DPDK

DATA PLANE DEVELOPMENT KIT

Next Gen Virtual Switch

CloudNetEngine
Founder & CTO
Jun Xiao

Agenda

- Thoughts on next generation virtual switch
- Technical deep dive on CloudNetEngine virtual switch
- Q & A

Major vSwitches categorized by workloads

Enterprise type workloads		NFV type workloads
vSphere	KVM	KVM
Standard vSwitch	Linux Bridge	OVS-DPDK
vDS	OVS	
Cisco Nexus 1000v	Cisco Nexus 1000v	6WIND vSwitch
Cisco AVS		
HP FlexFabric 5900v		

Contexts – what's happened?

- SDN
  - Normally deal with "enterprise" workloads
  - Mainly focus on decoupling control & forwarding planes and control plane scalability
  - Virtual switch CPU efficiency is overlooked

- NFV
  - Mainly focus on "hairpin" network traffic performance
  - Virtual switch feature set is less considered
  - Exclusively using pre-allocated resources

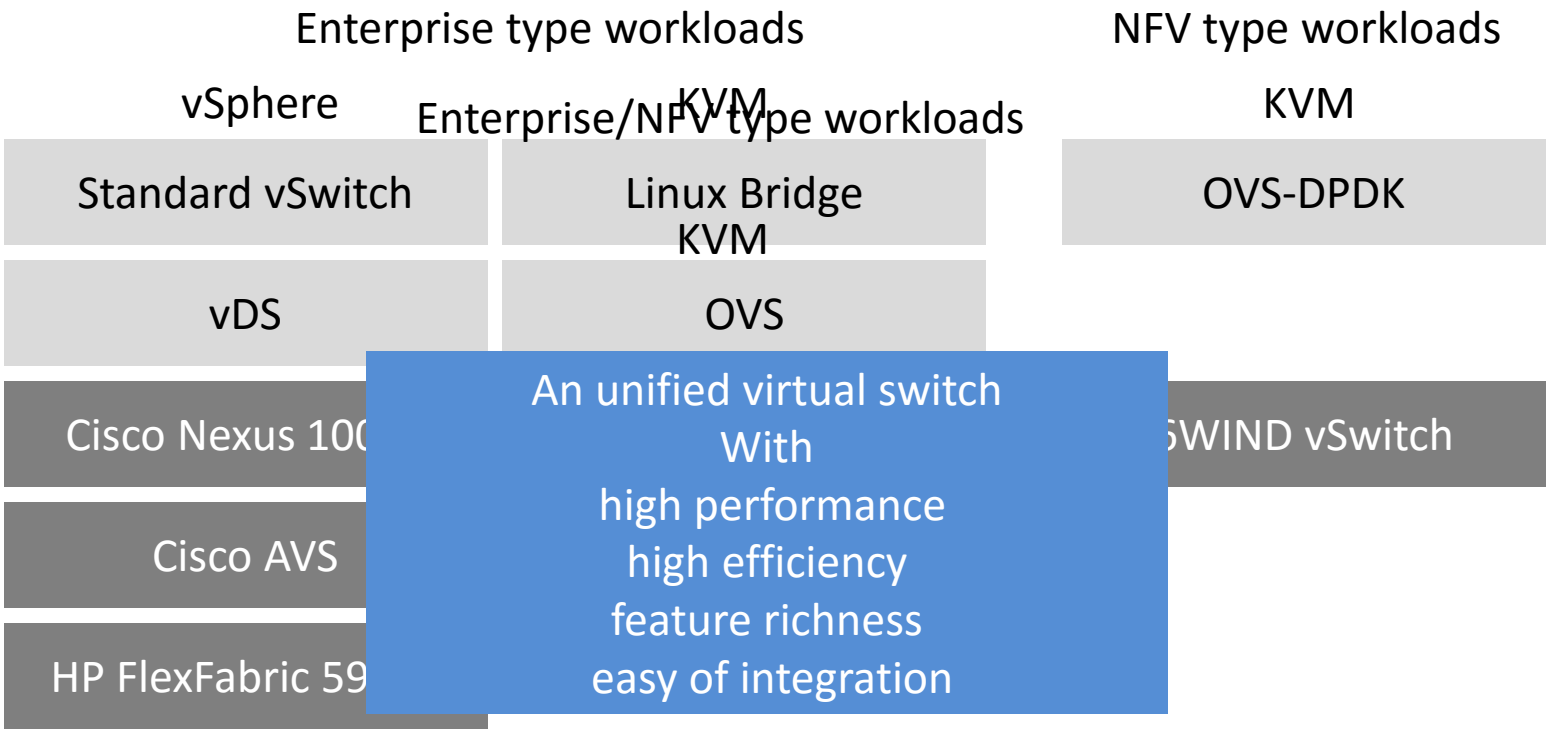
Contexts – Everything is application driven

