An Analysis and Empirical Study of Container Networks

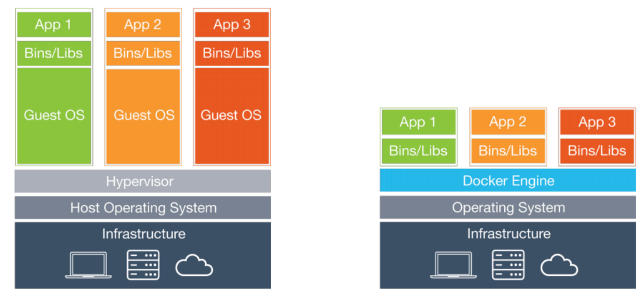
Kun Suo[*], Yong Zhao[*], Wei Chen[^], Jia Rao[*]

University of Texas at Arlington[*], University of Colorado, Colorado Springs[^]

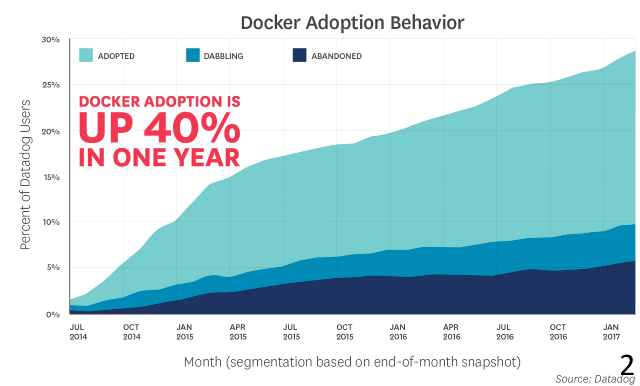
INFOCOM 2018@Hawaii, USA

The Rise of Containers

- Containers are a lightweight alternative to virtual machines for application packaging
- Key benefits of containers
  - ✓ Rapid deployment
  - ✓ Portability
  - ✓ Isolation
  - ✓ **Lightweight , efficiency, and density**
- Increasingly and widely-adopted in data centers
  - ✓ Google Search launches about **7,000** containers every second



(a) Architecture of a virtual machine (b) Architecture of Docker



Container Networks in the Cloud

Cloud	Network on a single host	Network on multiple hosts
Amazon EC2	Bridge (Default) None Host	NAT (Default) Overlay Third party solutions
Docker Cloud	Bridge (Default) None Container Host	Overlay (Default)
Microsoft Azure	NAT (Default) Transparent Overlay L2Bridge L2Tunnel	NAT (Default) Transparent Overlay L2Bridge L2Tunnel
Other clouds	…	…

- **Typical use case: containers running in VMs**
- **Challenging to select an appropriate container network**

Container Networking Projects

docker

vmware NSX

ROMANA

PROJECT CALICO

canal

midokura

cilium

OvS
Open vSwitch

weaveworks

nuagenetworks

Aporeto

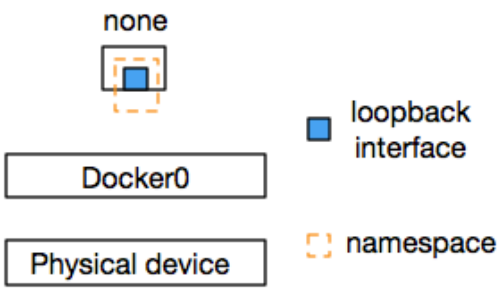
Container networks provide connectivity among isolated, sandboxed applications

- **A qualitative comparison**
  - ✓ Applicable scenarios
  - ✓ Security isolation

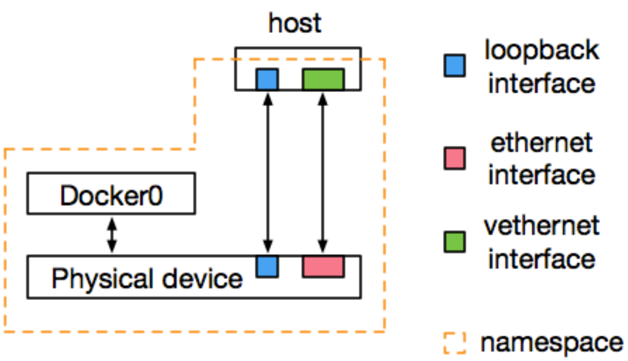
- **An empirical study**
  - ✓ Throughput / Latency
  - ✓ Scalability
  - ✓ Overhead/start-up cost

Container Networks on a Single Host

- **None**
  - ✓ A closed network stack and namespace
  - ✓ High security isolation

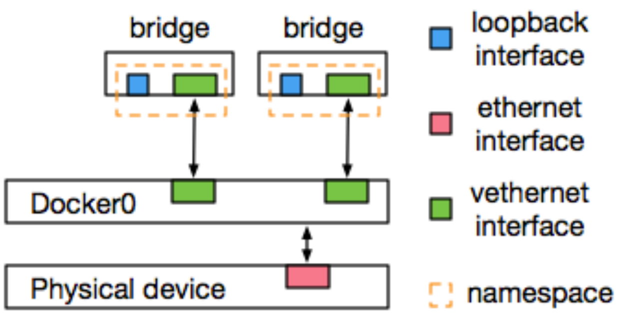


- **Host mode**
  - ✓ Share the network stack and namespace of the host OS
  - ✓ Low security isolation

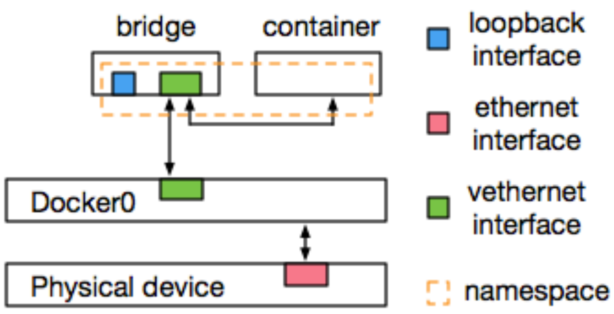


Container Networks on a Single Host

- **Bridge mode**
  - ✓ The default network setting of Docker
  - ✓ An isolated network namespace and an IP address for each container
  - ✓ Moderate security isolation



