OVS Acceleration using Network Flow Processors

Johann Tönsing

2014-11-18
Agenda

• Background: OVS on Network Flow Processors
  • Network device types => features required => acceleration concerns

• OVS Acceleration Options
  • OVS (or OpenFlow) Agent Only
  • Offloading OVS with Fallback
    • Examples:
      • Userspace <=> accelerator
      • Userspace <=> kernel <=> accelerator
      • Userspace <=> (kernel OR accelerator)
      • Userspace <=> (kernel AND accelerator)

• Observations

• Evolution

• Conclusions
Summary: OVS on Network Flow Processors

OpenStack Neutron Plugin → OpenFlow Controller
Orchestration Application or Adapter → Carrier OSS/BSS

CPU (x86/ARM...) running Linux/...

OpenFlow

OVS: OpenFlow Agent

OVSDB
OF-Config

NFP

Actions
Load Bal
Meter / QoS
Hdr Mod
...

Client
Rx

Network Flow Processor Powered Networking Device

OpenFlow 1.x Dataplane

Classifier

OVS / OF Table

OVS

Server
Tx
Summary: OVS on Network Flow Processors

Network Device Types
- FlowNIC in Server
- Intelligent Top of Rack
- NFV / Middlebox Platform
- OpenFlow Switch / Gateway

Combinations of these
Also usable as hybrid device: SDN + L2/L3/NAT/crypto...

High performance / capacity intelligent devices
Improved efficiency
Summary: OVS on Network Flow Processors

Network Device Types
- FlowNIC in Server
- Intelligent Top of Rack
- NFV / Middlebox Platform
- OpenFlow Switch / Gateway

Combinations of these

Also usable as hybrid device: SDN + L2/L3/NAT/crypto...

High performance / capacity intelligent devices

Improved efficiency

Network Flow Processor Powered Networking Device
NFP: 200+ core / 200+G networking / 200G PCIe
(200G=2x100G full duplex: line rate=300Mpps, complex processing target = 60Mpps, 6Mfps)
Summary: OVS on Network Flow Processors

Network Device Types
- FlowNIC in Server
- Intelligent Top of Rack
- NFV / Middlebox Platform
- OpenFlow Switch / Gateway

Combinations of these
Also usable as hybrid device: SDN + L2/L3/NAT/crypto...

High performance / capacity intelligent devices
Improved efficiency

Network Flow Processor Powered Networking Device
NFP: 200+ core / 200+G networking / 200G PCIe
(200G=2x100G full duplex: line rate=300Mpps, complex processing target = 60Mpps, 6Mfps)
Summary: OVS on Network Flow Processors

Network Device Types
- FlowNIC in Server
- Intelligent Top of Rack
- NFV / Middlebox Platform
- OpenFlow Switch / Gateway

Combinations of these
Also usable as hybrid device: SDN + L2/L3/NAT/crypto...

High performance / capacity intelligent devices
Improved efficiency

CPU (x86/ARM...) running Linux/...

Orchestration
Application or Adapter
Carrier OSS/BSS

OpenFlow
Controller

OpenFlow
Agent

OVSDB
OF-Config

Actions
Load Bal / Meter / GoS
Hdr Mod...

 PCI DMA Driver

200Gbps

PCIe

Network Flow Processor Powered Networking Device
NFP: 200+ core / 200+G networking / 200G PCIe
(200G=2x100G full duplex: line rate=300Mpps,
complex processing target = 60Mpps, 6Mfps)

© 2014 Netronome Systems, Inc.
Summary: OVS on Network Flow Processors

Network Device Types
- FlowNIC in Server
- Intelligent Top of Rack
- NFV / Middlebox Platform
- OpenFlow Switch / Gateway

Combinations of these
Also usable as hybrid device: SDN + L2/L3/NAT/crypto...

High performance / capacity intelligent devices
Improved efficiency

Network Flow Processor Powered Networking Device
NFP: 200+ core / 200+G networking / 200G PCIe
(200G=2x100G full duplex: line rate=300Mpps, complex processing target = 60Mpps, 6Mfps)
Summary: OVS on Network Flow Processors

Network Device Types
- FlowNIC in Server
- Intelligent Top of Rack
- NFV / Middlebox Platform
- OpenFlow Switch / Gateway

Combinations of these
Also usable as hybrid device: SDN + L2/L3/NAT/crypto...

