Performance Considerations of Network Functions Virtualization using Containers

Jason Anderson, et al. (Clemson University)

2016 International Conference on Computing, Networking and Communications, Internet Services and Applications

Speaker: Tao Gao

2018-04-20

Group Meeting Presentation (Paper Review)

VM-based NFV Implementation

- Hardware virtualization incurs significant performance and efficiency costs.
  A traditional virtual machine is an abstraction of physical hardware giving each VM ***a full server hardware stack*** *including virtualized network adapters, storage and CPU.*

- Heterogeneous packaging and virtualization platforms may result in fragmented distribution and complex orchestration.

- VM images can be large, resulting in significant time to deploy and migrate between hosts.

- Network I/O, which is of critical importance to NFV, can also suffer in many configurations.

Container-based NFV Implementation

- Since applications in containers run on the host OS without hardware indirection, they can run more efficiently than their VM-based counterparts and allow higher application density on a host.
- Docker's novel packaging can remove some of the variability in hosting requirements that a VNF may express.
- Containers do not require packaging the operating system in their image, and as such generally consume much less disk space than comparable VMs, decreasing time to deploy and migrate.

One important challenge is not sufficiently addressed by containers is network I/O.

Virtualization Technology

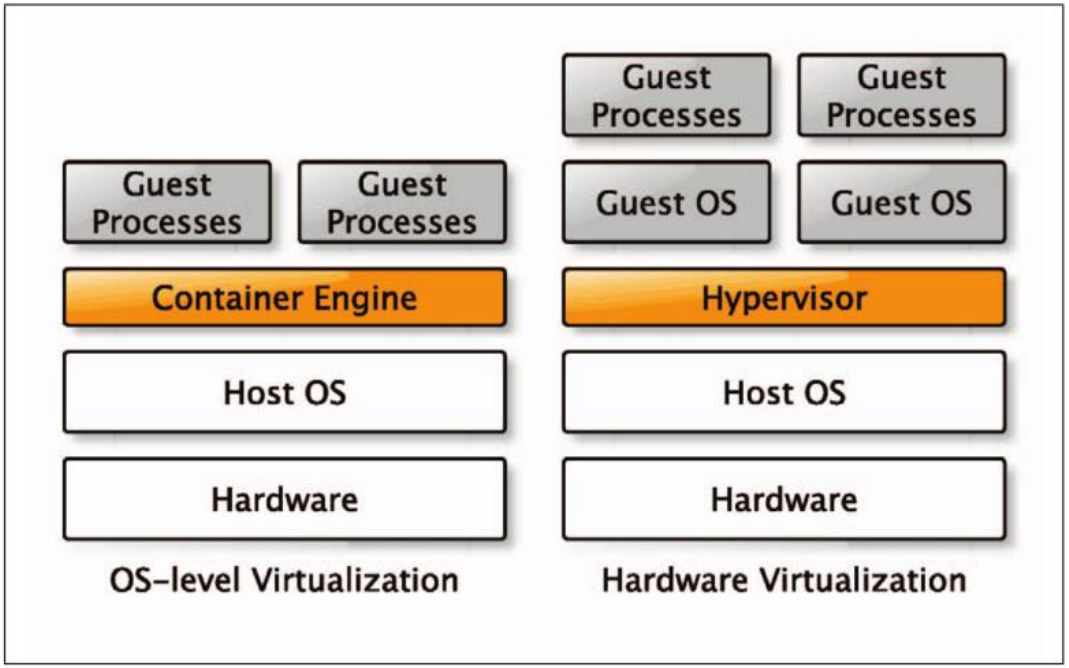

Fig. 1. Software layers of container-based (OS-level) and hypervisor-based (hardware-level) virtualization

A perfectly tuned container system can have as many as four to six times the number of server application instances as is possible using Xen or KVM on the same hardware [1].

[1] S. Yangui, M. Mohamed, S. Tata, and S. Moalla, "Scalable service containers," in Cloud Computing Technology and Science (CloudCom), IEEE Third International Conference on, pp. 348–356, IEEE, 2011.