- **Container mode**
  - ✓ A group of containers share one network namespace and IP address
  - ✓ Low isolation within the same group and moderate isolation across groups

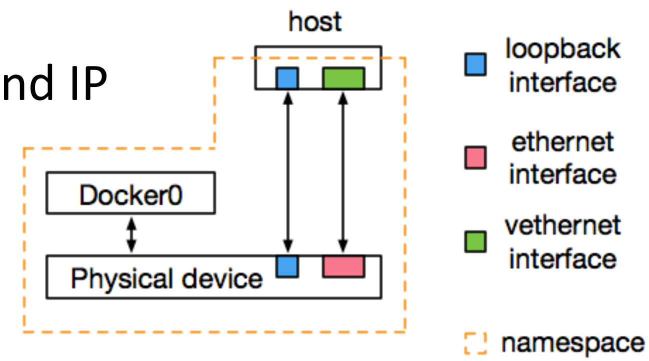


Container Networks on a Single Host

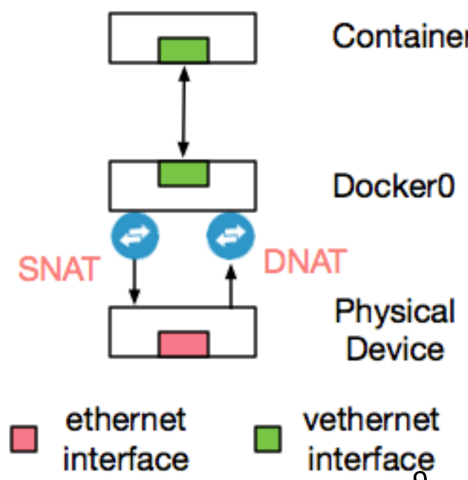
Network	Intra-machine communication	Inter-machine communication	Access to external networks	Namespace	Security
None	/	/	/	Independent, isolated	High
Bridge	docker0 bridge	/	Bind host port, NAT	Independent, isolated	Moderate
Container	Inter-process communication	/	Port binding, NAT	Shared with group leader	Medium
Host	Host network stack	Host network stack	Host network stack	Shared with the host OS	Low

Container Networks on Multiple Hosts

- **Host mode**
  - ✓ Communicate through host network stack and IP
  - ✓ Pros: near-native performance
  - ✓ Cons: no security isolation

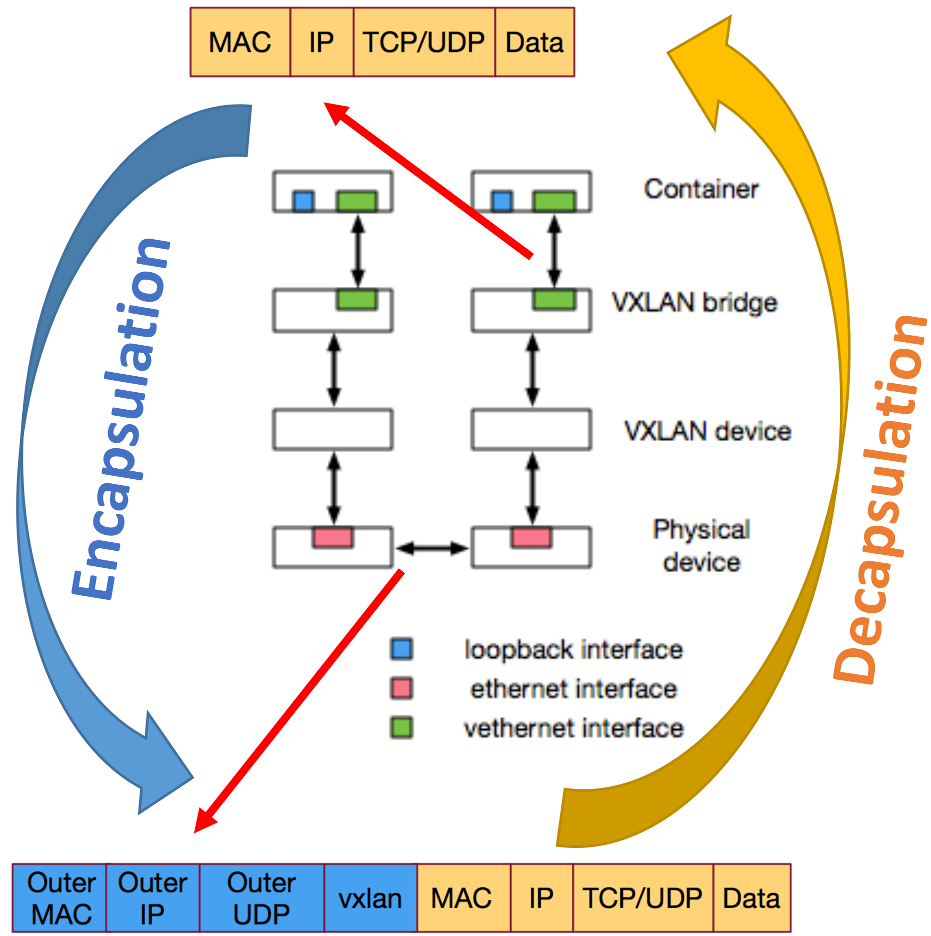


- **Network address translation (NAT)**
  - ✓ Bind a private container IP to the host public IP and a port number. The `docker0` bridge translates between the private and public IP addresses
  - ✓ Pros: Easy configuration
  - ✓ Cons: IP translation overhead, inflexible due to host IP binding and port conflicts



