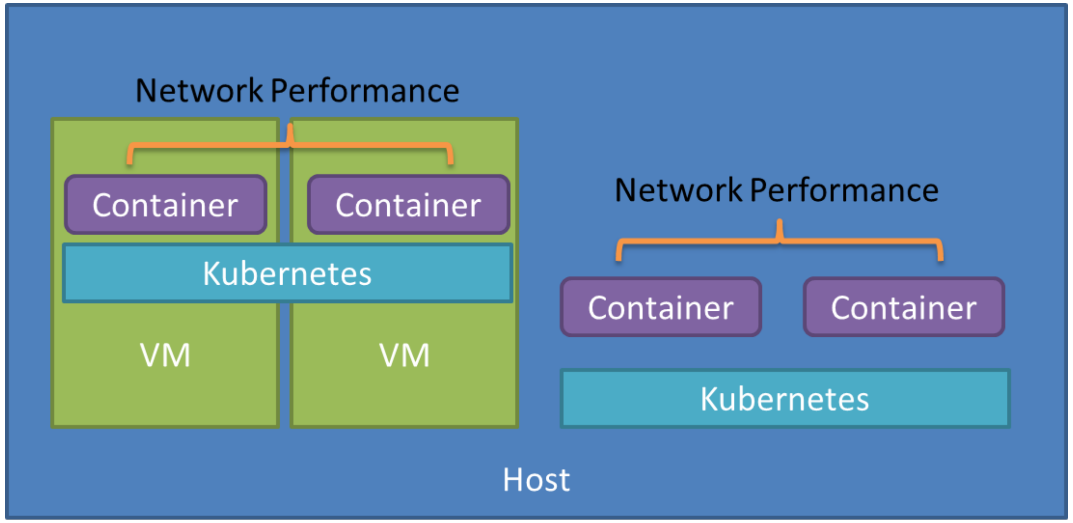
□ Method

□ Experiments

□ Conclusion

Experiments Setup

- The goal of our experiments：**Packet Processing**
  - We reduce the side effects from network equipment
- We first estimate the baseline of host and VM, then compare the solutions along with the network models individually

