Off-path SmartNIC Port Binding with OVN

OVS+OVN '21
SmartNICs

• An overloaded term

• Some classifications include:
  ■ Packet processing flow:
    • Off-path SmartNICs
    • On-path SmartNICs
  ■ Hardware design:
    • ASIC-based
    • FPGA-based

• We will focus on **ASIC-based off-path** SmartNICs
Off-path SmartNIC

User space
- OVN controller
  - ovsdb
  - ovs-vswitchd

Host kernel
- br-int
- OVS System Datapath
  - br-bond0
    - bond0 HW LAG
    - P0
    - P1
- VF0
- VF1
- VF drivers

Add-in Card Hardware
- NIC
  - NIC Switch ASIC
  - Internal CPUs
  - Firmware
  - uplink 0
  - uplink 1

Host Process

VF0
VF1
VF drivers
Off-path SmartNIC DPU

- **User space**
  - No networking agents
  - Host Process

- **Host kernel**
  - PF driver (no eswitch)
  - VF driver
  - PF0
  - VF0
  - VF1

- **Add-in Card Hardware**
  - **DPU Userspace**
    - OVS system Datapath
    - br-int
    - VF0 rep
    - VF1 rep
    - ovs-tc
  - **DPU Kernel**
    - NIC Switch ASIC
    - Internal CPUs
    - Firmware
    - PCIe 1
    - uplink 0
    - uplink 1

- **Dedicated:**
  - CPU
  - RAM
  - Kernel
  - Flash
  - PCIe Complex
DPU: key takeaways

- Data Processing Unit (DPU)
  - Embedded system: **dedicated CPU**, NIC and other components
  - NIC is integrated with the main board using an I/O interconnect (e.g. PCIe)
  - NIC is shared by dedicated CPU and host CPU via **separate** I/O hierarchies

- Off-path architecture:
  - **Slow path**: packets flow via NIC cores
    - OVS system datapath (offload via tc)
    - OVS + DPDK (offload via rte_flow)
  - **Fast path**: direct flow via ASIC to the destination bypassing NIC cores
DPU: Control Plane Challenges

● Current infra software expects same-host topology (OpenStack, K8s, LXD etc.)
  ○ Hypervisor hostname ≠ OVS hostname on DPU
    ■ PCI addresses cannot be relied on to refer to PFs & VFs on different hosts
    ■ VF allocation on a hypervisor but programming on a DPU?
    ■ Hypervisor to DPU mapping? Multiple DPUs per host?

● Security boundary between host and DPU
  ○ How can we keep hypervisor & DPU isolation?
  ○ Generic host <-> DPU communication methods?
DPU: Control Plane Challenges

- CMS support
  - **Local** OVS & ovn-controller is expected for port/VIF plugging into bridges
  - Scheduling concerns: metadata about legacy SR-IOV vs offload vs DPU VFs
  - Missing control plane for remote VIF plugging and programming
  - Need intelligent chassis hostname lookup based on serial numbers

- OVN
  - Need to support port plugging directed by a CMS via **Logical Switch** Port info
  - Port plugging is traditionally outside of the OVN’s responsibility
DPU Support Design Goals

- Reuse the existing code as much as possible
- CMS-agnostic architecture (applicable to OpenStack, K8s, LXD etc.)
- Generic handling of different device families & vendors
- Upstream first approach
  - Rely on upstream projects and APIs (tc flower, OVS, switchdev, OVN)
  - Minimize new software or upstream it
- Avoid remote code execution on DPU by the hypervisor host
  - Analogy: treat DPU as a ToR Switch in a server
Auto-discovery

- Different hostnames
- Different PCIe topologies
- Same NIC
- Same PCI Vital Product Data (VPD)
- Same board serial number:
  - Unique
  - Read-only (ROM)
- Exposed on PFs if present
- Optionally on VFs

- Can reference DPU of a VF by SN
- CMS will use it for chassis hostname lookup

**(hostname 1)**

**DPU**

**NIC**

**Hypervisor**

**OVN chassis on DPU**

**CMS**

Different hostnames may exist:

- Can reference DPU of a VF by SN
- CMS will use it for chassis hostname lookup

**OVN**

**Board Serial Number**

**PCI VPD**

**DPU NIC**

**PF0**

**VF0**

... PF1

... VF0

hostname 2

**Hostname 1**

**PCle 1**

**PCle 2**

**P0**

**P1**

**P0**

**P1**

**hpf0**

**vf0hpf0**

**hpf1**

**vf0hpf1**

**CMS**

**PCle 1**

**PCle 2**

**Different hostnames**

**Different PCIe topologies**

**Same NIC**

**Same PCI Vital Product Data (VPD)**

**Same board serial number:**

- Unique
- Read-only (ROM)

**Exposed on PFs if present**

**Optionally on VFs**
• Appeared in PCI 2.1 local bus spec, inherited in PCIe
  ○ fully compatible format in PCIe
• Optional in the specs
  ○ but modern NICs have it
• Board Serial Number: unique, read-only, factory assigned (ASCII alphanumeric)
• Exposed on PFs
  ○ Firmware may optionally expose VPD on VFs
• Linux 2.6.26+ exposes a binary VPD blob via sysfs
Why Board Serial?