Networking Technologies

1) **Network Device Virtualization**
   - *Macvlan*: Allows the abstraction of a single network interface into multiple network interfaces with different hardware addresses assigned to them.
   - *Single Root I/O Virtualization (SR-IOV)*: Allows a physical PCIe device to present itself to the OS as multiple separate PCIe devices called virtual functions (VFs), each with dedicated queues for transmitting and receiving packets.

2) **Software Switches:**
   - *Linux bridge*: A layer 2 software switch included with the Linux kernel, and is commonly used to connect two Ethernets together.
   - *Open vSwitch (OVS)*: Commonly used in a virtualized environment to interconnect VMs within a host.

Performance Comparison

- Latency

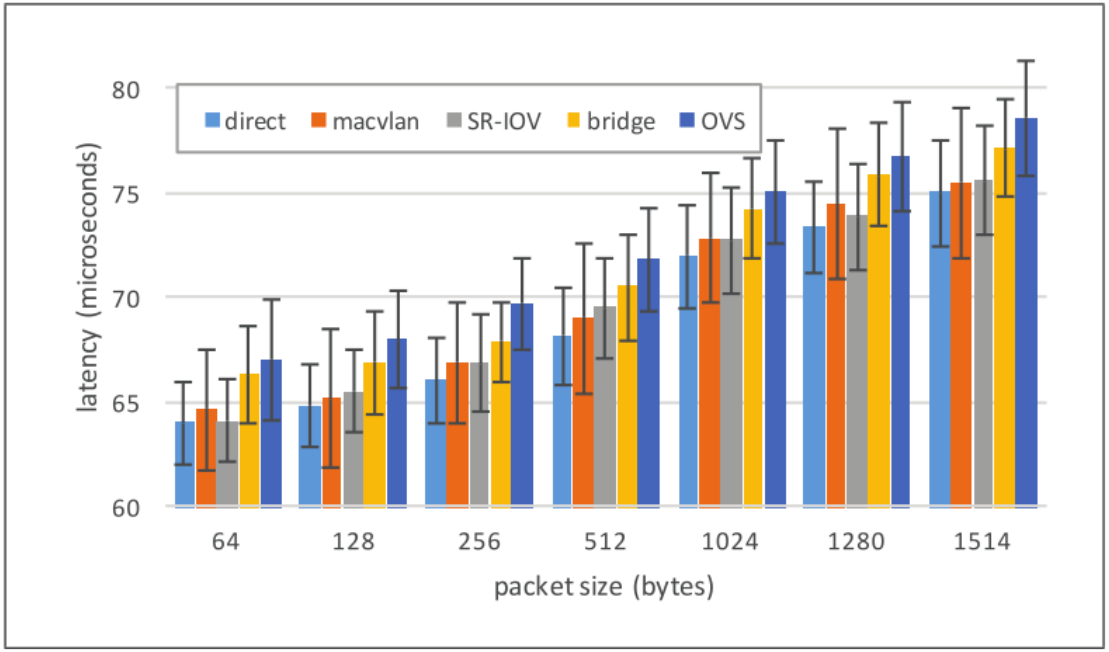

Fig. 2. Latency and standard deviation of various-sized Ethernet packets from an external host to processes in Docker containers, using different networking technologies.

Device virtualization mechanisms such as macvlan and SR-IOV incurred less processing delay than software switches.

