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
## Container Networks in the Cloud

<table>
<thead>
<tr>
<th>Cloud</th>
<th>Network on a single host</th>
<th>Network on multiple hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon EC2</td>
<td>Bridge (Default)</td>
<td>NAT (Default)</td>
</tr>
<tr>
<td></td>
<td>None</td>
<td>Overlay</td>
</tr>
<tr>
<td></td>
<td>Host</td>
<td>Third party solutions</td>
</tr>
<tr>
<td>Docker Cloud</td>
<td>Bridge (Default)</td>
<td>Overlay (Default)</td>
</tr>
<tr>
<td></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Container Host</td>
<td></td>
</tr>
<tr>
<td>Microsoft Azure</td>
<td>NAT (Default)</td>
<td>NAT (Default)</td>
</tr>
<tr>
<td></td>
<td>Transparent Overlay</td>
<td>Transparent Overlay</td>
</tr>
<tr>
<td></td>
<td>L2Bridge</td>
<td>L2Bridge</td>
</tr>
<tr>
<td></td>
<td>L2Tunnel</td>
<td>L2Tunnel</td>
</tr>
<tr>
<td>Other clouds</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

docker

NSX

ROMANA

CALICO

canal

mido
kura

cilium

OvS

nuage networks

A poreto
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

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

<table>
<thead>
<tr>
<th>Network</th>
<th>How it works</th>
<th>Protocol</th>
<th>K/V store</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Sharing host network stack and namespace</td>
<td>ALL</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NAT</td>
<td>Host network port binding and mapping</td>
<td>ALL</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Overlay</td>
<td>VXLAN or UDP or IPIP</td>
<td>Depends</td>
<td>Depends</td>
<td>Encrypted support</td>
</tr>
<tr>
<td>Routing</td>
<td>Border Gateway Protocol</td>
<td>Depends</td>
<td>Yes</td>
<td>Encrypted support</td>
</tr>
</tbody>
</table>
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
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
Overhead of packet encapsulation and decapsulation; additional bridge processing; prolonged processing inside the kernel; etc.

All overlay networks consumed much more CPU (Mostly in softIRQ).
Diagnosis of Overlay Networks

W/o container

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

Container in overlay network

Overhead on VXLAN

Overhead on bridges
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

Containers on multi VMs (1:N-1)

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

Bridge mode

Docker overlay

Docker overlay

Bridge mode

Docker overlay

Docker overlay

The communication over the overlay network is a major bottleneck for container network across hosts.
Network Startup Time

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

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.

Overlay and routing-based networks require much longer startup time.

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