- Client Server Era -> Mobile Cloud Era
  - Social, Mobile, Cloud, Big data
- Proliferated east-west network traffic in clouds
- Networking intelligences are shifting from core network to network edge
- Application agilities require network infrastructure to transform to a software-defined model

Key requirements on next generation virtual switch

- Performance
  - Support variety of virtualized workloads
- CPU efficiency (part of performance, but highlight it here)
  - Improve ROI for massive cloud investments
- Feature richness
  - For both public and private cloud use cases
- Easy of integration
  - Live in the ecosystem

CloudNetEngine's thoughts

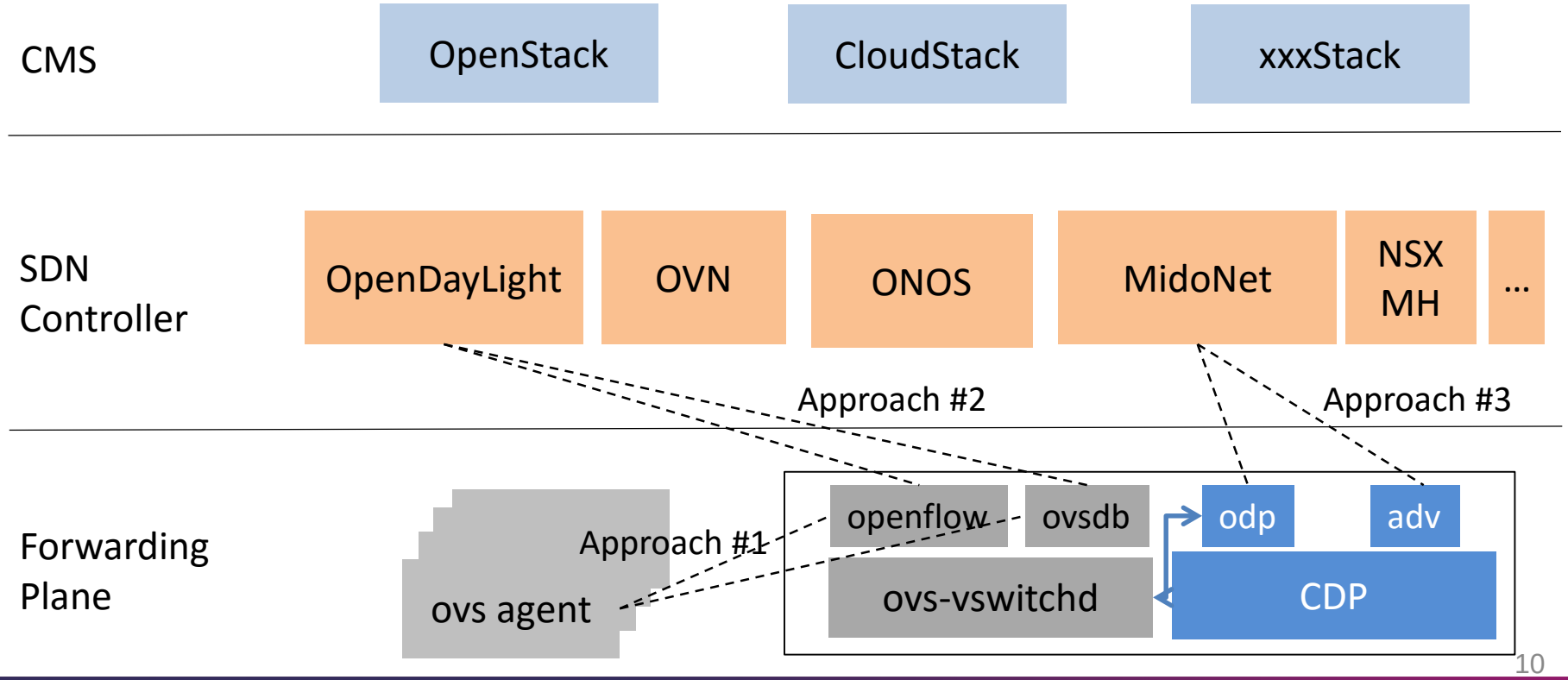


Technical deep dive on CloudNetEngine virtual switch

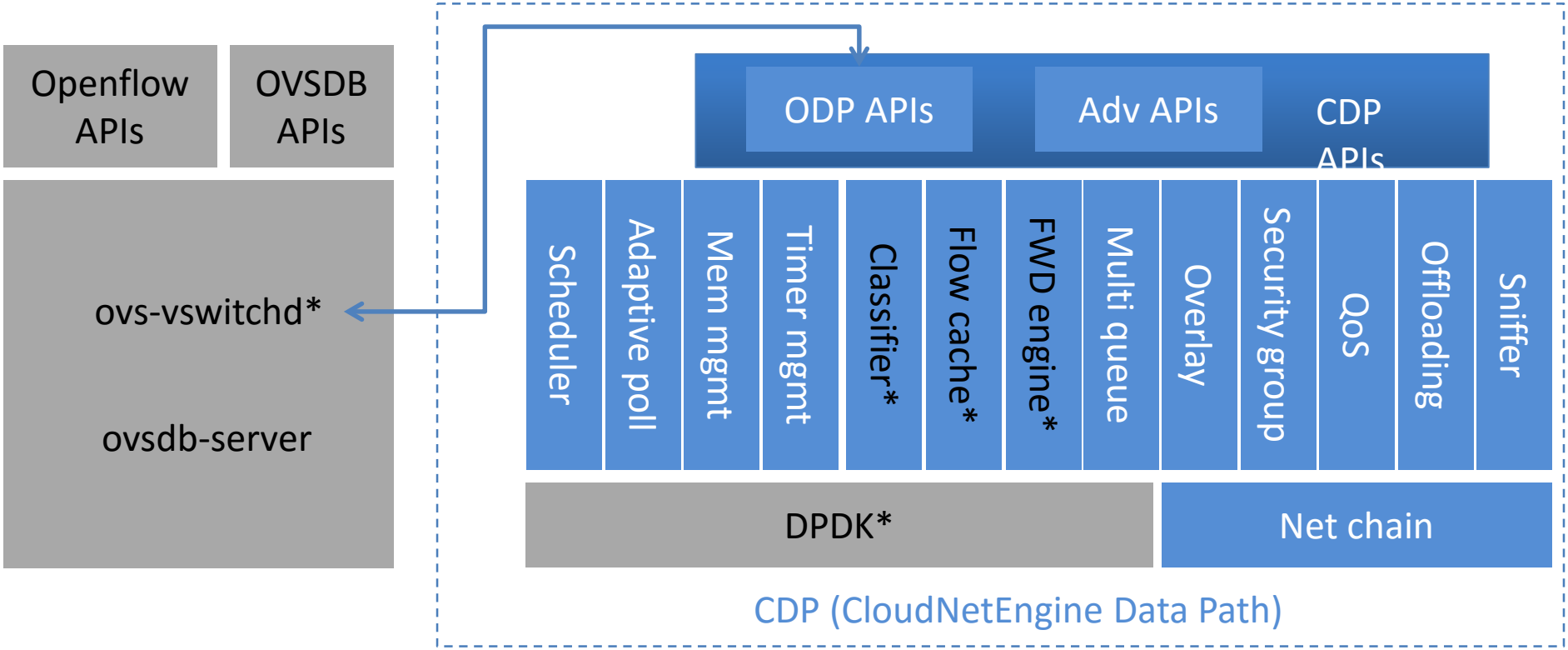
To build a great virtual switch,

we have to think out of box and break the silos.

Easy of integration – Live in the ecosystem



CloudNetEngine virtual switch architecture



Performance – rte_mbuf

- Put together fields which need reset in one cacheline, so reset can be vectorized
- Do bulk mbufs free when they are coming from the same mempool
- Make mbuf processing more robust (mbuf error is hard to fix)
  - Make a mbuf (or a mbuf chain) always integrity (pkt_len, data_len, next, nb_seg fields are quite relevant)
  - Do mbuf sanity check wherever possible and inject random allocation fault to catch errors

Performance – Instructions per cycle