High performance / capacity intelligent devices
Improved efficiency

Network Flow Processor Powered Networking Device
NFP: 200+ core / 200+G networking / 200G PCIe
(200G=2x100G full duplex: line rate=300Mpps,
complex processing target = 60Mpps, 6Mfps)
Summary: OVS on Network Flow Processors

Network Device Types
- FlowNIC in Server
- Intelligent Top of Rack
- NFV / Middlebox Platform
- OpenFlow Switch / Gateway

Combinations of these
Also usable as hybrid device: SDN + L2/L3/NAT/crypto...

High performance / capacity intelligent devices

Improved efficiency

Network Flow Processor Powered Networking Device
NFP: 200+ core / 200+G networking / 200G PCIe
(200G=2x100G full duplex: line rate=300Mpps, complex processing target = 60Mpps, 6Mfps)
Summary: OVS on Network Flow Processors

Network Device Types
- FlowNIC in Server
- Intelligent Top of Rack
- NFV / Middlebox Platform
- OpenFlow Switch / Gateway

Combinations of these

Also usable as hybrid device: SDN + L2/L3/NAT/crypto...

High performance / capacity intelligent devices

Improved efficiency
**Summary: OVS on Network Flow Processors**

**Network Device Types**
- FlowNIC in Server
- Intelligent Top of Rack
- NFV / Middlebox Platform
- OpenFlow Switch / Gateway

**Combinations of these**
- Also usable as hybrid device: SDN + L2/L3/NAT/crypto...

**High performance / capacity intelligent devices**
- Improved efficiency

---

**Network Flow Processor Powered Networking Device**

NFP: 200+ core / 200+G networking / 200G PCIe
(200G=2x100G full duplex: line rate=300Mpps, complex processing target = 60Mpps, 6Mfps)
Summary: OVS on Network Flow Processors

**Network Device Types**
- FlowNIC in Server
- Intelligent Top of Rack
- NFV / Middlebox Platform
- OpenFlow Switch / Gateway

**Combinations of these**
Also usable as hybrid device: SDN + L2/L3/NAT/crypt...

**High performance / capacity intelligent devices**

**Improved efficiency**

**NFP:** 200+ core / 200+G networking / 200G PCIe
(200G=2x100G full duplex: line rate=300Mpps, complex processing target = 60Mpps, 6Mfps)

OpenStack Neutron Plugin
Orchestration Application or Adapter
Carrier OSS/BSS

CPU (x86/ARM…) running Linux/…

Any app on server
Middlebox apps: IDS / IPS / Firewall
App ID + QoS
Recording
Network Antivirus
Antispam
Protocol Modules: OSPF / BGP
ARP / ICMP
SIP / RTP
etc
OpenFlow Controller and/or other control / management software

**Network Flow Processor Powered Networking Device**

**Extensions**
Summary: OVS on Network Flow Processors

Network Device Types
- FlowNIC in Server
- Intelligent Top of Rack
- NFV / Middlebox Platform
- OpenFlow Switch / Gateway

Combinations of these
Also usable as hybrid device: SDN + L2/L3/NAT/crypto...

High performance / capacity intelligent devices
Improved efficiency

Network Flow Processor Powered Networking Device
NFP: 200+ core / 200+G networking / 200G PCIe
(200G=2x100G full duplex: line rate=300Mpps,
complex processing target = 60Mpps, 6Mfps)
Summary: OVS on Network Flow Processors

Network Device Types
- FlowNIC in Server
- Intelligent Top of Rack
- NFV / Middlebox Platform
- OpenFlow Switch / Gateway

Combinations of these
Also usable as hybrid device: SDN + L2/L3/NAT/crypto...

High performance / capacity intelligent devices
Improved efficiency

Acceleration approaches: offload kernel vs userspace

Network Flow Processor Powered Networking Device
NFP: 200+ core / 200+G networking / 200G PCIe
(200G=2x100G full duplex: line rate=300Mpps,
complex processing target = 60Mpps, 6Mfps)
OVS Acceleration Approaches: OVS Agent vs OVS Offload/Fallback

“OVS = Agent, NFP = Switch”

• Fallback reasons: table lookup miss (table overflow vs. not full but no entry matched), unsupported classification (e.g. unsupported protocol), unsupported action, and/or deliberate hand-off to agent (e.g. packet in to controller)
• Most appropriate approach depends on features / perf. needed (related to type e.g. NIC in server, intelligent TOR, mbox / gateway)
• In part depends on performance required vs. achievable with attached x86 (or other CPU)

All variations support packet delivery from NFP directly to (host / guest) x (kernel / user mode), tunnel handling in NFP...

