Open vSwitch DPDK Acceleration Using HW Classification
Rony Efraim
DPDK summit Dublin Oct 2016
ASAP\(^2\) take advantage of ConnectX-4 capability to accelerate or offload “in host” network stack

Three main use cases

**ASAP\(^2\) Direct**
- Full vSwitch offload

**ASAP\(^2\) Flex**
- vSwitch acceleration
- VNF/VM acceleration
Every switch (virtual or physical) has a notion of “packet processing pipeline”
- (Push/pop vlan, Tunnel Encap/decap operations, QoS related functionality: (Metering, Shaping, Marking, Scheduling), Switching action)

Typical ingress pipeline of a virtual switch can be:

ASAP²-Flex is a framework to offload part of the packet processing – one or more pipeline stages, onto the NIC HW engines
The “last” two actions in the pipeline, the switching decision and Tx operation are left to the SW based dataplane of the virtual switch (e.g. OVS datapath module or OVS-DPDK etc.)
This will allow VMs to use Paravirt interfaces as the actual switching decision is done in the SW and the virtual switch dataplane is NOT bypassed (just accelerated)
Each offloaded pipeline stage can result in one of the following:

- Packet format change (e.g. decapsulated packet)
- Some decision about the packet forwarding behavior, embedded in Metadata that will be passed on to the virtual switch dataplane in the SW
  - E.g. the Classification stage will result with a FLOW_ID that will be carried on with the packet to the SW dataplane

The SW based forwarding plane can leverage on the Offloading scheme:

- It can use the Metadata “hints” from the HW to accelerate its operation
  - E.g. classification via X-tuple (be it 5 or 12) in HW, notify SW dataplane on classification result
  - The SW dataplane can now classify on the FLOW_ID provided in the metadata instead of the more complex X-tuple classification
- If the HW decapsulation was used, the SW need not perform the actual decap action
- QoS can be enforced by the HW (shaping, rate limiting, packet scheduling to achieve bandwidth guarantee etc.)
ConnectX-4 Classification & Actions

- Classification based on
  - L2: S/D-MAC, Ethertype, VLAN’s
  - L3: IPv4/IPv6 s/d IP Protocol / Next header …
  - L4: S/D Port flags
  - Tunneling: vxlan VNI …
  - Inner packet L2/L3/L4
  - Different mask per flow

- Action
  - drop
  - Allow
  - flow id assignment
  - count
  - forward to ring
  - encap/decap tunnel
Current openVswitch over DPDK

- PMD receive the packets
- RSS to cores
- DP-IF classify the packets
- Action forward to VF
openVswitch using HW classification

- For every OVS flow DP-if should use the DPDK filter to classify with Action tag (report id) or drop.
- When receive use the tag id instead of classify the packet
- for Example:
  - OVS set action Y to flow X
    - Add a flow to tag with id 0x1234 for flow X
    - Config datapath to do action Y for mbuf->fdir.id = 0x1234
  - OVS action drop for flow Z
    - Use DPDK filter to drop and count flow Z
    - Use DPDK filter to get flow statistic
What missing for OVS in Current Flow filters

- All current flow filters are either “fixed” or “RAW”
  - E.g. the ntuple filter is limited.
  - E.g. the flex looks at X first packet bytes as a bytestream and compares (hence if there’s VLAN the Flow spec will be different then if there isn’t, even if the interesting field for classification is IP…)

- No filter support 12 tuple
- No counter per flow, required for droop.
New Generic flow interface - RTE_ETH_FILTER GENERIC

- **Requirements for a new API:**
  - Flexible and extensible without causing API/ABI problems for existing applications.
  - Should be unambiguous and easy to use.
  - Support existing filtering features and actions listed in Filter types.
  - Support packet alteration.
  - In case of overlapping filters, their priority should be well documented.
  - Support filter queries (for example to retrieve counters).
  - Support egress (TX) matching and specific actions.

- **Concept:**
  - Define Flow spec fields as a TLV
  - Define list of Actions for a matched Packet (as a TLV)
    - Flow_tag, Drop, count etc…

- **For more info:** [https://rawgit.com/6WIND/rte_flow/master/rte_flow.pdf](https://rawgit.com/6WIND/rte_flow/master/rte_flow.pdf)
More Complex Use Cases: Nested Virtual Switch Offload

- Multiple VMs, each running multiple containers
- Container connected via PV, VMs are connected with VF (SRIOV)
  - ASAP²-Direct (SRIOV)
    - for switching packets directly to the VMs
  - ASAP²-Flex (DPDK)
    - within each VM to accelerate the “inner” virtual switch
time for questions
Thank You