Open vSwitch

Kazuki Hyoudou

Fujitsu Laboratories Ltd.

NUMA-aware DPDK-OVS for High Performance NFV Platform

What is NFV?

- Network Appliances
  - Proprietary Hardware
  - Vendor specific API/CLI
  - High CAPEX & OPEX
  - Need long term to update
- Network Functions Virtualization (NFV)
  - NW functions by software
  - Standard Hardware

Refer to http://portal.etsi.org/NFV/NFV_White_Paper2.pdf

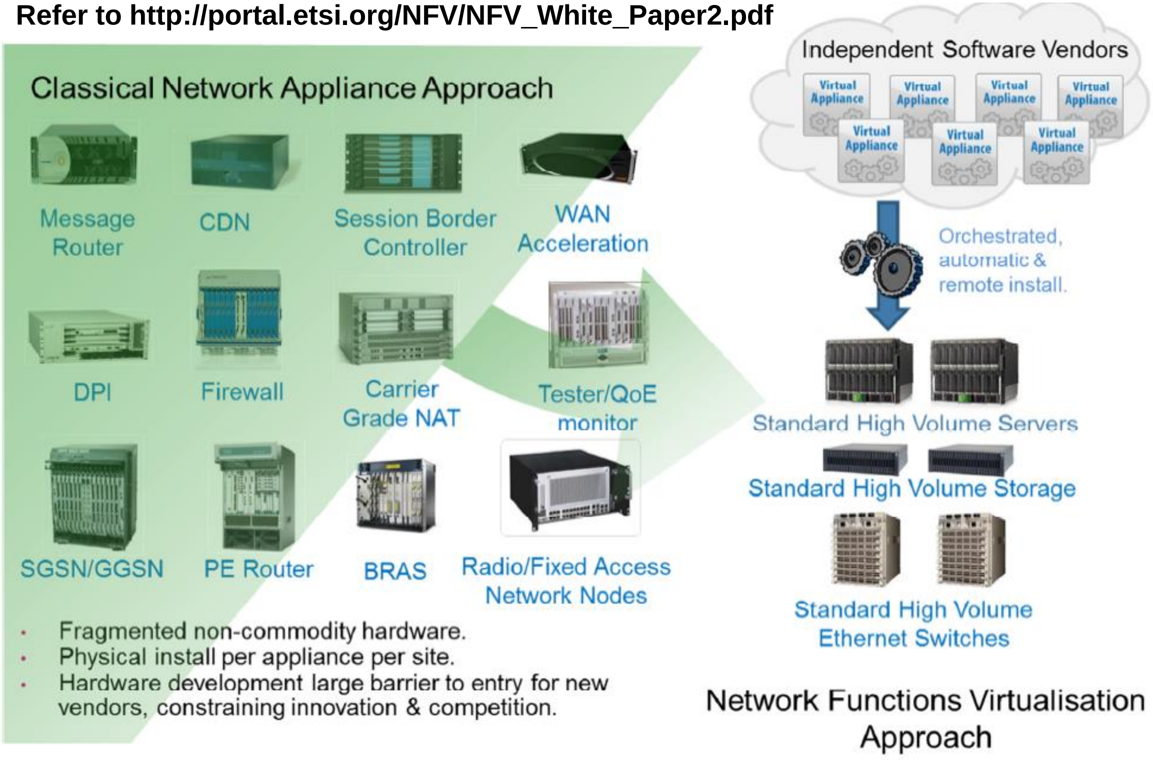
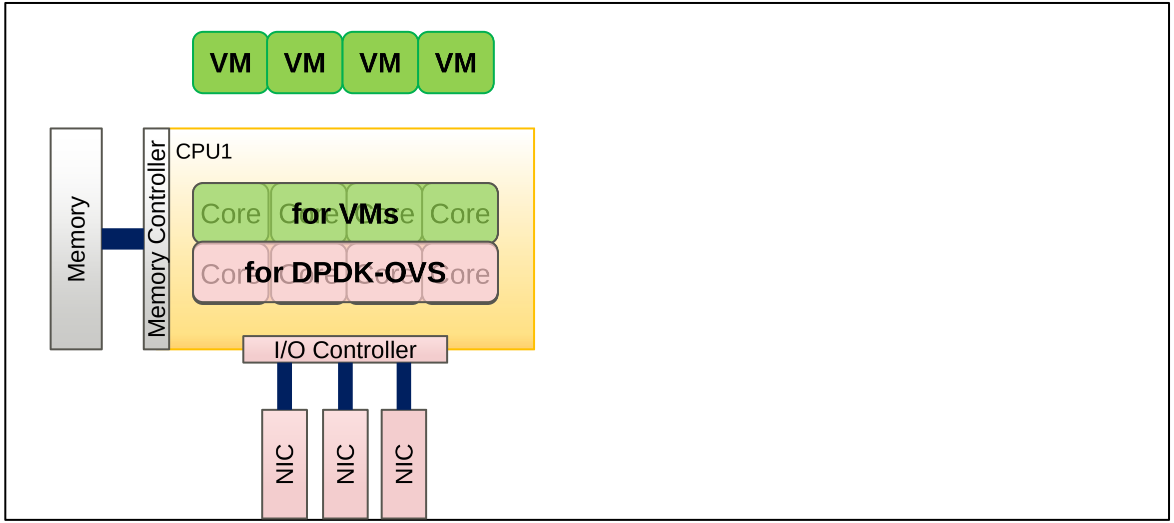