Performance Comparison

- Latency

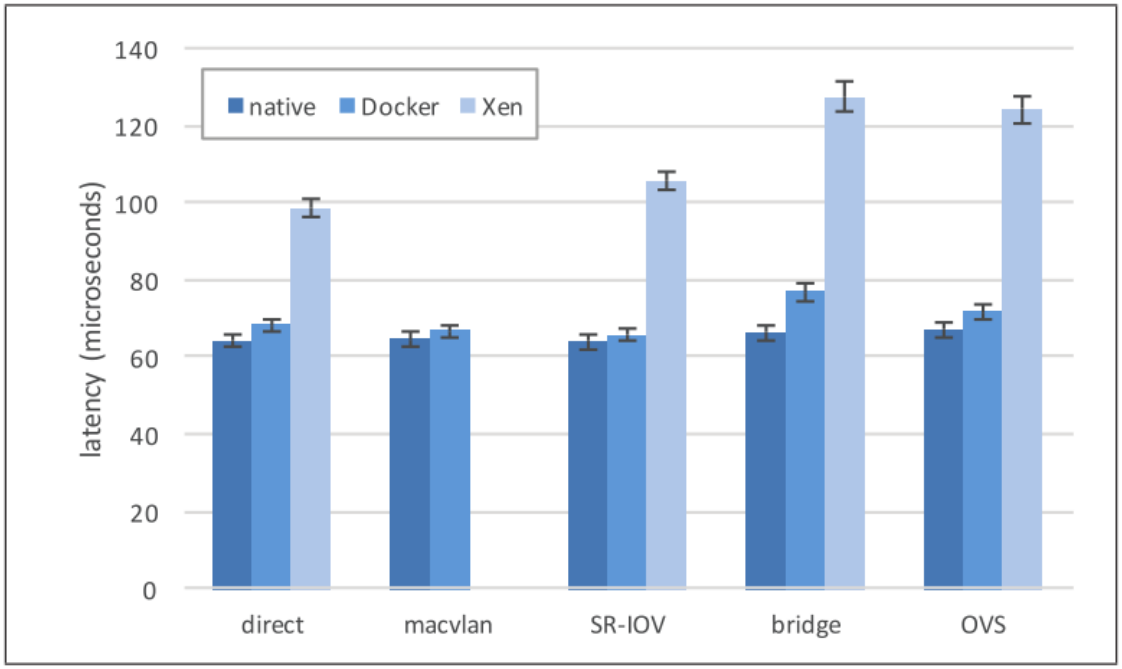

Fig. 3. Latency and standard deviation of 64-byte Ethernet packets from an external host to processes running natively, in Docker containers, and in Xen virtual machines.

With each networking technology, containerized applications carry an additional latency cost compared to native applications, but less of a penalty than the equivalent Xen VMs .

Performance Comparison

- Efficiency

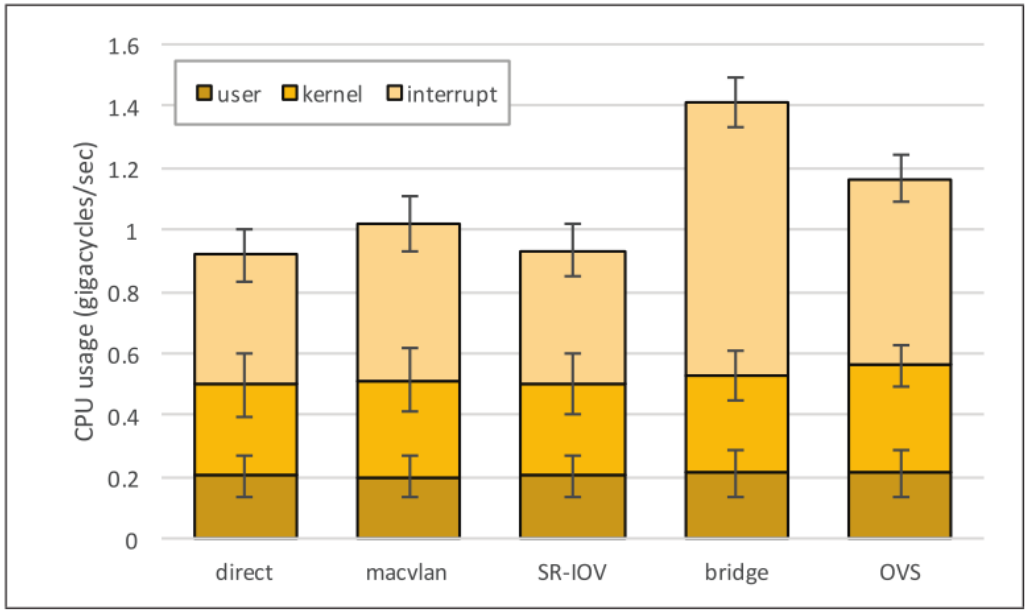

Fig. 4. Computational efficiency and standard deviation of networking mechanisms forwarding 64-byte Ethernet packets from an external host to a Docker container, classified into user processes, kernel processes, and interrupt servicing.