- Parallelly processing packets as possible as we can in a reasonable batch
  - It’s hard if there are branches for the parallelled logic
- For virtio vhost user backend rx/tx, partial logic can be parallelled as all valid packets use at least one device descriptor
- For flow extraction, it’s hard as packets might have different format from very beginning of the extraction, i.e. L2 header

Performance – Load balance

- Place a new RX queue to the least loaded thread
- Migrate a heavy load RX queue to the least loaded thread
- “load” is calculated from most recent samplings on CPU usage, for thread load, some other pseudo metrics should also be factored in

CPU efficiency – Adaptive polling

- Busy polling is very bad for light load
- Userspace interrupt + polling is not good enough
- An adaptive interrupt + polling is needed
  - CPU usage doesn’t need to be perfectly linear to network load
  - Performance for heavy load should be on par with pure polling

CPU efficiency – Packet Group metadata

- An unified architecture to support a variety of workloads
  - “NFV” workload means less features
  - “enterprise” workload means rich features
- Packet Group Metadata
  - Associated with a vector of packets
  - Virtual switch checks packet group metadata before entering a feature logic, thus per packet check is avoided

CPU efficiency – Save packet copy

- In normal case
  - VM <-> VM needs two packet copies
  - VM -> PNIC needs one packet copy
- Under TX zero copy
  - VM <-> VM needs only one packet copy
  - VM -> PNIC don’t need packet copy
- Simple? But you have do it much more sophisticated
  - Needs a packet “done” notification mechanism to update virtio vrings
  - Have to copy reasonable length of headers to prevent guest attacks

Feature richness – Extensibility

- A lightweight and efficient framework (net chain) to plugin new features
  - It's RCU protected so that updating a net chain won't have any performance penalty on the datapath
  - A net chain can use packet group metadata to very quickly decide whether the net chain is applicable to the input packet vector or not

Feature richness – Security Group

- BPF with JIT is great to match on arbitrary packet fields
  - NPF of NetBSD is great, and it's very similar as Linux nftables
- Comparing to OVS conntrack based security group
  - OVS conntrack based security group leverages OVS recirculation mechanism which means flow tables must be lookuped again after a recirculation injection

Feature richness – Overlay

- When tunnels encap/decap are completely done in userspace, do we have to do routing/neighbor control completely in userspace?
  - Linux kernel routing table (thus routing protocols) can be fully leveraged because each userspace TEP(tunnel endpoint) has also a corresponding kernel tap device
  - Neighbor management still needs some userspace solicited ARP requests because kernel ARP table entries can be timeouted