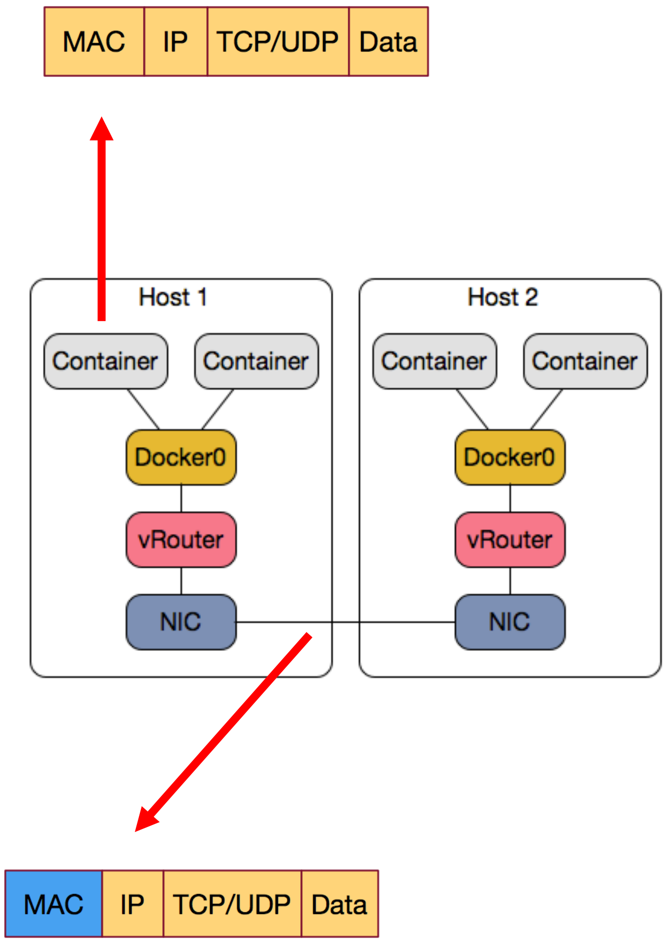
Container Networks on Multiple Hosts

- **Overlay network**
  - ✓ A virtual network built on top of another network through packet encapsulation
  - ✓ Examples: IPIP, VXLAN, and VPN, etc.
  - ✓ Pros: isolation, easy to manage, resilient to network topology change
  - ✓ Cons: overhead due to packet encapsulation and decapsulation, difficult to monitor



Container Networks on Multiple Hosts

- **Routing**
  - ✓ A network layer solution based on BGP routing
  - ✓ Pros: high performance
  - ✓ Cons: BGP not widely supported in datacenter networks, limited scalability, not suitable for highly dynamic networks or short-lived containers



Container Networks on Multiple Hosts

Network	How it works	Protocol	K/V store	Security
Host	Sharing host network stack and namespace	ALL	No	No
NAT	Host network port binding and mapping	ALL	No	No
Overlay	VXLAN or UDP or IPIP	Depends	Depends	Encrypted support
Routing	Border Gateway Protocol	Depends	Yes	Encrypted support

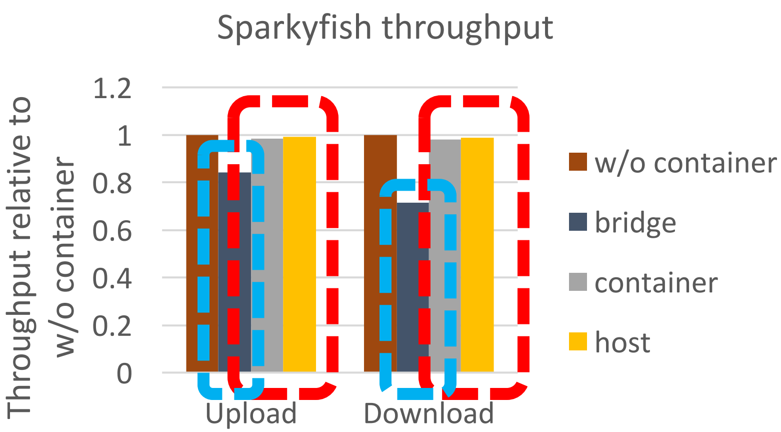
An Empirical Study

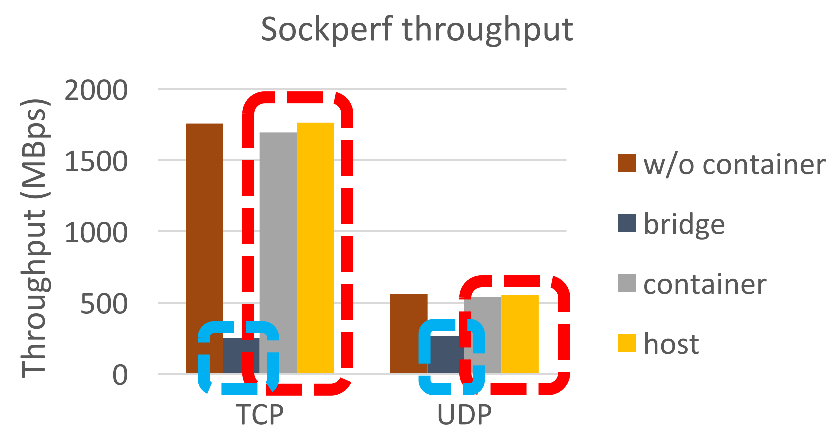
- Containers in a single VM
  - ✓ *How much are the overheads of single-host networking modes?*
- Containers in multiple VMs on the same PM
  - ✓ *How much are the overheads of cross-host networking?*
- Containers in multiple PMs vs. containers in multiple VMs on different PMs
  - ✓ *The interplay between VM network and container networks?*
- Impact of packet size and protocol
- Scalability and startup cost