SR-IOV had nearly the same computational efficiency as direct assignment of the interface, while macvlan, the Linux bridge, and OVS increased overhead per packet substantially.

Performance Comparison

- VNF Chain Performance

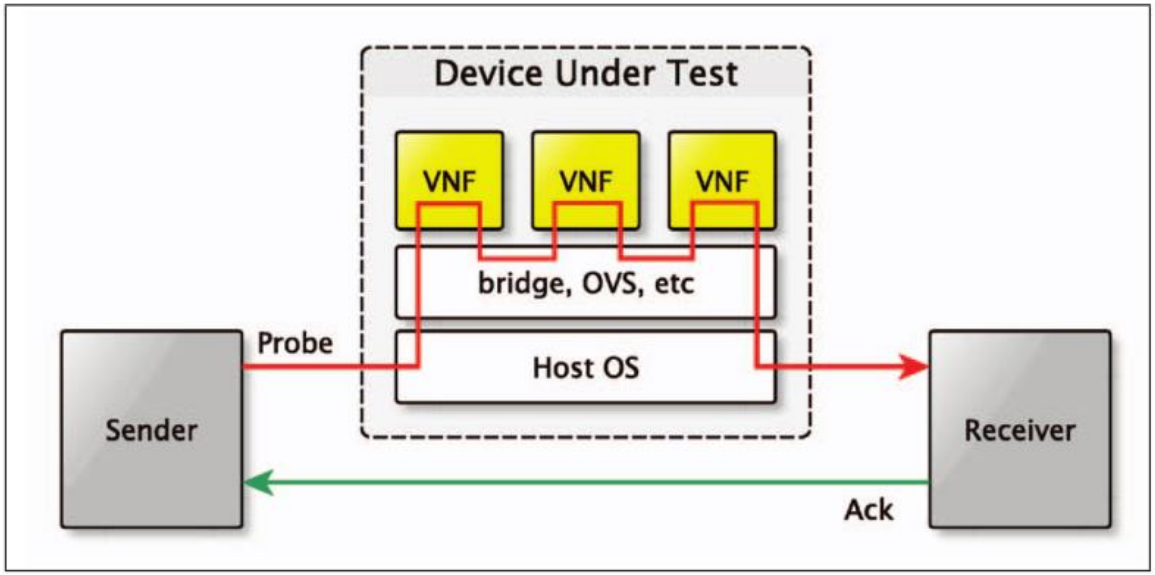

Fig. 5. Experiment architecture for simulated chains of VNFs. Probe packets emitted by the sender are routed through each VNF in the chain. Responses by the receiver bypass the VNF host.