Feature richness – Overlay tunnel acceleration

- In a typical cloud environment, there are a lot of tunnels setup between L2 neighbors
- All existing tunnel protocols carry over non-trivial useless headers for such “neighbor tunnels”
- A tunnel acceleration method can be used
  - Adaptive MAC encapsulation when acceleration prerequisites are met
  - All “tunnel metadata” are still preserved thus transparent to up layer SDN solutions
  - E.g. in standard VXLAN case, 64 bytes workload can only have 56% effective utilization rate, while the acceleration mechanism can increase to 74.4%

Feature richness – Offloading

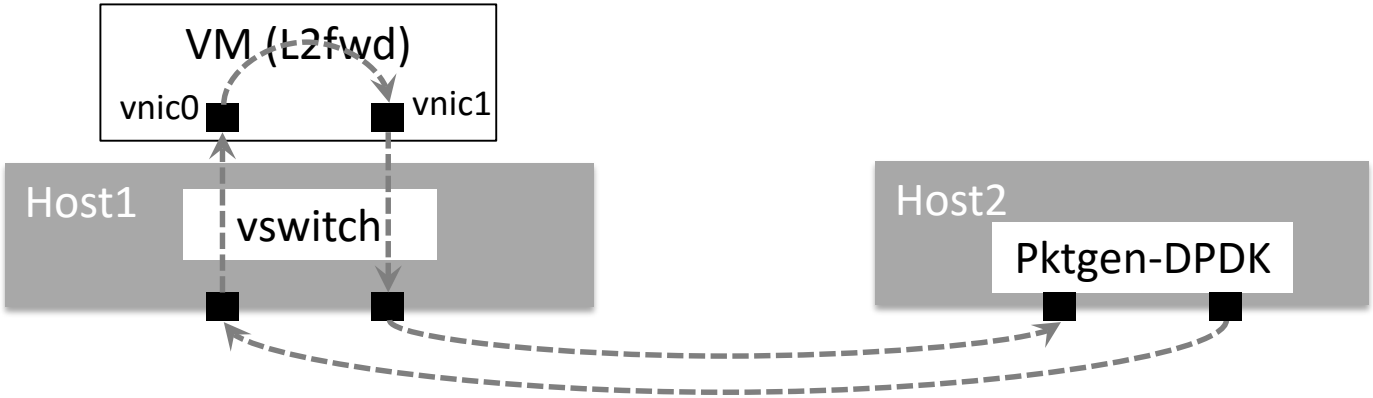
- There are quite a few offloading use cases in a virtual switch
  - VM <-> VM: native S/W offloading
  - VM -> PNIC: native H/W offloading
  - VM -> PNIC(w/ overlay offload): overlay H/W offloading
  - VM -> PNIC(w/o overlay offload): overlay S/W offloading

Feature richness – Sniffer

- What tool is used most often for network troubleshooting?
  - tcpdump
- How to get it done?
  - Based on BPF JIT and pcap-filter syntax can be fully supported
  - libpcap extention for userspace virtual switch port capture
  - tcpdump frontend don’t need change at all