© 2014 Netronome Systems, Inc.
“OVS Agent Only” Features

- Dataplane feature set “limited” to what switch/NIC hardware (NFP) supports
  - Could exceed what OVS dataplane offers (e.g. set individual header fields, not just entire L2/L3 header like OVS kernel datapath)
  - Netronome’s implementation actually fairly complete (with high performance):
    - OpenFlow 1.3+ with 250 tables, Ethernet / VLAN / MPLS / IPv4+IPv6 / TCP / UDP... matching + actions (set/push/pop/dec TTL), metering, QoS, logical port style tunnel termination/origination (VXLAN / [NV-]GRE / ...)
    - Extensions: fragmentation, ICMP stack, IP stack (route lookup / ARP processing) for tunnels, load balancing, improved QoS...

- Approach can support (host or guest) x (user or kernel mode) data delivery (incl. SR-IOV)
  - Apps (kernel netdev / user mode driver) modeled as virtual ports - after app transited, packets re-injected into pipeline
  - Apps (user mode driver) can trigger egress fastpath - after app transited, packet sent directly to egress port
  - Load balancing to application instances (static + stateless vs. dynamic + stateful)

- Each physical switch/NIC port represented by a VNIC instance in x86 (typically host kernel netdev)
  - Exposes physical port statistics, link state -- Linux commands e.g. ethtool / ip / ifconfig just work
  - In-band control: my MAC traffic diverted, broadcast / multicast copied (without needing OpenFlow table entries)
  - Permits sniffing traffic (tcpdump) - detects promiscuous mode on/off (could offload BPF too)

- Flow tracking table
  - Manual population / modification via API (n M mods/s) or OpenFlow (mods/s are limited by controller / agent)
  - Auto learning - n M microflows/s - optionally using API to update policy / obtain statistics
Offloading with Fallback: Offload OVS Kernel vs. Userland

- Fallback to OVS x86 code to support features not in NFP (yet) or entries not present in NFP tables (e.g. DRAM-less NIC)
- OVS kernel offloaded to NFP, fallback to OVS kernel (and fallback onward to OVS userland as usual)
  - Feature set typically reflects (but can sometimes exceed) OVS kernel datapath capabilities
  - Could be regular OVS userland - but also different userland (BigSwitch, Midokura, PLUMgrid...)
- OVS userland offloaded to NFP, fallback to OVS userland (via zero copy driver e.g. DPDK)
  - OVS kernel datapath not involved / required (no impact on features + performance, fewer issues with upstreaming to kernel.org)
  - Either way a VNIC instance (e.g. kernel netdev) per physical port is useful - statistics, link state, sniffing, in-band control (OpenFlow / SSH ...)

All variations support packet delivery from NFP directly to (host / guest) x (kernel / user mode), tunnel handling in NFP...
• NFP sends fastpathed (non-fallback) packets directly to (host or guest) x (kernel or userspace) - e.g. to SR-IOV VF VNIC3
• Fallback traffic sent to host (kernel / userspace): presented to OVS via e.g. VNIC OFP1' --- say OVS outputs to VNIC3'
• Reflector forwards from VNIC3' to VNIC3 (no need to re-process in pipeline as already processed by fallback code)
Offloading with Fallback: Combos

• Hybrid OVS userland / kernel offload + fallback
  • Easy: multiple switch instances: some userland, others kernel - each physical port attached to one of these
  • Harder: within a switch instance, some traffic sent to kernel, other traffic to userland
    • Based on what?
      • Determination whether userspace processing will be needed - if so, skip kernel?
        • Does not make sense if userspace fallback is minority of traffic / userspace is slower than kernel
      • Default to userspace, change to kernel when kernel processing is needed (for e.g. conntrack, IPsec, etc.)
        • Useful if userspace is faster than kernel
  • Is determination accurate?
    • Hopefully easy and accurate for entire microflow if based on traffic type (stateless)
    • Handing over or starting mid-flow can causes issues (entry state missing in tables, statistics wrong etc.)
    • Not urgent to implement on high functionality / capacity / throughput platform like NFP as most traffic is handled by it..