Performance Comparison

- VNF Chain Performance

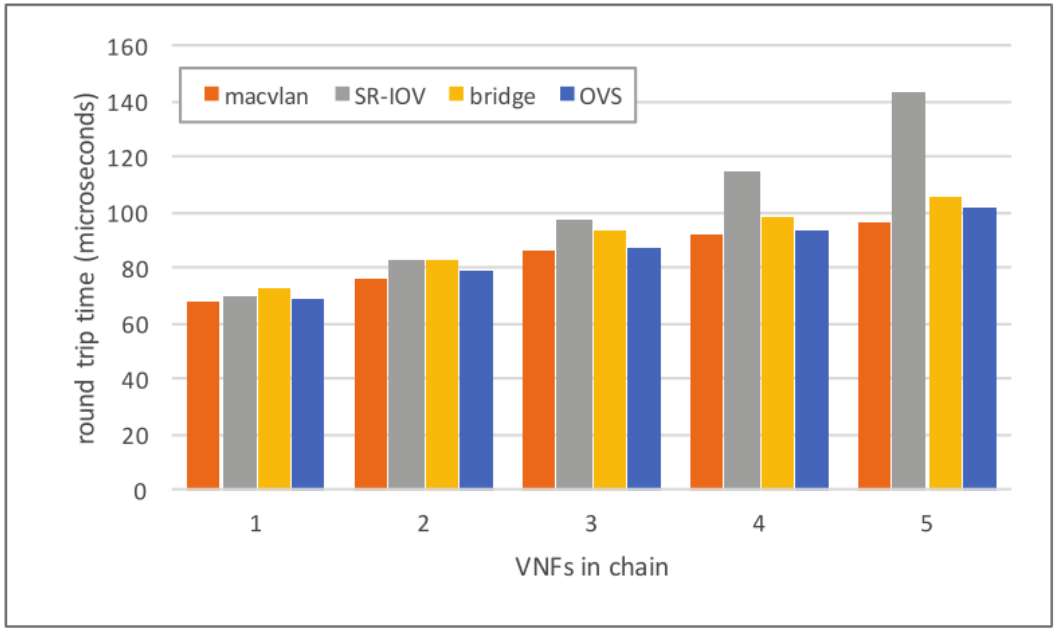

Fig. 6. Relative delay cost of packet-forwarding chains of VNFs on one host, classified by networking technology.

SR-IOV incurs a large cost to transport packets across the PCI bus for classification by the NIC. Hybrid schemes for packet forwarding may be required to take advantage of SR-IOV in certain service chain deployments

An Empirical Case for Container-driven Fine-grained VNF Resource Flexing

Amit Sheoran, Sonia Fahmy, et al. (Purdue University)

2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)

Introduction

- Compare the performance and resource usage of three Virtualized Network Functions (VNFs) with bare metal (BM), container, and Virtual Machine (VM) instantiations, at a variety of load levels and resource allocation configurations.
- The three VNFs we benchmark are:

  (1) The Mobility Management Entity (MME) of the Evolved packet core (EPC) architecture for cellular networks

  (2) the Suricata multi-threaded Intrusion Detection System (IDS)

  (3) the Snort single-threaded IDS

Introduction

(1) **Plane of operation**:

- MME: a control plane element; CPU-intensive, do not stress the data plane of the network.
- IDS: a data plane entity, network-intensive.

(2) **Software architecture**:

- Snort: single-threaded.
- MME and Suricata: multi-threaded systems.

(3) **Network positioning**:

- MME: a stateful transactional element that communicates with other EPC elements to handle user requests.
- IDSes: typically deployed as stand-alone middle-boxes.

EPC Experiments

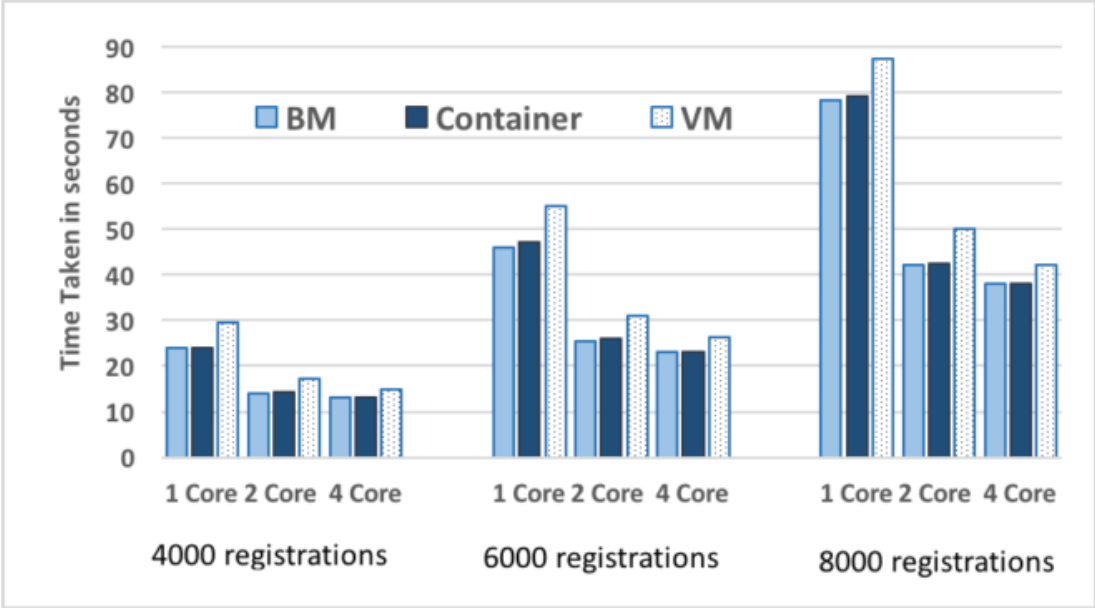

Fig. 7: Time taken to handle registration requests.