Performance comparisons

NFV workload test configuration



CPU:
- Xeon E5-2620 v3 2.40GHz
- 6 physical cores, 12 logical cores

NIC:
- 82599ES 10-Gigabit

MEM:
- 16G

Host OS:
- Ubuntu 15.04 x86_64 + KVM
- Qemu 2.2.0

Guest OS:
- 2 vCPUs/ 2vNICs/ 4G memory
- Ubuntu 15.04 x86_64
- L2fwd

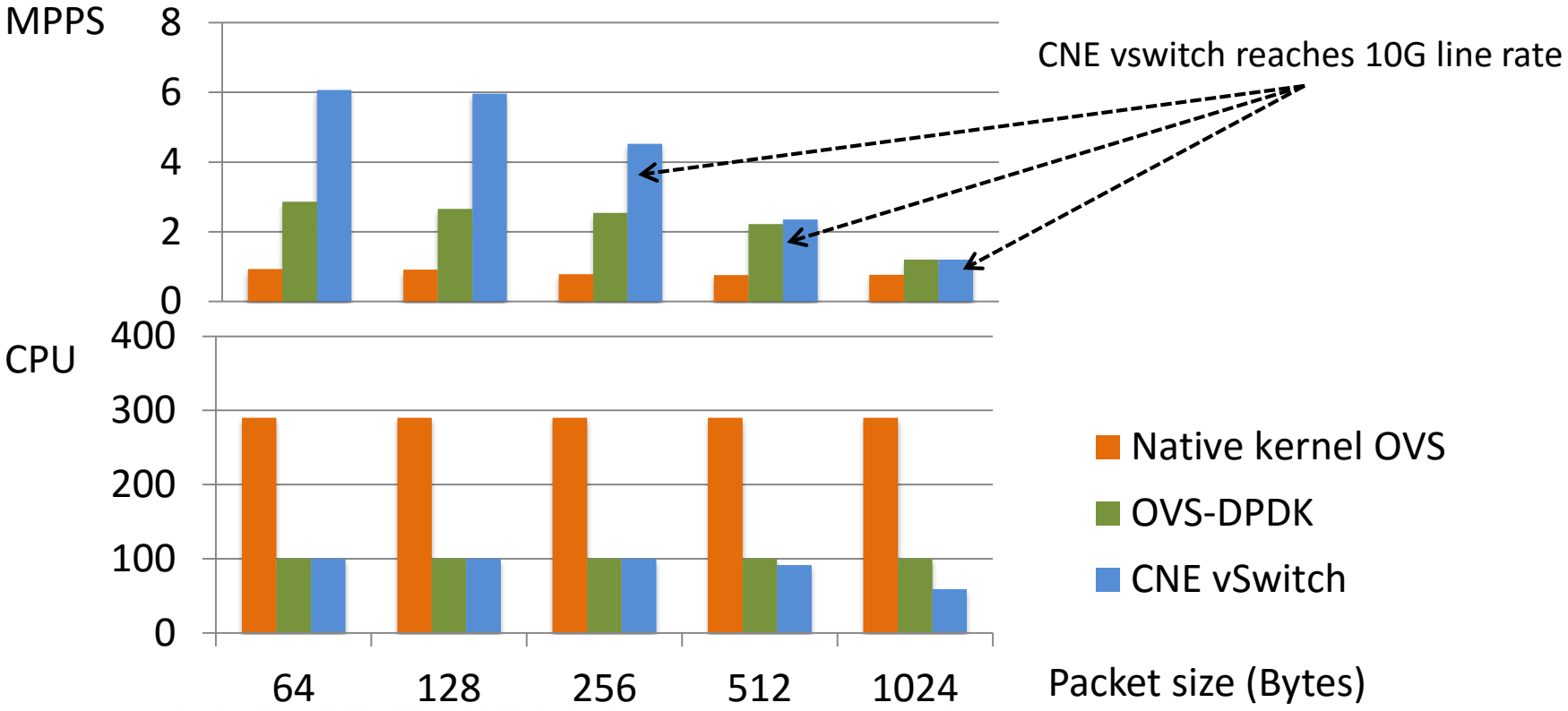
Pkt Generator:
- Pktgen-DPDK

Native kernel OVS:
- OVS 2.4
- OS bundled kernel module

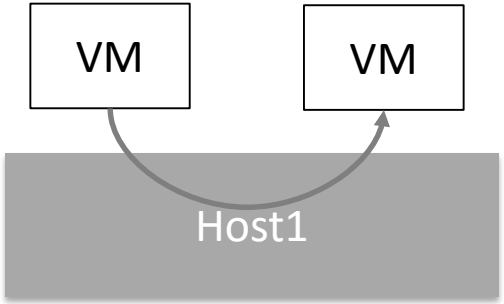
OVS-DPDK:
- OVS 2.4
- DPDK v2.0.0

CNE vswitch:
- Technical preview update 4

NFV workload performance comparisons