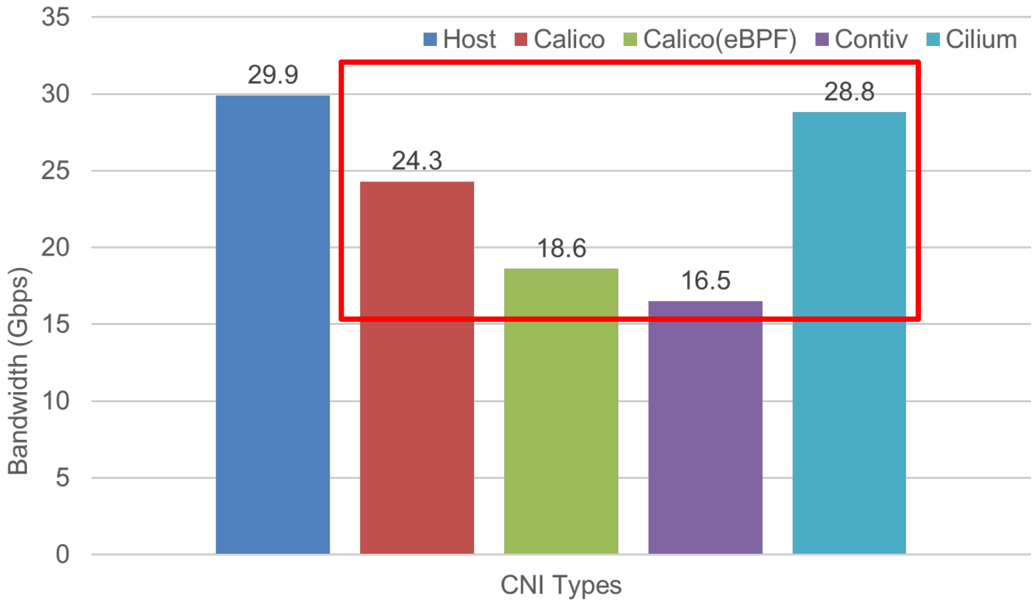


Testbed Specs

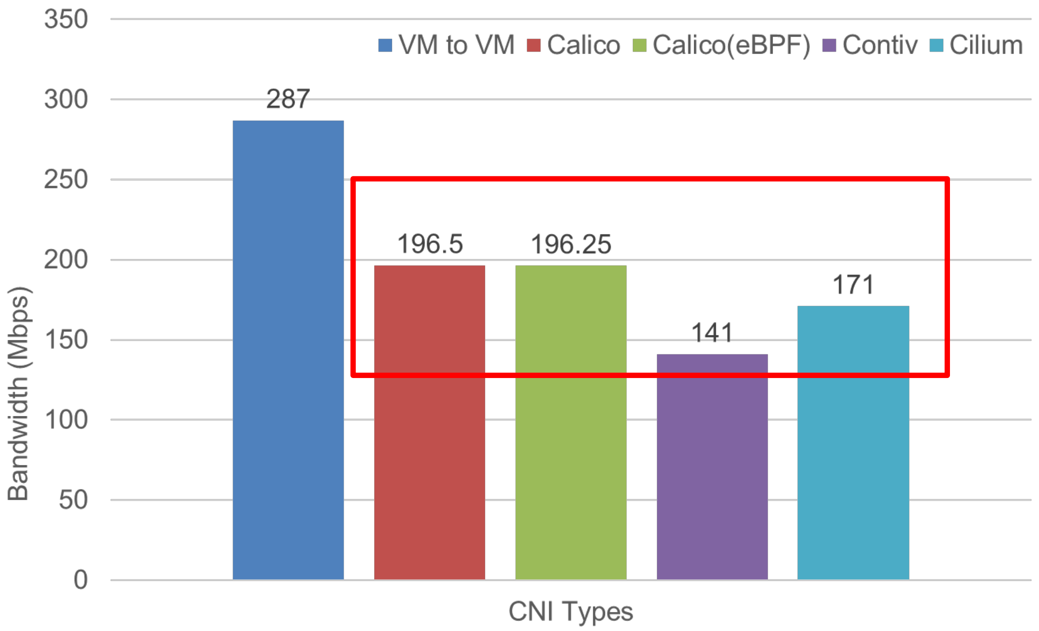
- Hardware
  - CPU：
    - Two Intel Xeon Processor X5670 2.93-GHz processors
    - Each processor had 12 physical cores, and hyper-threading was enabled.
  - Ram：96GB
- Software
  - Linux kernel：5.4.0-66- generic operating system Ubuntu 20.04.1
  - Docker：19.03
  - Kubernetes：1.20
- Virtualization platform
  - VM： KVM
  - Kubernetes CNI：Calico

Scenario – Single VM/Host

- Kernel space acceleration can help to reach nearline in host, user space acceleration is the worst
- The performance on vm is quite different as we thought, all performance decreasing on VM, even Cilium is worse than container baseline