- ✓ VMs incur significantly higher overhead than the bare metal setup and Docker containers.
- ✓ When two cores are available to the system, performance of the MME significantly increases because the processing thread uses one of the CPU cores.

EPC Experiments

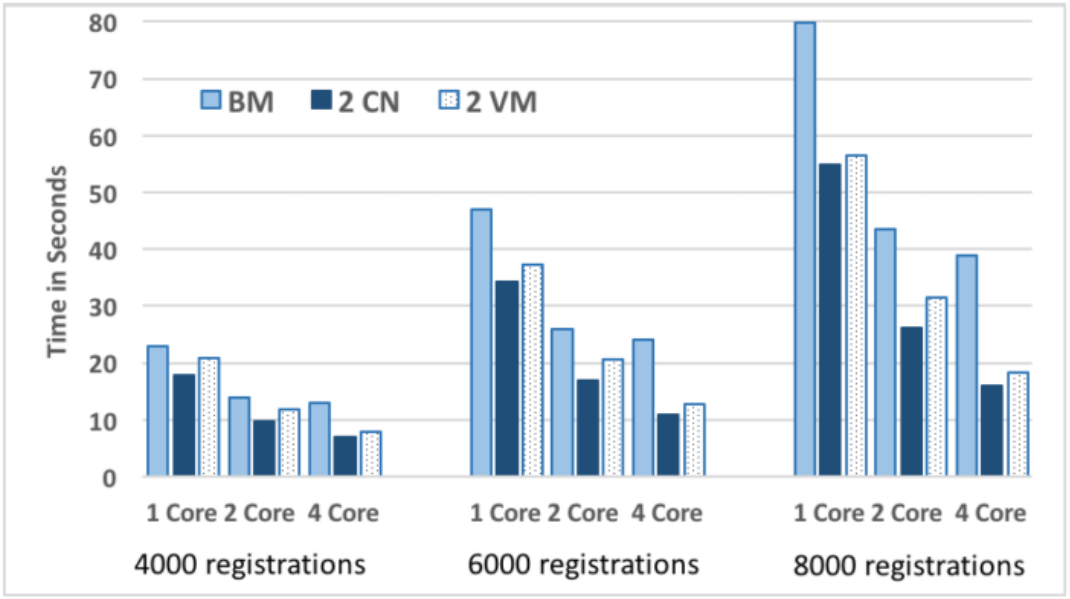

Fig. 8: Time taken to handle registration requests by bare metal and two instances of VMs or containers.

- ✓ The time taken to handle the registration requests is considerably reduced when traffic is split across two instances with similar processing resources.
- ✓ As the number of CPU cores increase from 2 to 4 the time taken by the MME to handle registration requests decreases.