Enterprise workload test configuration



CPU:
- Xeon E5-2620 v3 2.40GHz
- 6 physical cores, 12 logical cores

NIC:
- 82599ES 10-Gigabit

MEM:
- 16G

Host OS:
- Ubuntu 15.04 x86_64 + KVM
- Qemu 2.2.0

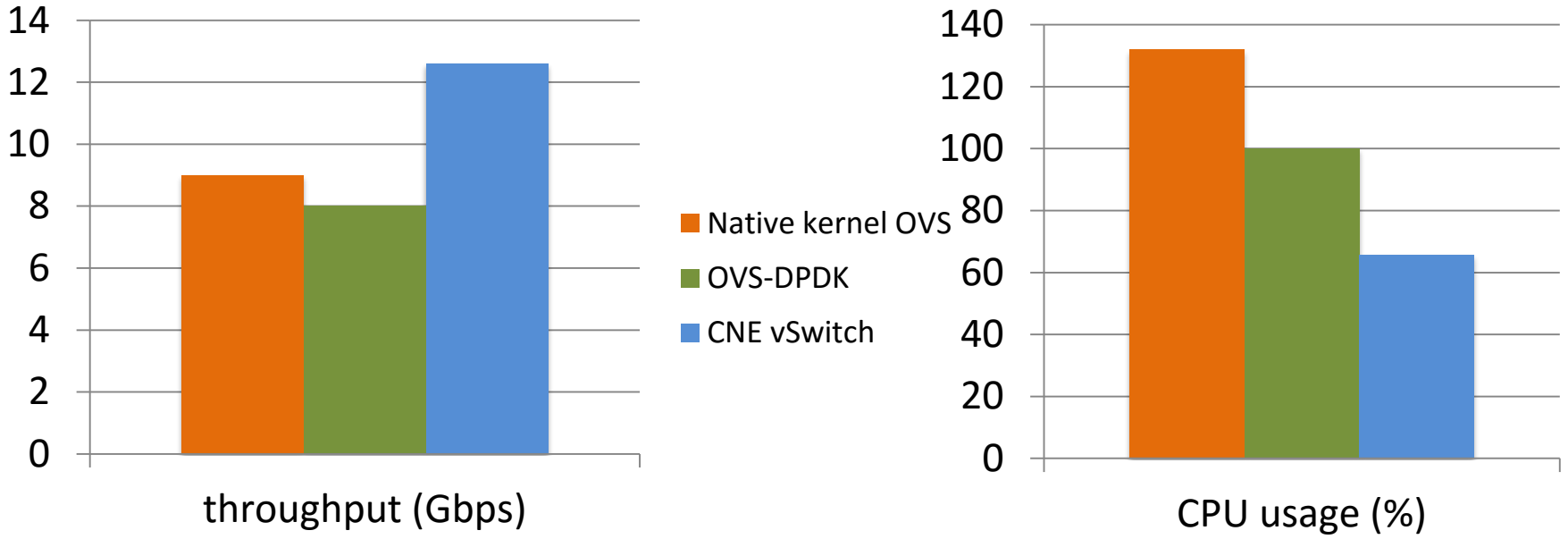
Guest OS:
- Ubuntu 15.04 x86_64
- iperf 2.05

Native kernel OVS:
- OVS 2.4
- OS bundled kernel module

OVS-DPDK:
- OVS 2.4
- DPDK v2.0.0

CNE vswitch:
- Technical preview update 4

Enterprise workload performance comparisons



Q & A

www.cloudnetengine.com

info@cloudnetengine.com

Twitter: @cloudnetengine


DPDK
DATA PLANE DEVELOPMENT KIT