Figure 1: Vision for Network Functions Virtualisation

Important aspects for NFV Platform

- Use of commodity (non-proprietary) hardware
- Standard I/F for VNF such as virtio
- I/O throughputs
- The number of I/O ports (Eth I/F)
- The number of VNFs in a BOX
- etc.

To increase VNFs and I/O per BOX …

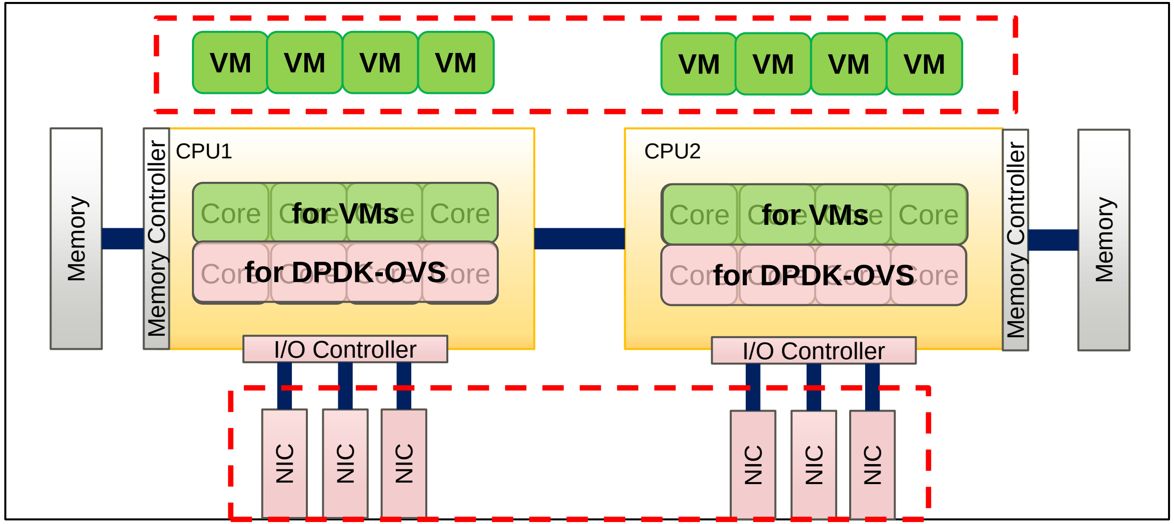


IA server (BOX)

- Assumptions
  - a CPU has 8 cores and 3 I/O slots
  - use 2 cores for physical port and more 2 cores for vhost-user
  - a VM occupies one core
- A BOX supports
  - 4 VMs (VNFs)
  - 3 NICs