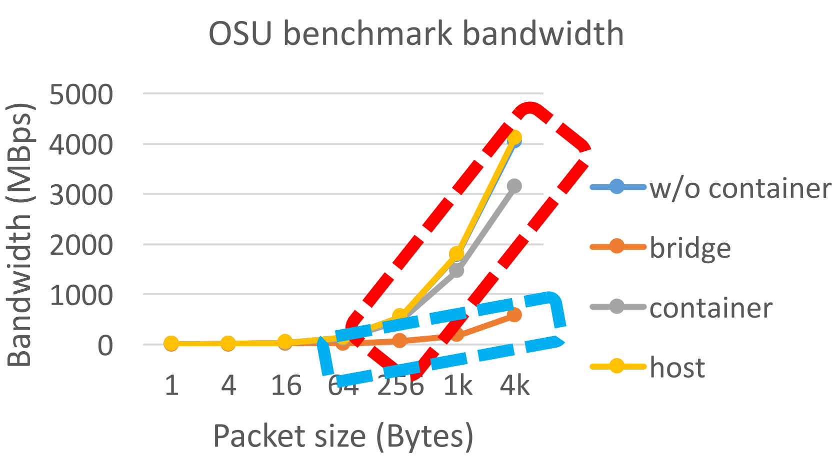
Experiment settings

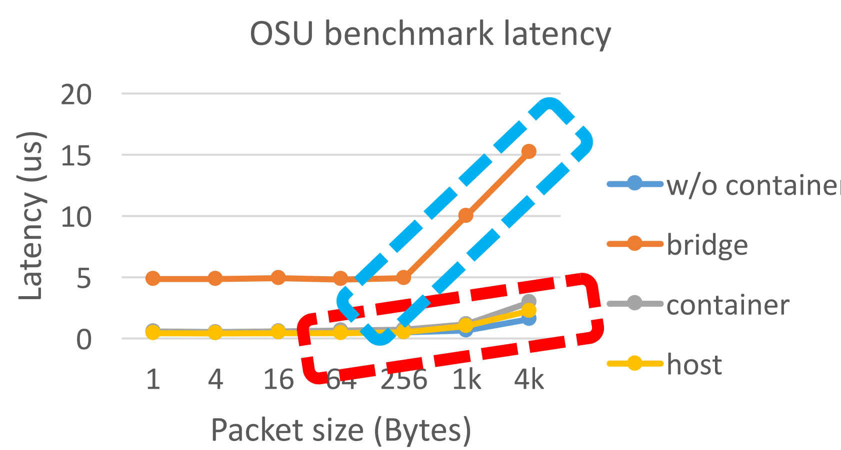
- **Hardware**
  - ✓ Two DELL PowerEdge T430 servers, equipped with a dual ten-core Intel Xeon E5-2640 2.6GHz processor, 64GB memory, a 2TB 7200RPM SATA hard disk, Gigabit Ethernet
- **Software**
  - ✓ Ubuntu 16.10, Linux kernel 4.9.5, KVM 2.6.1 as hypervisor, Docker CE 1.12, rtl8139 NIC drivers
  - ✓ Etcd 2.2.5, weave 1.9.3, flannel 0.5.5 and calico 2.1
- **Benchmarks**
  - ✓ Netperf 2.7, Sockperf 2.8, Sparkyfish 1.2, OSU benchmarks 5.3.2

Container Networking in a Single VM







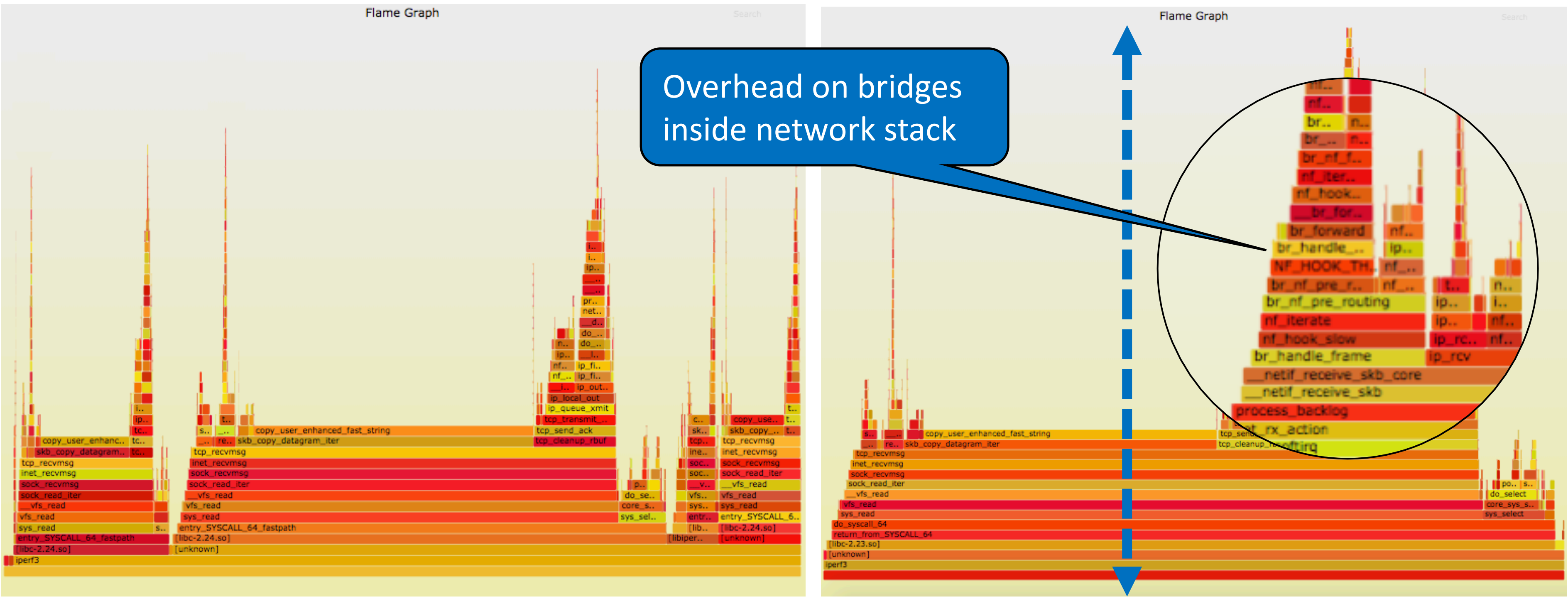


The **container mode** and **host mode** achieved close performance to the baseline while the **bridge mode** incurred significant performance loss.