Suricata Experiments

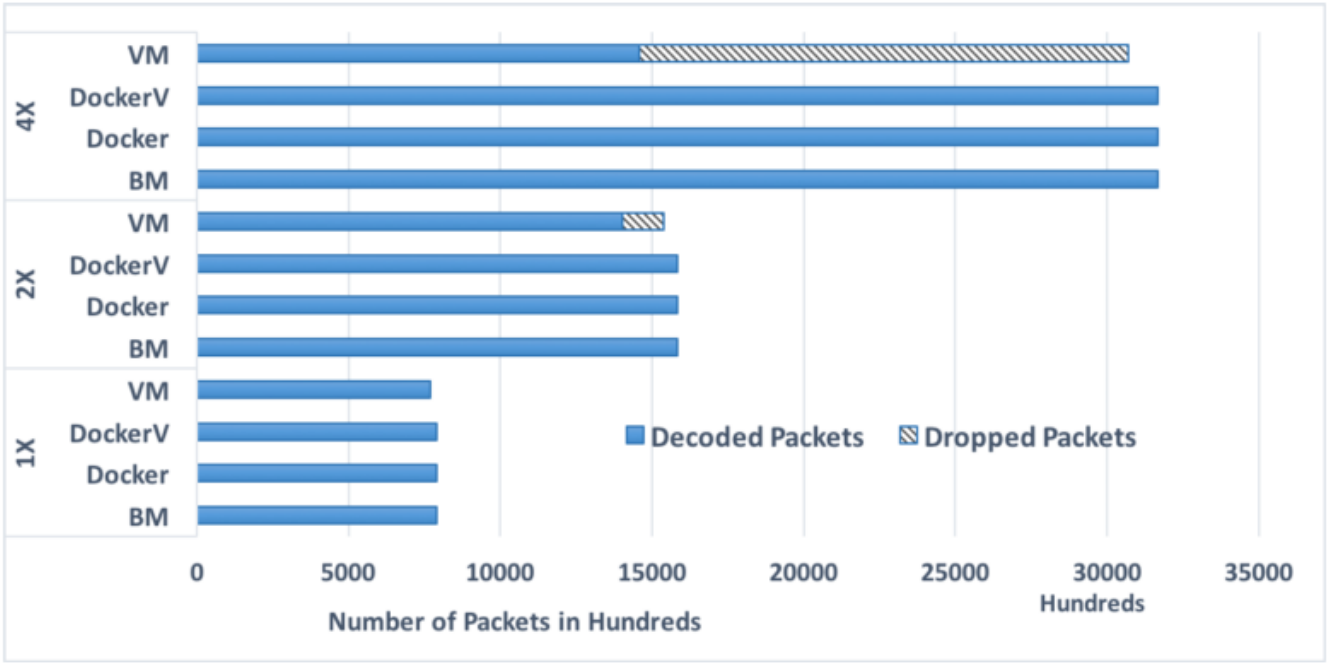

Fig. 9: Cumulative packets decoded and dropped by Suricata in four setups at all loads.

- ✓ Only the VM setup exhibits packet drop starting at 2X load, and almost all increased load above 2X is dropped.
- ✓ CPU is nearly saturated by Suricata at 2X load level and therefore almost all additional traffic generated at 4X load is dropped.