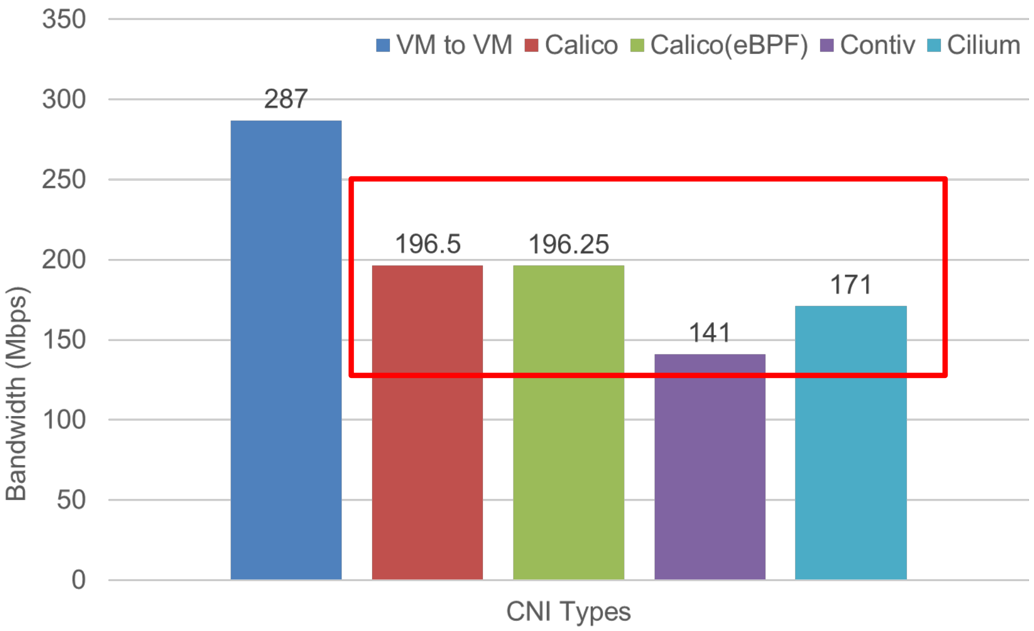
CNF on bare mental host



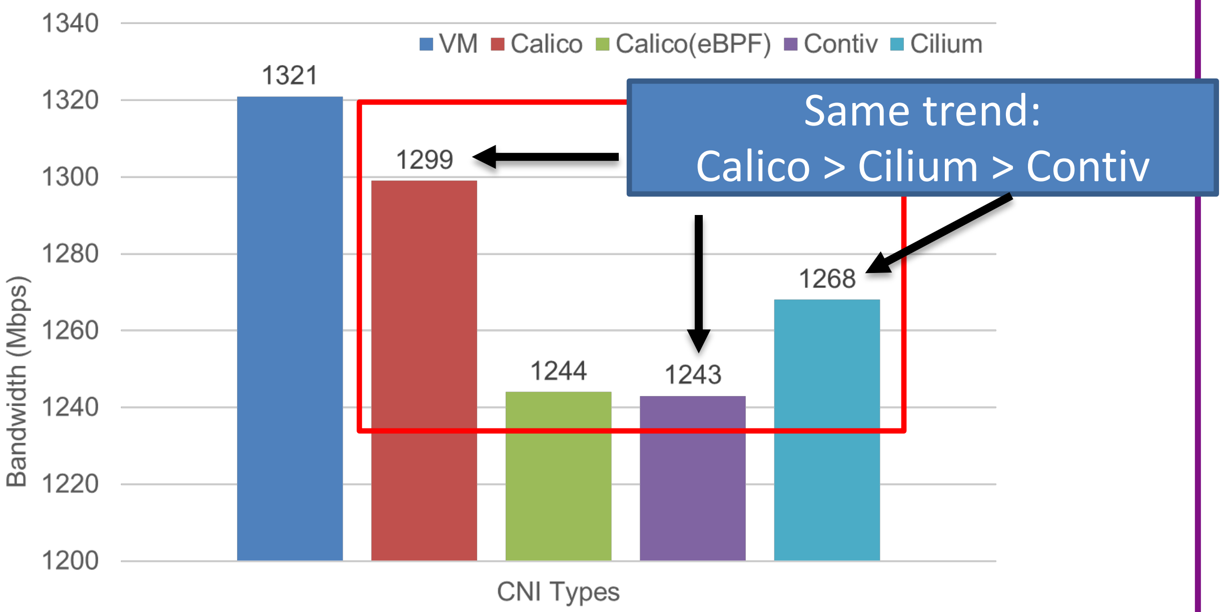
Cross VM On Single Host

Scenario – Single VM with SR-IOV

- According to our related work, VM can get acceleration by using hardware technique
- Both VM and Container **baseline has increased, but the trend is still the same**
- Since the container baseline has increased, Cilium still can't reach container baseline but better than Calico eBPF mode like host