Diagnosis of Bridge-based Container Networking

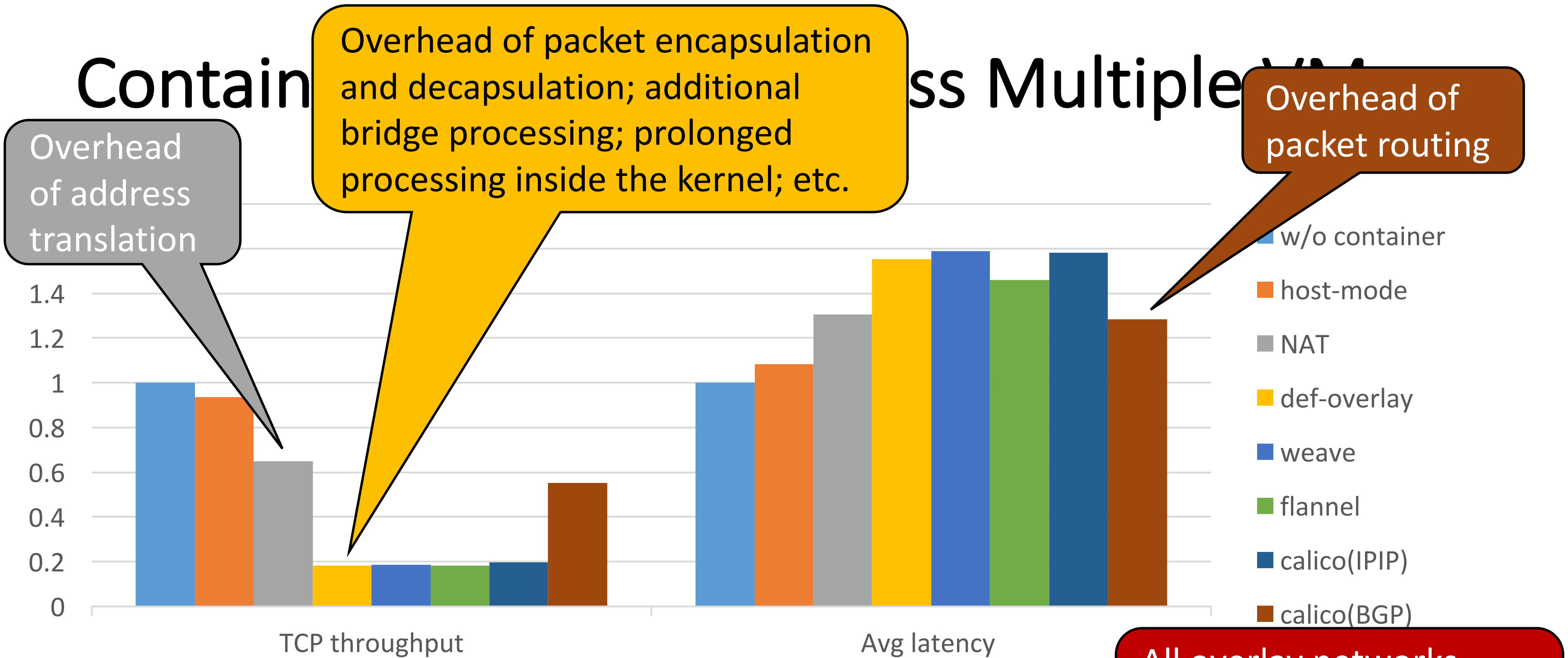
W/o container

Bridge mode



- Longer critical path of packet processing due to centralized bridge docker0
- Higher CPU usage and possible queuing delays

Contain ss Multiple

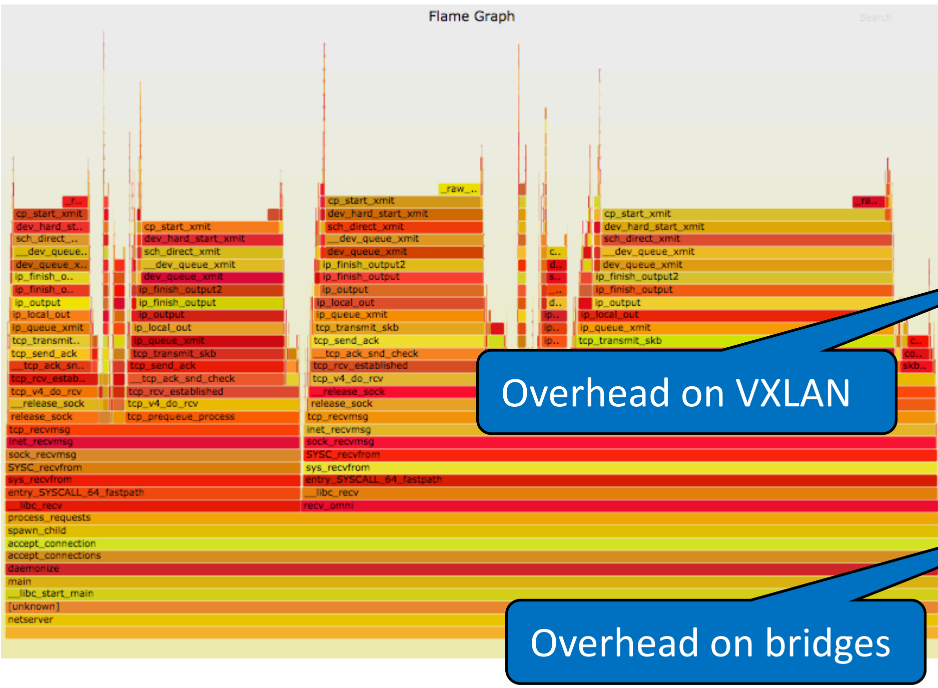


All overlay networks consumed much more CPU (Mostly in softIRQ).