To increase VNFs and I/O per BOX …

- One of the easiest way is to use multi-socket server, But …

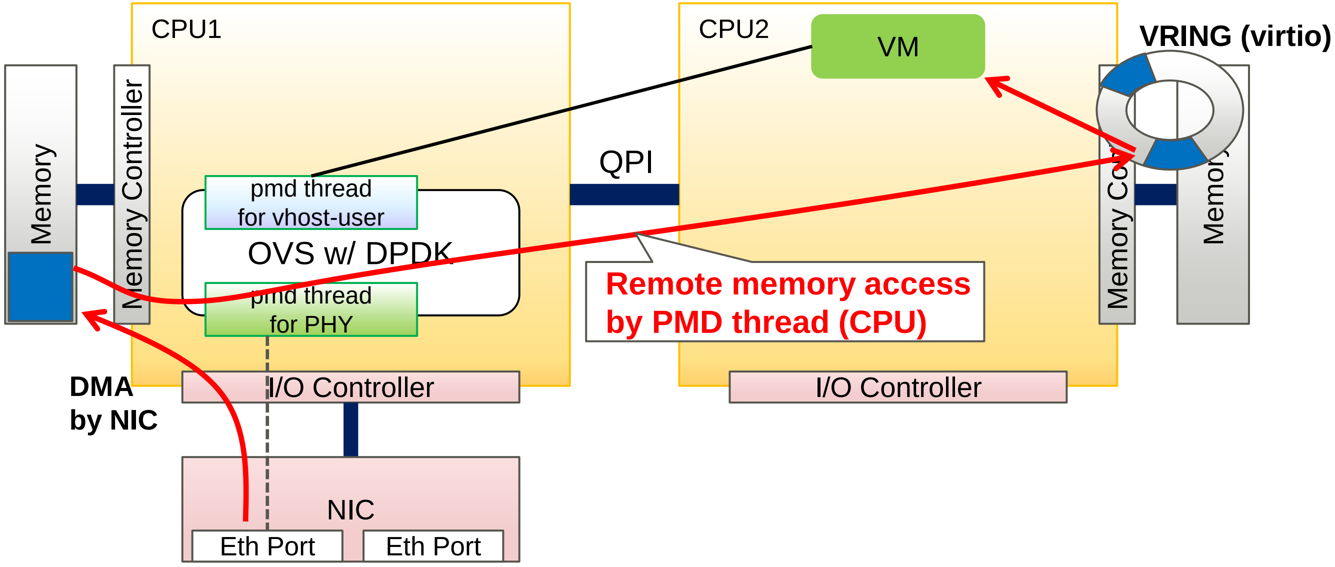


IA server (BOX)

- Assumptions
  - a CPU has 8 cores and 3 I/O slots
  - use 2 cores for physical port and more 2 cores for vhost-user
  - a VM occupies one core
- A BOX supports
  - 4 VMs → 8 VMs
  - 3 NICs → 6 NICs

Performance Issue of OVS on NUMA system

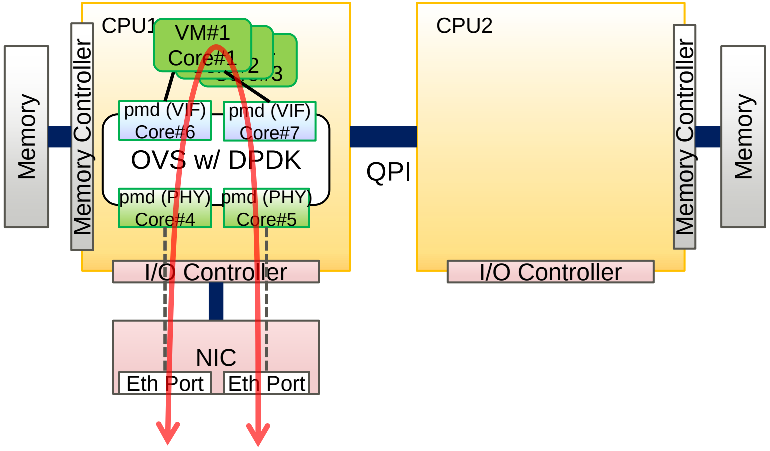
- If the VM is running on a NUMA node different from the node to which NIC is connected, its throughput decreases significantly.



