Accelerate Container Network Performance

Founder of CloudNetEngine
Jun Xiao

DPDK Summit - San Jose – 2017
Agenda

• Problem statements
• Performance comparisons
• Techniques behind the scene
• Q & A
Performance comparisons
Test workloads:
- ENT. workload (iperf3) in clear linux
- NFV workload (DPDK testpmd) in native container
Host H/W
- CPU: Xeon E5-2620 v3 2.40GHz
- 6 physical cores, 12 logical cores
- NIC: XL710 40-Gigabit Ethernet
- MEM: 32G

Host S/W
- Ubuntu 16.04 + KVM
- qemu-lite 2.7.1
- docker.ce 17.09

Native Container config (for NFV test):
- 1G hugepage for testpmd
- 2 virtio_user virtual devices
- testpmd from dpdk 17.05.02

Clear Linux config (for NON NFV test)
- 2 vCPUs
- 1G hugepage
- 1 virtio PCI device
- iperf 3.1.1

Performance Test Configuration

Virtual Switches Under Test

Native OVS
- OVS 2.8
- kernel module bundled with Linux kernel 4.4.0

OVS-DPDK
- OVS 2.8
- DPDK v17.05

CNE vSwitch
- CNE vSwitch 2.0
Clear Linux
Iperf Server
virtio kernel driver
virtio PCI device

Clear Linux
Iperf Client
virtio kernel driver
virtio PCI device

Host1
vswitch

Host2
vswitch

XL710 40G

iperf TCP Two Hosts Test Topology

Gbps (Higher is better)

VM1-VM3

Native OVS
OVSDPDK
CNE vSwitch

CPU % (Lower is better)

Host1
Host2

Native OVS
OVSDPDK
CNE vSwitch
Native Container

Testpmd (io mode fwd)

Virtio PMD driver

Host1

Vswitch

Host2

Pktgen-DPDK

XL710 40G links

NFV PVP 64 bytes Test Topology

MPPS (Higher is better)
- Native OVS (N/A)
- OVS-DPDK
- CNE vSwitch

CPU % (Lower is better)
- Native OVS (N/A)
- OVS-DPDK
- CNE vSwitch
Techniques behind the scene
Performance is meaningless without reliability

- ovs-vswitchd
  - Control / Data plane decoupling!
  - Generic Netlink

- vSwitch datapath
  - virtual ports for containers / virtual machines
  - Switching core
  - Uplink ports
Offloading

- Packet categories:
  - TSO (native, VLAN, QINQ, VXLAN, GENEVE, VLAN+VXLAN, VLAN+GENEVE...)
  - UFO (native, VLAN, QINQ, VXLAN, GENEVE, VLAN+VXLAN, VLAN+GENEVE...)
  - CKSUM (native, VLAN, QINQ, VXLAN, GENEVE, VLAN+VXLAN, VLAN+GENEVE...)

- Egress port “HW” capabilities:
  - VLAN
  - QINQ
  - TSO
  - UFO
  - CKSUM
  - VXLAN
  - GENEVE

- Mapping a packet requested offload to egress port “HW” capabilities:
  - Doing S/W offload if constraints are not meet
Worker threads load balance

- Place a new RXQ to the least loaded worker thread.
- Migrate a heavy load RXQ to the least loaded worker thread.
CPU efficiency: Interrupt + polling

- Pure polling is very bad for CPU efficiency under light load.
- vNICs and pNICs have very different interrupt handling characters.
- Performance for heavy load should be on par with pure polling.

A worker thread’s main loop

- Process “busy” mode rxqs
  Demotion “busy” rxqs to “intr” mode if criteria meets

- Epoll all fds of “intr” mode rxqs
  (epoll will block if no outstanding “busy” mode rxq)

- Process “intr” mode rxqs which have interrupts arrived
  Promotion “intr” rxqs to “busy” mode if criteria meets
“ZERO” copy

- Sounds simple? But not really as virtio TXQ has limited TX descriptors:
  - Traffic shaping queuing
  - Uplink TX is asynchronous thus queuing
  - Possible other queuing mechanisms ...

- DPDK vhost_user lib zero copy mechanism has many limitations.
- Have to copy reasonable length of packet header to prevent guest attacks.
Uniform architecture to support ENT./NFV workloads

- A net chain is RCU protected so that updating a net chain won’t have any performance penalty on the datapath.
- Most of the features supporting ENT. Workload can be implemented as a net chain entry, e.g. offloading, shaping, LRO etc.
- Packet group metadata is introduced to very quickly decide whether a net chain entry is applicable to a packet group or not, thus no per packet processing is spending on non applicable entries.
High performance packet sniffer

tcpdump
libpcap
vSwitch datapath adapter

Control channel
Packet channel over shared memory

vSwitch datapath

Sniffer proxy
BPF JIT Filter (net chain entry)

Switching core logic

virtual ports for containers / virtual machines

Uplink ports
Q & A

www.cloudnetengine.com

Jun.xiao@cloudnetengine.com

Twitter: @cloudnetengine
CloudNetEngine virtual switch architecture

Openflow APIs
OVSDP APIs

ovs-vswitchd* (dpif-netlink)
ovepdb-server

OVS datapath APIs
Contrack*
FWD engine*
Multi queue
Jumbo frame
Vhost user backend
Timer mgmt
Mem mgmt
Adaptive poll
Scheduler

DPDK*

CDP (CloudNetEngine Data Path)

CDP APIs
Net chain
tunneling
QoS
Bonda
Ofloading
Sniffer
Host H/W
- CPU: Xeon E5-2620 v3 2.40GHz
- NIC: XL710 40-Gigabit Ethernet
- MEM: 32G

Host S/W
- Ubuntu 16.04 + KVM
- qemu 2.5.1

Guest H/W
- 4 vCPUs/ 2vNICs/ 4G memory for NFV tests
- 1 vCPUs/ 1vNICs/ 1G memory for non-NFV tests
- virtio offload flags are all enabled
- vNICs use default queues

Guest S/W
- buildroot kernel 4.4.3 x86_64
- testpmd io mode forward for NFV test
- iperf 3.1.1

Virtual Switches Under Test

Native OVS
- OVS 2.8
- kernel module bundled with Linux kernel 4.4.0

OVS-DPDK
- OVS 2.8
- DPDK v17.05

CNE vSwitch
- CNE vSwitch 2.0

Performance Test Configuration
Virtual Machine
- Iperf Server
  - virtio kernel driver
  - virtio PCI device

Virtual Machine
- Iperf Client
  - virtio kernel driver
  - virtio PCI device

Host1
- vswitch

iperf TCP Single Host Test Topology

Graphs:
- Gbps (Higher is better)
- CPU % (Lower is better)

Legend:
- Native OVS
- OVS-DPDK
- CNE vSwitch
NFV PVP 64 bytes Test Topology

Virtual Machine
testpmd (io mode fwd)
virtio PMD driver
virtio PCI device

Host1
vswitch
XL710 40G links

Host2
Pktgen-DPDK

MPPS (Higher is better)

<table>
<thead>
<tr>
<th></th>
<th>Native OVS</th>
<th>OVS-DPDK</th>
<th>CNE vSwitch</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPPS</td>
<td>3</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

CPU % (Lower is better)

<table>
<thead>
<tr>
<th></th>
<th>Native OVS</th>
<th>OVS-DPDK</th>
<th>CNE vSwitch</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU %</td>
<td>300</td>
<td>100</td>
<td>150</td>
</tr>
</tbody>
</table>