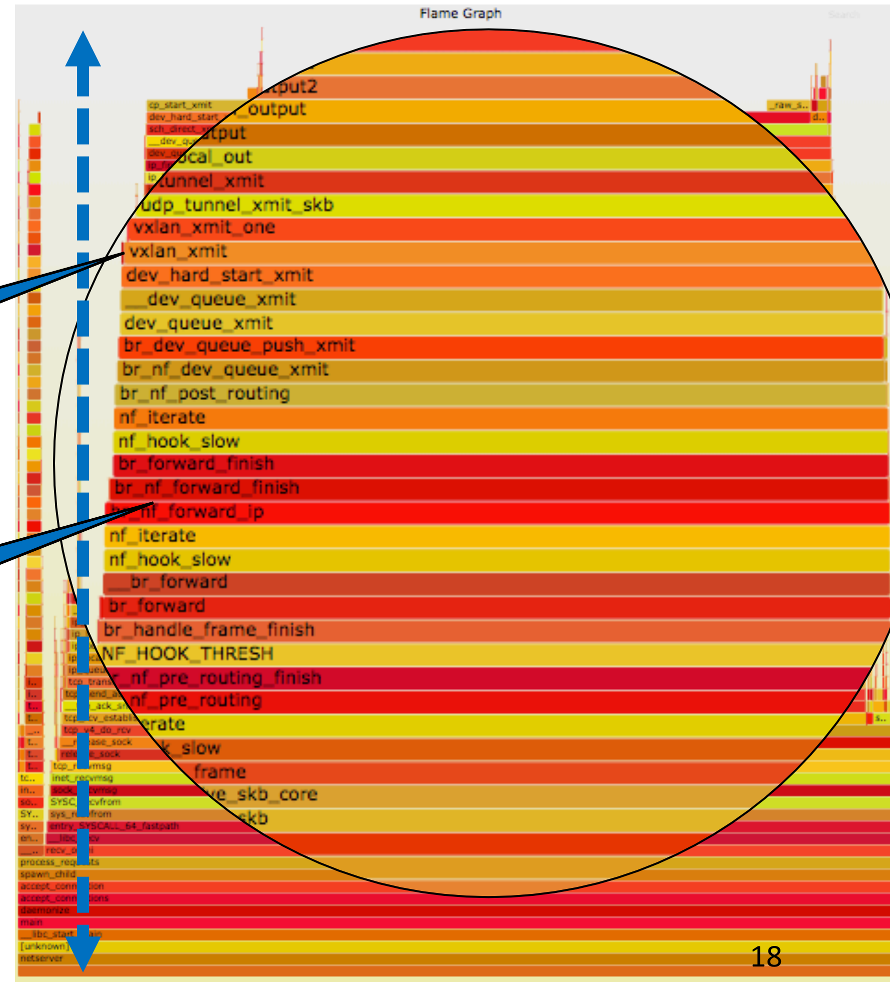
Network	CPU (c)	CPU (s)	S.dem (c)	
w/o container	33.37	18.41	75.466	1.626
Host mode	34.19	22.04	82.403	53.127
Def Overlay	38.75	42.91	95.867	106.145
Weave	54.92	47.56	150.678	130.478
Flannel	42.96	40.44	127.118	119.659
Calico(IPIP)	38.53	40.53	107.854	113.465
Calico(BGP)	37.72	36.92	99.665	95.035

Diagnosis of Overlay Networks

W/o container



Container in overlay network

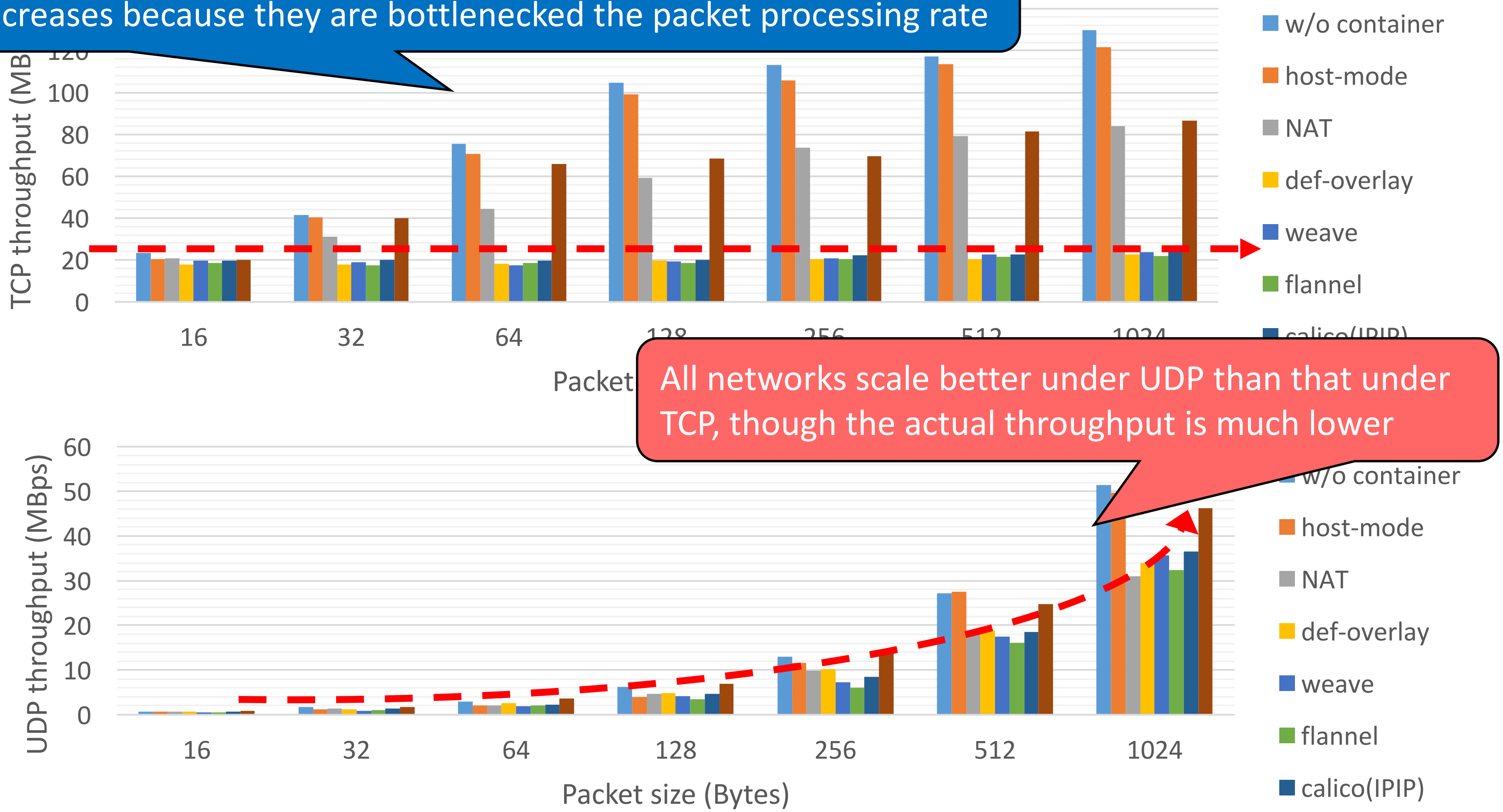


- much longer critical path inside the kernel
- more CPU usage, more **soft interrupt processing**