Cross VM On Single Host



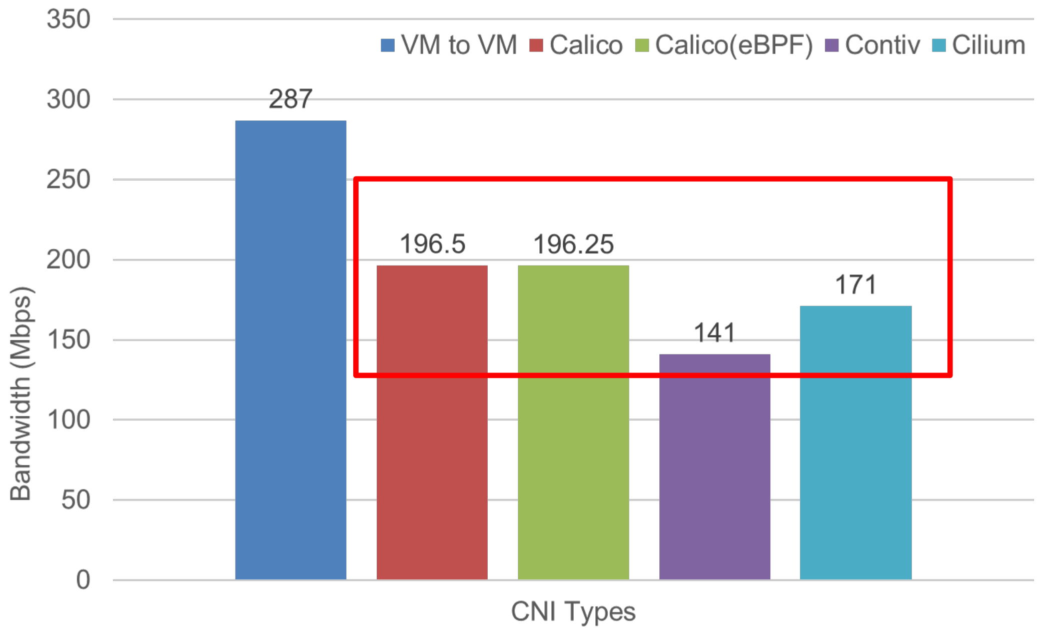
Same trend:
Calico > Cilium > Contiv

Cross VM Isolated On Single Host with SR-IOV

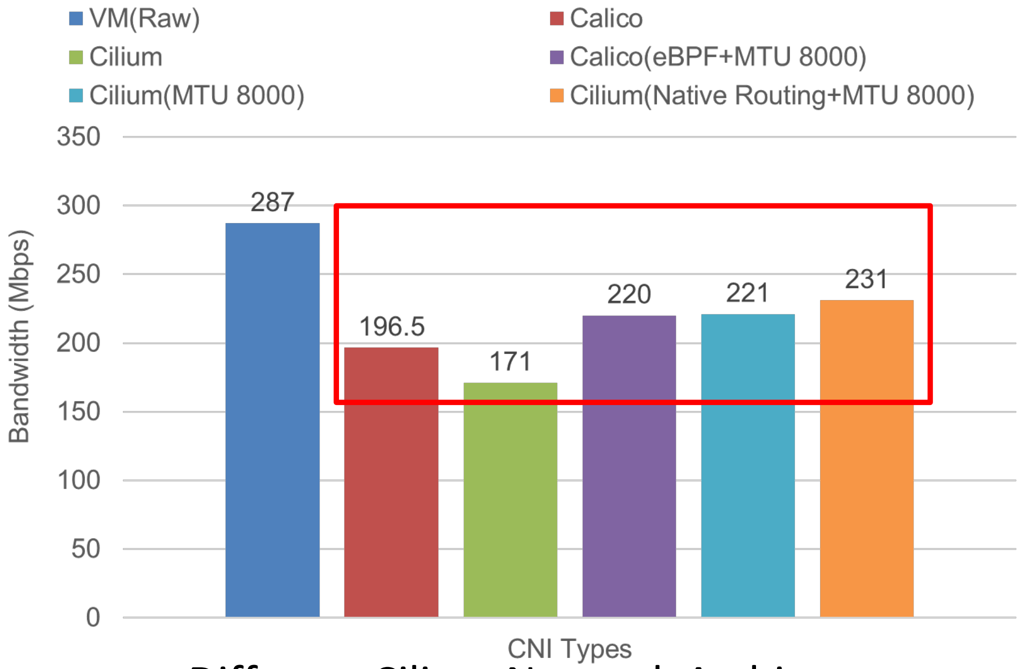
國立清華大學
NATIONAL TSING HUA UNIVERSITY

Scenario – Kernel Space Tuning

- The MTU size impact significantly on network, the performance can sometime increased after adjusting the MTU size
- Due to eBPF programmable in Kernel, packet processing benefits from using native routing instead of overlay network



Cross VM On Single Host



Different Cilium Network Architecture

Agenda

- □ Introduction
- □ Method
- □ Experiments
- □ Conclusion

Conclusion

- □ Kubernetes CNI with the user space and the kernel space severally impact the CNF packet processing performance
- □ Kernel Space Acceleration on the host or on the VM after tuning can get best performance result
- □ User Space acceleration looks not helping in container packet processing even with hardware support
- □ We could perform more comparisons of network features on container like security or IP management for further research