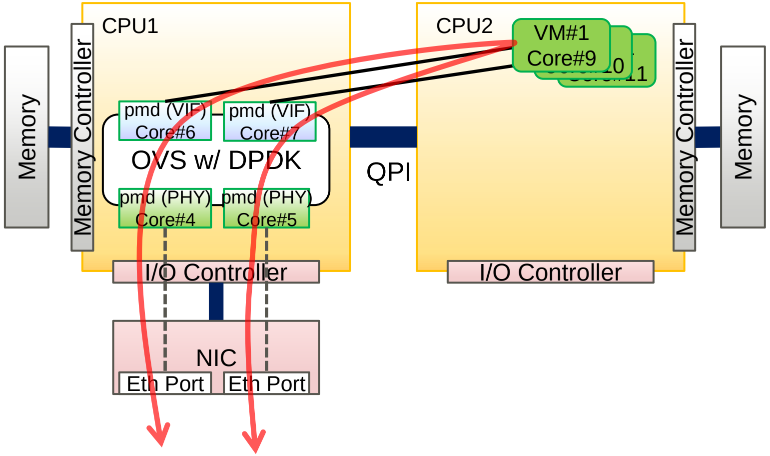
Pre-test to Study the Issue

- Measure the total throughput of 3 VNFs (L3 forwarder w/o DPDK) with 2 cases as follows;

Case1: All components are in the same NUMA node (no flows go through QPI)



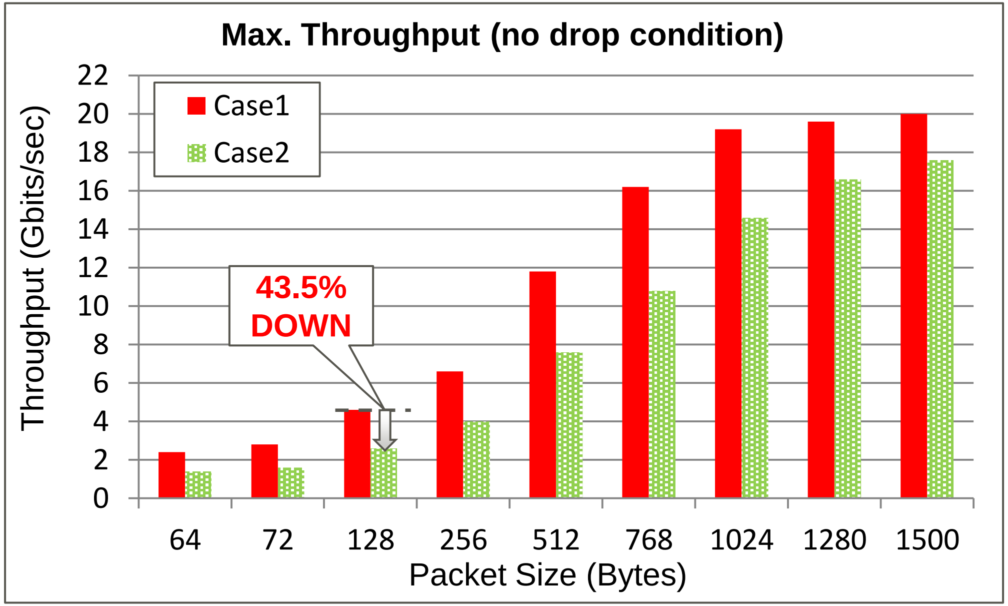
Case2: VMs are running on different node (flows go through QPI by CPU)



Result of Pre-test

- Throughput goes down 30 to 40%!
  (Max. decreasing is 43.5% in this test)

- CPU: Xeon® E5-2667 v3 @3.20GHz, 8 Cores x2
- Memory: 128 GB (DDR4-2133 16G x4 x2)
- NIC: Intel® X710-DA2 (PCIe Gen3 x8)
- OS: CentOS 7.1
- OVS: v.2.4.1 w/ DPDK v.2.0.0



	% of Case1	
pkt size	Case1	Case2
64	100.0	58.3
72	100.0	57.1
128	100.0	56.5
256	100.0	60.6
512	100.0	64.4
768	100.0	66.7
1024	100.0	76.0
1280	100.0	84.7
1500	100.0	88.0

Analysis of the Issue

- Consider the cause of the issue
  - QPI link has enough bandwidth (307 Gbps bi-directional bandwidth)
  - Remote memory access latency for copying data and lock operations are High!
    - L3 cache: 10ns, Local memory: 70ns, Remote memory: 120ns
- Measure performance counters concerning L3 cache miss event

Event Name	Case1	Case2
MEM_LOAD_UOPS_RETIRED.L3_MISS	552,945	6,708,912
MEM_LOAD_UOPS_L3_MISS_RETIRED.LOCAL_DRAM	554,079	644,671
MEM_LOAD_UOPS_L3_MISS_RETIRED.REMOTE_DRAM	11	171,975
MEM_LOAD_UOPS_LLC_MISS_RETIRED.REMOTE_HITM	69	5,585,607

