Exploration of Large Scale Virtual Networks

Open Network Summit 2016

David Wilder
wilder@us.ibm.com
A Network of Containers

Each container has a unique “network name-space” consisting of:
- Virtual network interfaces
- Routing and arp tables
- Network statistics
Project Objectives

- Build and evaluate large virtual networks on Linux using:
  - Docker,
  - Linux Bridge,
  - OpenVSwitch.

- Learn how virtual networks:
  - scale.
  - perform.
  - break.
What I Measured:

- Aggregate bandwidth
- TCP connection rate
- Latency – broadcast and unicast
- Up to 1600 docker nodes
- IPV4
Some details..

- 80 processor power8 (SMT=8)
- 132 GB
- Linux kernel version: 4.4.0 rc5+ (ppc64LE)
- OpenVswitch V2.5.90
- Docker 1.5.0
Throughput Measurements

- OpenVswitch
  - In kernel flow table
  - ovs-vswitchd

- Docker containers
  - 800 network streams

- iperf clients
  - 1
  - 2
  - 4
  - ...
  - 800

- iperf servers
  - 801
  - 802
  - 803
  - ...
  - 1600
Predictions

- Iperf is not performing any real I/O and will become cpu bound.
  - CPU load average $\approx$ Number of iperf streams.
  - Throughput will level out when the number of stream equals the number of processors.
CPU Utilization:

Load average: measured vs predicted.
Network: OpenVswitch
Number of CPUs: 80
Aggregate Bandwidth

Aggregate bandwidth vs. number of network streams.
Network: openVswitch
Number of CPUs: 80
Linux Bridge compared to OpenVswitch

Aggregate Bandwidth: OpenVswitch Vs. Linux Bridge.
Number of CPUS: 80
Issues

- **Linux Bridge**
  - Limited to a maximum of 1024 ports.
  - Linux/net/bridge/br_private.h
    - `#define BR_PORT_BITS 10`
    - `#define BR_PORT_BITS 14 /* (16k ports) */`

- **OpenVswitch**
  - At over 400 streams iperf sporadically failed to establish a connection returning the error “No route to host”.
Measuring TCP Connection Rate

Ovs-benchmark

ovs-vswitchd (user)

112,010 netlink sockets

Connection requests

OpenVswitch (kernel)
Flow Table

Docker containers

1
2
4
800

801
802
803
1600

Docker containers
TCP Connection Rate

Connection Rate: OpenVswitch Vs. Linux Bridge.
Number of CPUS: 80

Connections/Second vs. Number of streams

- openVswitch
- Linux Bridge
Measuring latency of unicast and broadcast packets.

Sender

OpenVswitch / Linux Bridge

Broadcast and Unicast UDP packets with timestamps

Receivers

Idle containers

1600
Unicast Latency

Average latency for unicast packets: OpenVswitch Vs. Linux Bridge. Number of CPUs: 80
Packet Flooding.

sysctl -w net.core.netdev_max_backlog=10000
Broadcast Packet Latency

Average Latency for broadcast packets: OpenVswitch Vs. Linux Bridge.
Number of CPUs: 80

- **OpenVswitch**
- **Linux Bridge**
Multiple Bridge Configuration - Eleven Bridges

- 1 Sender
- Same bridge: Bridge br1, Bridge br10, Bridge br9
- Different bridge: Bridge br2, Bridge br3, Bridge br4, Bridge br5, Bridge br6, Bridge br7, Bridge br8, Bridge br9, Bridge br10

Up to 1600 Containers were evenly distributed between br1 through br10
Broadcast Latency – single vs. multiple bridges

Average Latency for broadcast packets: Single Bridge vs. Multi-bridge. Number of CPUs: 80

- Blue line: 11 Linux Bridges BR_PORT_BIT=10
- Red line: Single bridge BR_PORT_BIT=14
Tuning

- Prevent broadcast packet loss due to overflowing per-cpu queue:
  
  ```
  sysctl -w net.core.netdev_max_backlog=10000
  ```

- Prevent Neighbor (ARP) table overflows
  
  ```
  sysctl -w net.ipv4.neigh.default.gc_thresh1=1024
  sysctl -w net.ipv4.neigh.default.gc_thresh2=2048
  sysctl -w net.ipv4.neigh.default.gc_thresh3=4096
  ```

  ```
  dmesg: neighbour: arp_cache: neighbor table overflow!
  ```
Increasing the nofiles limit for ovs-vswitchd

- Ovs-vswitchd creates many netlink sockets to communicate with the kernel. 112K sockets in our set-up.

- Each netlink socket requires a file descriptor.

- The value of nofiles for the ovs-vswitchd must be at least: (Number of cpus * Number of switch ports).

```
$ prlimit -p \\
  `cat /var/run/openvswitch/ovs-vswitchd.pid` \\
  --nofile=200000
```
In Summary.....

- Throughput becomes a function of CPU (loadaverage).
- Consider how contention for CPU will affect throughput.
- Massive layer 2 domains will have a large broadcast latency.
- I saw a decreased latency when using multiple bridges.
Legal Statement

This work represents the view of the authors and does not necessarily represent the view of IBM.
IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries.
Linux is a registered trademark of Linus Torvalds.
Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
Other company, product, and service names may be trademarks or service marks of others.