• Hybrid SDN / traditional networking, hybrid OpenFlow and non-OpenFlow features
  • Traditional forwarding = L2/L3
    • Again, easier if ports attached to distinct traditional or switch instances (internal ports to link them)
    • Otherwise need one mechanism (SDN/traditional) to be primary, other secondary, or need third classifier as tie-breaker
  • Non-OpenFlow features like tunnel handling (e.g. VXLAN / IPsec) / firewalling / NAT / other “black box” features
    • Invocation of features via built in behavior (e.g. my MAC / traffic type) vs. explicitly via an OpenFlow action / logical port etc.
    • Some need IP stack (ARP, route lookup, frag tracking, fragment / defragment) or other kernel functionality

• Examples as implemented on NFP
  • Logical port style VXLAN / GRE termination handled via OVS kernel style mechanisms (lookup tunnel header + inner header together in tables)
  • IP stack (route lookup to obtain egress port + source IP) and ARP processing for tunnel origination, leveraging Linux stack (entries are cached on NFP)
  • IKE for IPsec, leveraging Linux usermode code
  • Fragment tracking (without reassembling fragments) + peeking into tunnels (without terminating them) + flow tracking -- implemented “before” OVS / OpenFlow
Observations

- NFPs support all these acceleration approaches (implemented agent, usermode offload, kernel offload)
  - Certain approaches may be more or less suited to more limited acceleration hardware

- Offload with fallback degrades gracefully (features e.g. classification / actions, also capacity)
  - Fallback to kernel vs to usermode?
    - Where are the existing standard features (kernel e.g. conntrack, vs. usermode e.g. richer OpenFlow)?
    - Which parts of the code do OVS community customize most often / readily?
    - Which is most performant (e.g. DPDK or other userspace driver avoiding some kernel overheads vs. cache in kernel)?

- Offloading OVS kernel does not necessarily yield the highest performance - examples of issues:
  - OVS kernel datapath functionality limited, e.g. broad brush actions: decrement TTL => replace L3 header
  - Number of entries in OVS kernel tables could explode or experience churn (even with megaflow changes)

- Offloading OVS userspace simplifies supporting acceleration hardware with varying intelligence / capacity
  - In kernel / at netlink (DPIF), entries have been “compiled” to low level => easier when seeing higher level intent in user mode

- Deciding where to perform processing (userspace / kernel / accelerator, OVS specific code / standard Linux kernel code etc.) could be complex
  - NFP’s OVS kernel offload is fully featured, has large table capacity etc. => easy to offload all kernel table entries with small kernel patch
  - For more limited devices, decision making may be more complex - functionality split => table entry positioning - best handled in userspace
  - Can still decide in userspace to offload directly to accelerator, vs. via kernel to accelerator
Evolution of Dataplane Flexibility

Offload / Acceleration (drop in replacement)
- x86 User Mode
- x86 Kernel
- NFP
- Data Path
- Match
- Act

x86 Apps Calling APIs
- x86 User Mode
- x86 Kernel
- NFP
- Data Path
- Match
- Act

Datapath Extensions in NFP
- Native Code (e.g. C)
- NFP
- Data Path
- Match
- Act

Flexible Datapath Abstraction
- OpenFlow 2.x, P4, PIF, eBPF...
- NFP
- Data Path
- Match
- Parse
- State
- Act
- App

- OVS/OpenFlow Match Action
- Linux IPsec
- Linux Firewall
- Linux L2/L3 Fwd
- RDMA
- ...

- Flow API
- Load Balancing API
- Crypto API
- Forwarding API
- RDMA API
- ...

- Custom tunnel
- Custom action
- Custom matching
- ...

- Protocol agnostic flexible parsing
- Arbitrary arrangement of matching tables
- Matching without tables
- State storage / retrieval
- Complex actions
- Event handling

© 2014 Netronome Systems, Inc.
Conclusions + Next Steps

• Performance examples (details depend on traffic patterns, number and type of flow entries / actions etc.)
  • Using “20G” NFP, measured >20x speedup vs. unaccelerated OVS (IP + MPLS forwarding use case, multi table)
    • Difference can be larger for more complex actions / tunnels / traffic patterns - e.g. VXLAN, IPsec, high flow setup rates
    • Observed flow tracker performing flow learning (5-tuple) at 12Mfps
  • Expect further improvement (~3x-10x) for “200+G” NFP
  • Capacities: millions or tens of millions of flow entries, 100,000s of tunnels (requirements vary per device type)
  • => Order(s) of magnitude improvement achievable using acceleration

• Questions for OVS community
  • Where to focus going forward - implement features in kernel (for which OSes) vs. implement in userspace (cross OS / lightweight) vs. both
  • Considerations: TTM, software only performance, ease of acceleration, leveraging existing code / developer skill sets, ease of maintenance

• Questions for Linux and other OS communities
  • Leverage OVS vs implement different mechanisms e.g. tunnel termination/origination, QoS, eBPF / PIF...

• Questions for all
  • How best to support acceleration hardware, without duplicating efforts
  • (Excluded due to lack of time: specific API proposals for table manipulation APIs, acceleration “objects” and APIs: Linux/OVS/other...)