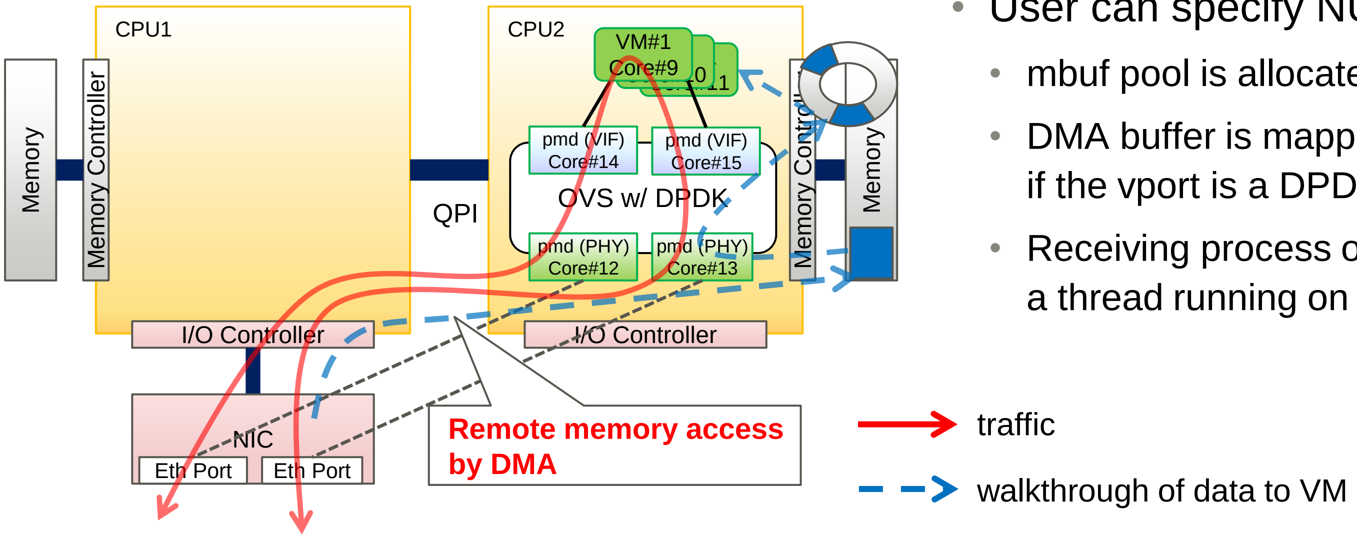
Remote memory access after L3 miss

Removing remote memory access by CPU,
Is it expected to increase throughput?

Additional Test for Assumption

Removing remote memory access by CPU using DMA

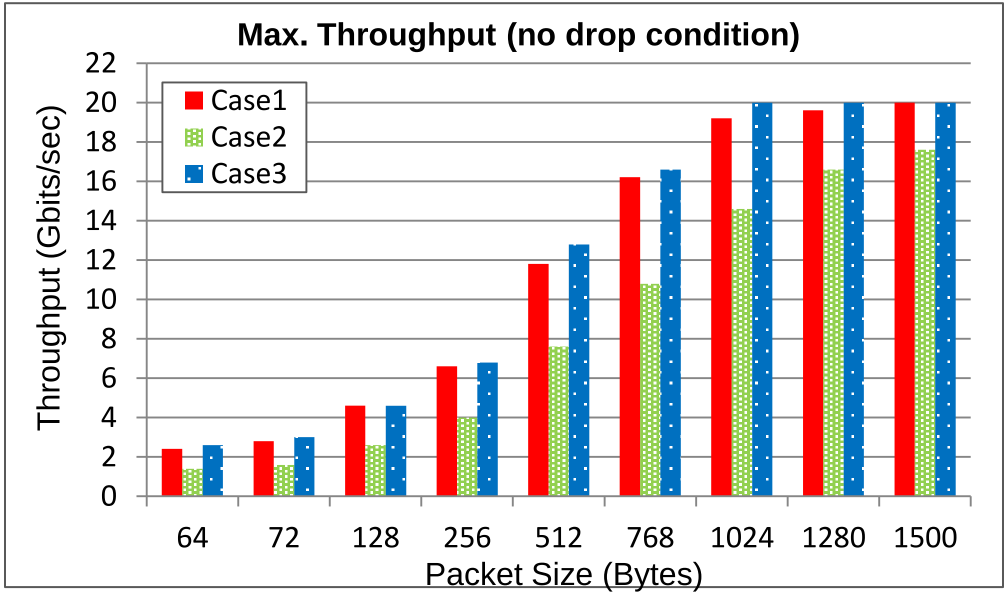
Case3: Only NIC is connected to the NUMA node which is different from all software are running (flows go through QPI by DMA)



