Userspace OVS with HW Offload and AF_XDP

Linux Plumber
Aug 24, 2020
William Tu, VMware
Agenda

Introduction
• OVS Kernel and userspace datapath
• DPDK and AF_XDP netdev interface
• tc-flower and rte_flow offload

Design and Evaluation
• Userspace datapath with tc-flower offload and AF_XDP
• Performance
What is OVS?

SDN Controller

OpenFlow

ovs-vswitchd

Datapath

Fast Path

Slow Path
OVS Linux Kernel Datapath

Slow path in userspace

Fast Path in Kernel

1. Kernel module: openvswitch.ko
2. Works on a wide variety of distributions and versions
3. Well tested, widely used, but with kernel overhead
OVS Userspace Datapath

SDN Controller

ovs-vswitchd

Userspace Datapath

Both slow and fast path in userspace

1. Used by OVS-DPDK
2. Fast due to kernel by-pass
3. Deploy/Debug DPDK is hard
4. Usually for appliance with dedicated HW

DPDK library

Hardware
Motivation

Customers deploy either one of the two:

- OVS Kernel Datapath:
  - stable, feature-rich, and for typical hypervisor/enterprise
- OVS-DPDK Userspace Datapath
  - high performance, used in appliance

However,

- Maintaining and running both datapaths is hard
- Can we have single datapath for both use cases?
XDP and AF_XDP

• XDP: eXpress Data path
  • An eBPF hook point at the network device driver level

• AF_XDP:
  • A new socket type that receives/sends raw frames with high speed
  • Use XDP program to trigger receive
  • Userspace program manages Rx/Tx ring and Fill/Completion ring.
  • Zero Copy from DMA buffer to user space memory, achieving line rate (14Mpps).

From “DPDK PMD for AF_XDP”
OVS AF_XDP netdev

**Goal**
- Available since OVS 2.12
- Use AF_XDP socket as a fast channel to userspace OVS datapath
- Flow processing happens in userspace
- Same datapath as used by OVS-DPDK
Performance

Physical-to-physical port, using 64B 1 flow and 1K flows, with different packet I/O

- Different Datapaths:
  - **KD**: Kernel Datapath
  - **UD**: Userspace Datapath

- Different Packet I/O:
  - **Phy-ovs-afxdp**: OVS’s AF XDP packet I/O code on physical port.
  - **Phy-dpdk-ixgbe**: DPDK’s ixgbe PMD on physical port
Summary

• Userspace datapath with AF_XDP netdev performs
  • Much better than kernel datapath
  • Slower than using DPDK netdev

• Future work
  • More improvement on the OVS AF_XDP netdev
  • Explore the idea of OVS HW offload:
    OVS-DPDK -> rte_flow, OVS kernel datapath-> tc-flower

• Observation
  • With AF_XDP netdev, userspace datapath can enable tc-flower offload.
HW offload: rte_flow v.s tc-flower APIs

- OVS HW offload interface
  - Translate the datapath flow into rte_flow or tc-flower
  - ovs/lib/netdev-offload- {dpdk, tc}.c

- Vendor driver
  - Check whether the API is implemented in vendor-specific driver
Target Use Case: Tunnel and Conntrack

• Simple 5 tuple match and action no longer meets today’s firewall requirements, typical use cases:
  • Each packet goes through three OVS datapath flows

• Example for incoming packets (rx):
  1. Match and Tunnel decap (ex: Geneve or VxLAN), Recirc
  2. Match on tunnel md and send to Connection tracking, Recirc
  3. Match on CT states and forward/drop

Requirements:
  A. Need to do all of them in hardware, no partial offload.
  B. if not A, process the flow in a fast SW path.
A. Kernel DP + tc-flower

**Pros:**
- Support tunnel and connection tracking
- Better integrate into Linux kernel
- Easier to ship and test

**Cons:**
- Fall-back performance in software OVS or TC (2Mpps)
B. DPDK rte_flow

Pros:
- Support tunnel encap
- Better SW fallback performance
- More active in community from different vendors

Cons:
- No connection tracking API support
- Need to deploy OVS-DPDK

https://events19.linuxfoundation.org/events/dpdknorthamerica2019/program/schedule/
A flow could be processed:

1. In HW with tc-flower, if not
2. In XDP, which is safe and performant, if not
3. In OVS userspace with AF_XDP

- Better integration into Linux kernel
- Better fallback performance in XDP/userspace
- Each stage has its own limitations, need to probe its supported features
C. Userspace Datapath + tc-flower + AF_XDP

Performance of P2P using 1 flow, 64B UDP packet:

A. HW offload: 31Mpps
B. HW offload + VxLAN encap: 21Mpps
C. XDP[1]: 3.5Mpps
D. Userspace DP + AF_XDP poll-mode: 4.5Mpps (uses 2 cores)

Summary of using Approach-C

For enterprise use case:
• Use userspace datapath with AF_XDP **interrupt**-mode netdev
• If need more performance, enable software XDP processing or AF_XDP **polling**-mode

For high performance use case:
• Use HW offload through tc-flower (fastest)
• Use software XDP processing (2nd)
• Use userspace datapath with AF_XDP polling-mode netdev
Future Work

• Validate tc-flower offload for conntrack and tunnel decap

• OVS XDP Processing patch
  • [ovs-dev] [PATCH v4 0/5] XDP offload using flow API provider
  • https://mail.openvswitch.org/pipermail/ovs-dev/2020-August/373915.html

• More optimization for OVS AF_XDP netdev
  • OVS with AF_XDP - what to expect, OVS conference 2019
Thank you