Impact of Packet Size and Protocol

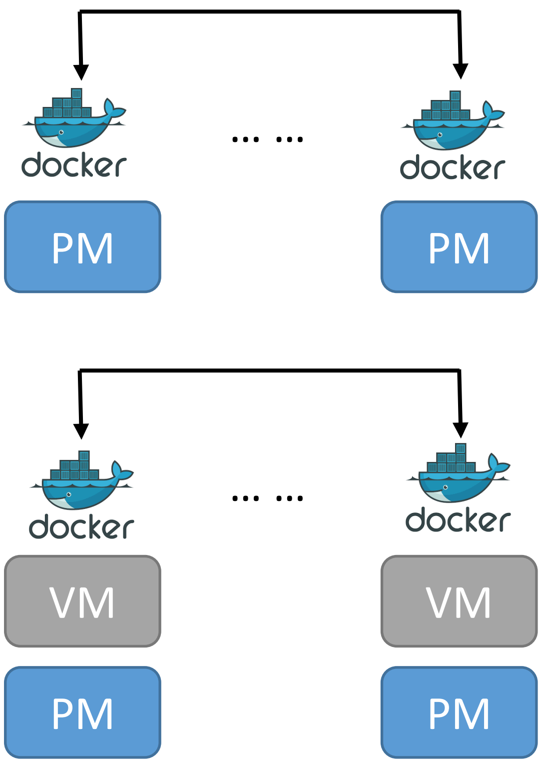
Under TCP, overlay networks do not scale as the packet size increases because they are bottlenecked the packet processing rate

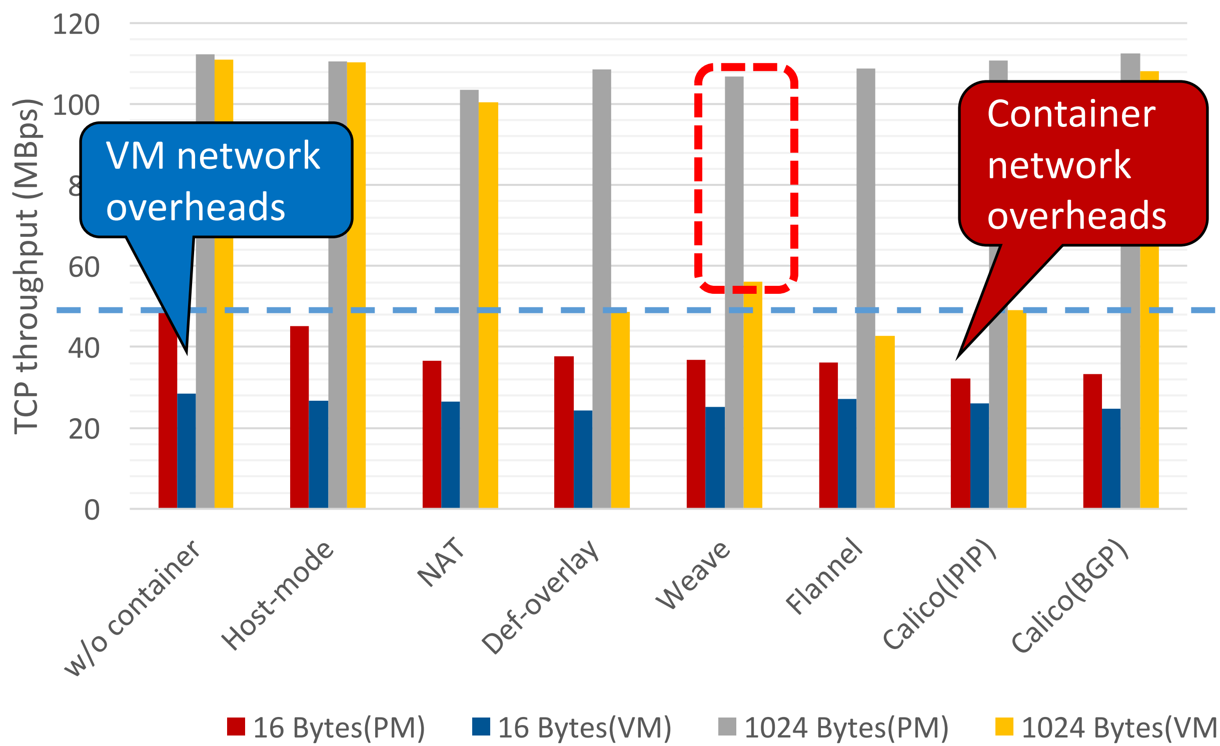
All networks scale better under UDP than that under TCP, though the actual throughput is much lower



Fixed packet rate, throughput should scale with packet size

Interplays between VM network virtualization and Container networks





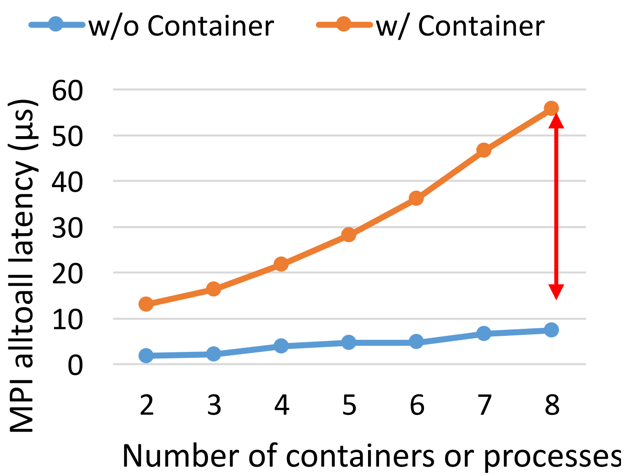
- The two-layer virtualization induces additional degradation on top of virtualization overheads
- Overlay networks suffer most degradation

Scalability

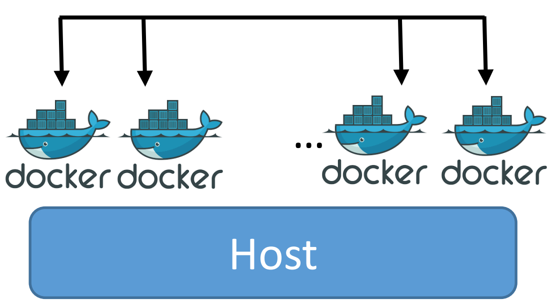
Bridge docker0 limits the scalability of container network in a single node