- Modify OVS 2.4.1 for this test as follows
  - User can specify NUMA-node per vport
    - mbuf pool is allocated on the specified node
    - DMA buffer is mapped to the specified node if the vport is a DPDK port (physical port)
    - Receiving process of the vport is handled by a thread running on a core of the specified node

Result of Additional Test

- Case3 (flows go through QPI by DMA) achieves high-throughput as same as case1 (no flows go through QPI) !



	% of Case1		
pkt size	Case1	Case2	Case3
64	100.0	58.3	108.3
72	100.0	57.1	107.1
128	100.0	56.5	100.0
256	100.0	60.6	103.0
512	100.0	64.4	108.5
768	100.0	66.7	102.5
1024	100.0	76.0	104.2
1280	100.0	84.7	102.0
1500	100.0	88.0	100.0

Summary on Studying the Issue

FUJITSU

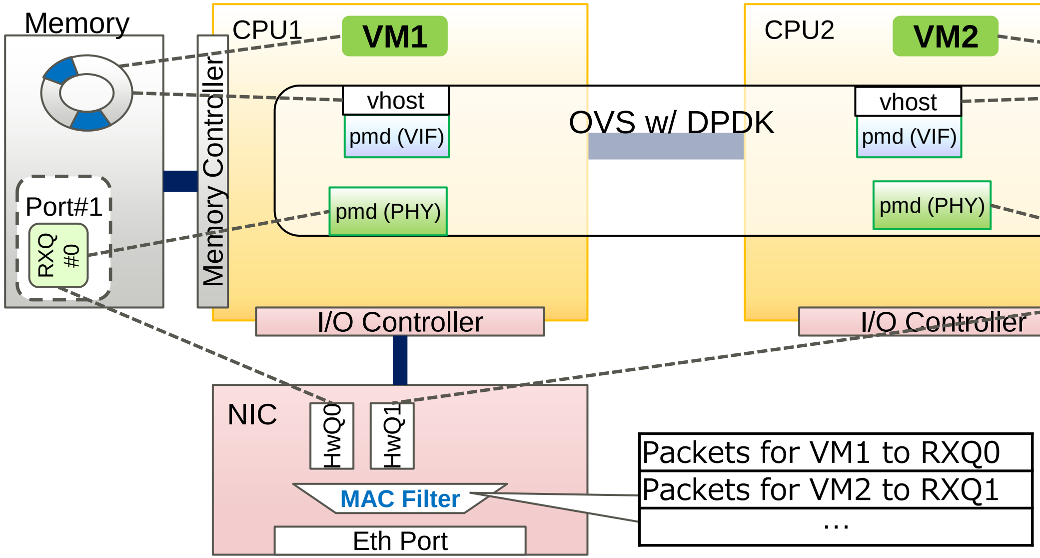
- "Flows go through QPI by CPU" (Case2) mainly influences throughput degradation on NUMA system.
- "Flows go through QPI by DMA" (Case3) achieves comparative performance with "no flows go through QPI" (Case1).

Considering the case VMs are running on all NUMA-nodes,
DMA should write flows to the memory of the NUMA node
on which their destination VM is running