- How and how many PFs are exposed depends on a particular device
  - Some HW can expose more PFs then there are uplinks
    - Depends on a firmware config and device family
  - Some PFs can be inactive when HW bonding is used
  - PFs are **virtual ports of the NIC switch** while we need a static entity
- MAC addresses are port-level IDs while the board SN is a board-level ID
  - MACs can reprogrammed while VPD SN cannot
  - What is a **burned-in MAC** for a **virtual** NIC switch port?
- => PF MAC usage becomes **unreliable** for DPU hostname mapping
OVN VIF Plug Provider framework

- Core OVN
  - CMS API
    - Logical_Switch_Port options
      - requested-chassis
      - plug-type
      - plug-mtu-request
    - Each VIF plug provider implementation provide namespaced options for lookup.
  - New requested_chassis column in Southbound DB populated by northd
    - Allows each ovn-controller to effectively monitor ports destined to it prior to having the port plugged
  - VIF plug providers register callbacks that OVN will use for lookup to inform insert/delete operations for ports/interfaces in the local Open vSwitch instance

OVN VIF

- **OVN VIF**
  - System and device specific code is kept in a separate project called OVN VIF
  - Ready to host multiple VIF plug provider implementations, will follow core OVN release cadence and branch strategy
  - GitHub PR based workflow, patches welcome!
    - [https://github.com/ovn-org/ovn-vif](https://github.com/ovn-org/ovn-vif)
  - **representor**
    - **CMS API**
      - Logical Switch Port options
        - vif-plug:representor:pf-mac
        - vif-plug:representor:vf-num
    - Supports SmartNIC DPUs that provide kernel representor ports exposed through the [devlink-port](https://github.com/ovn-org/ovn-vif) infrastructure
    - Provides devlink library based on the (great) Open vSwitch netlink library code
    - Support for runtime updates to lookup tables [in progress](https://github.com/ovn-org/ovn-vif)
Features/Functionality being added to support DPU SmartNIC use case

CMS integration: OpenStack & K8s

SmartNIC
ARM CPU

ovn-controller w/ovn-vif
ovs-vswitchd
ovsdb

NIC Switch
Internal CPUs
Firmware

OVS br-int
VF Rep
VF Rep
PF Rep
PF Rep
uPF
uPF

Uplink PFs bonded
hidden

Libvirt
Nova Compute
QEMU process
Kubelet
CNI

Placement
Neutron

Nova API
K8s API

OVN NB DB
OVN SB DB

uPF
uPF
uplink
uplink

DC fabric

TBD
5. Update port with VF details (bind)

Board Serial Number: MT2113X4242E
VF Logical Number: 1
PF MAC: a0:36:9f:00:00:42
plug-type: representor
plug:representor:pf-mac: a0:36:9f:00:00:42
plug:representor:vf-num: 1

6. Create LSP
requested-chassis: dpu-hostname-42

7. Create a paused VM With a VF attached
8. Wait for VIF plugging event
9. Bind-time network-vif-plugged event (via RMQ)
10. Unpause instance...
11. Get LSPs
12. Get Port Binding Details
13. Populate Port Bindings, fill requested_chassis column
14. Get Port Binding Details
15. Lookup PF & VF representor netdev name w/devlink.
16. Plug VF representor & Program flows
17. Offload Flows
Upstreaming Status

- **OVN**
  - Plug providers infrastructure upstreamed (will appear in 21.12)
  - New `ovn-org/ovn-vif` project created under ovn-org

- **CMS**
  - OpenStack Nova and Neutron specifications approved for Yoga
    - Nova and Neutron specs are merged
    - Nova and Neutron Code is submitted for review
  - Kubernetes
    - TBD
Call for Devices

- If you have a device you would like to test this on:
  - Ping us via e-mail/IRC
  - Send us a sample (or two)
- More devices tested will help us make adjustments if needed
Demo
Thank you. Questions?