Suricata Experiments

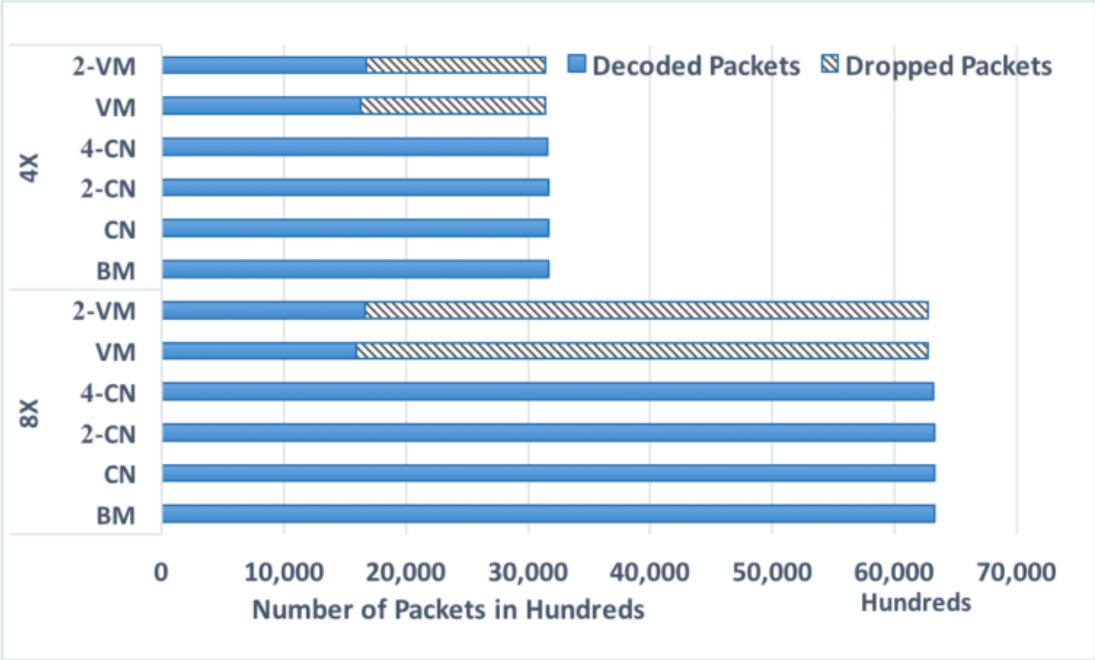

Fig. 10 Performance of Suricata with multiple VM and container instances.

- ✓ No observable performance difference between the single and multiple instance deployments.

Snort Experiments

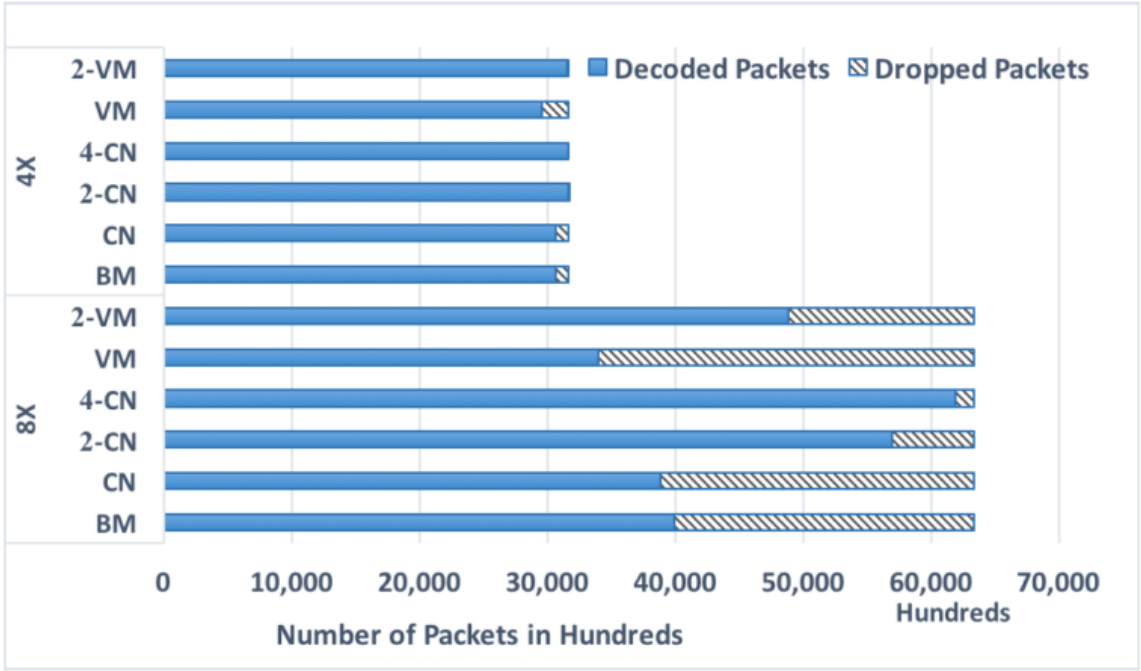

Fig. 11 Performance of Snort with multiple VM and container instances.

- ✓ The performance of Snort significantly increases when the incoming traffic is split among multiple instances.

Conclusions and Inspiration

- ✓ Containers can consume less CPU resource compared with VM, which improves the processing capacity for traffic.
- ✓ Containers consume less disk resource than VM, and are easier to migrate.
- ✓ Different attributes of different VNFs (e.g. single-thread or multi-thread) will affect the performance when allocated different volume of resource (e.g. CPU cores).
- ✓ Containers have less boot time, which contributes to reducing the latency.
- ✓ The network I/O technology for containers is not mature.

Thanks!