The communication over the overlay network is a major bottleneck for container network across hosts

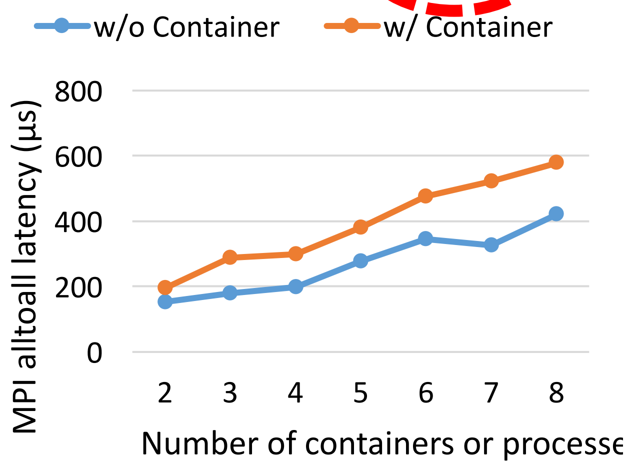
Containers on a single VM



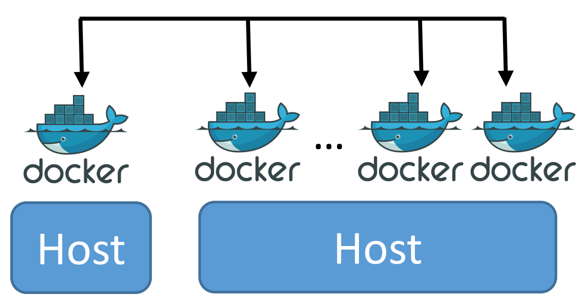
Bridge mode



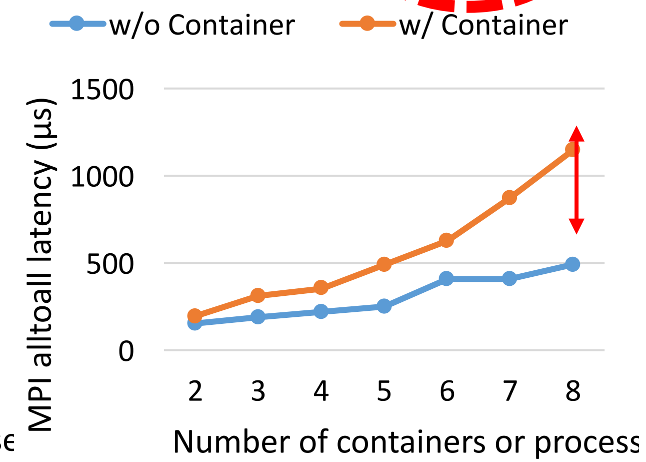
Containers on multi VMs (**1:N-1**)



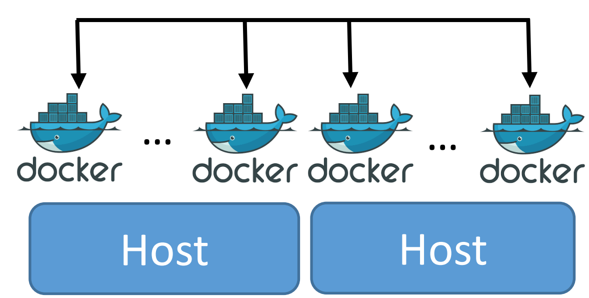
Docker overlay



Containers on multi VMs (**N/2:N/2**)



Docker overlay

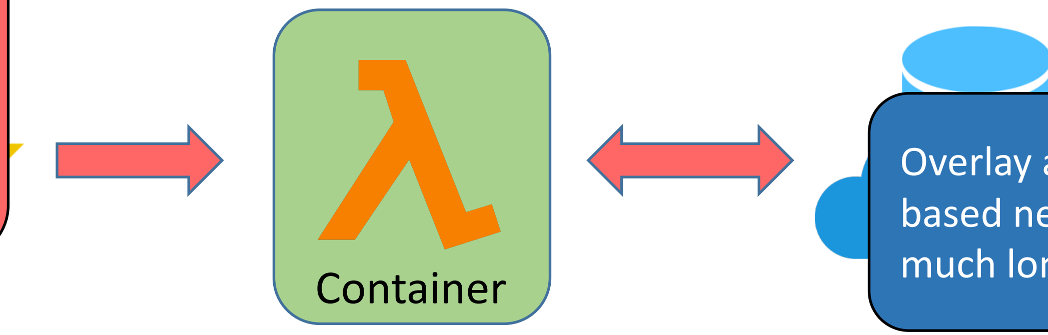


Network Startup Time

Network setup time in docker startup time; attaching to an existing network in the container mode requires least setup time

Function

Data / Cloud service



Overlay and routing-based networks require much longer startup time

Single host	Launch time	Multiple hosts	Launch time
None	539.6 ms	Host	497.4 ms
Bridge	663.1 ms	NAT	674.5 ms
Container	239.4 ms	Docker Overlay	10,979.8 ms
Host	497.4 ms	Weave	2,365.2 ms
/	/	Flannel	3,970.3 ms
/	/	Calico (IPIP)	11,373.1 ms
/	/	Calico (BGP)	11,335.2 ms

The startup time was measured as the time since a container create command is issued until the container is responsive to a network ping request.

Insights & Takeaways

- **Challenging to determine an appropriate network for containerized applications**
  - ✓ Performance vs. security vs. flexibility
  - ✓ Small packets vs. large packets
  - ✓ TCP vs. UDP

- **Bridging is a major bottleneck**
  - ✓ Linux bridge and OVS have the similar issue
  - ✓ Avoid bridge mode if containers do not need to access external networks

- **Overlay networks are most convenient but expensive**
  - ✓ The existing network stack is inefficient in handling packet encapsulation and decapsulation

- **Optimizing container networks**
  - ✓ Streamlining the asynchronous operations in the network stack
  - ✓ Making the network stack aware of packets of overlay networks
  - ✓ Coordinating VM-level network virtualization and container networks

Thank you !

Questions?