Trial Implementation for NUMA-aware DPDK-OVS

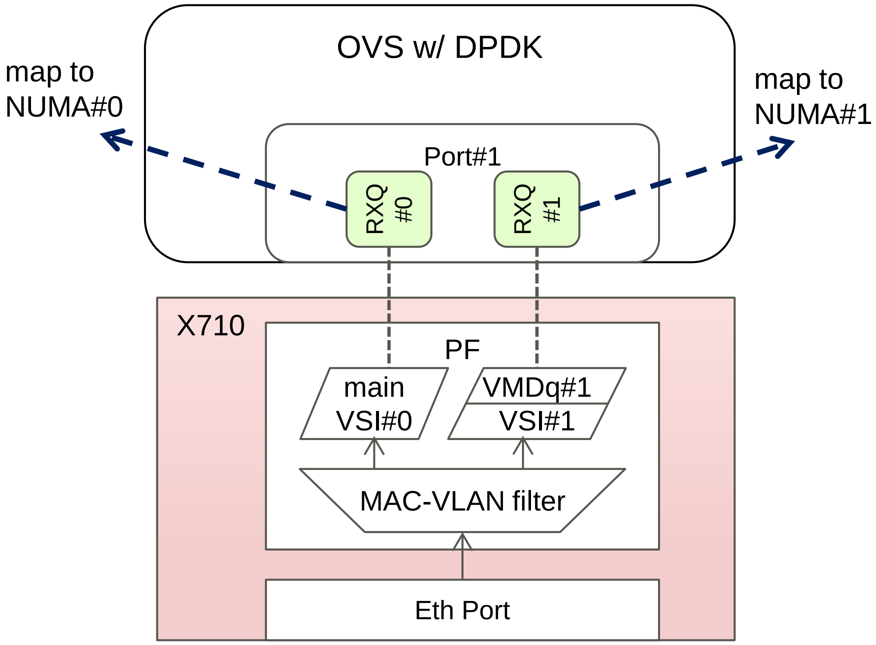
Concept and Basic Design

- Use multiple RXQs & MAC filter features of standard NIC
  - Each NUMA node has a RXQ (associated with a HwQ) on its own memory
  - Flows are forwarded to each node according to the destination MAC address by MAC filter



Trial Implementation w/ Intel® X710

Use VMDq and MAC-VLAN filter



- Dest. VSI can be specified corresponding to dest. MAC addr w/ MAC-VLAN filter
- Packets w/ unregistered MAC addr go to main VSI of PF
  - It only needs to register dest. MAC to go to NUMA#1
  - Broadcast frames can be handled w/o configuration
- Needs some modifications to DPDK
  - Default DPDK can not configure queues of specific VMDq, so we changed i40e PMD to specify the number of queues per VSI including main VSI.

VSI: Virtual Station Interface (a set of queues)

Memory Allocation for NUMA

Allocation of mbuf pool (dpdk_mp)

Current OVS allocates single mbuf pool for a dpdk port with socket_id (numa id) of the device

```
dpdk_mp = dpdk_mp_get(dev->socket_id, mtu);
```

We change it to allocate and hold multiple mbuf pools for each NUMA node for a dpdk port

```
dpdk_mp = dpdk_rte_mzalloc(sizeof(struct dpdk_mp) * n_numas);
for (numa_id = 0; numa_id < n_numas; numa_id++) {
    dpdk_mp[numa_id] = dpdk_mp_get(numa_id, mtu);
}
```

Setup Rx queues

Current OVS sets up all rx queues for a port on the same NUMA node specified by dev->socket_id

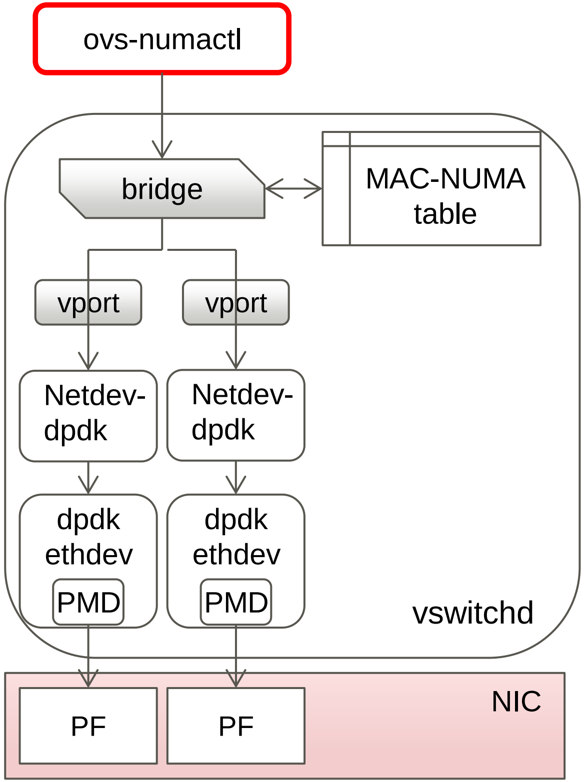
```
n_rxqs = MIN(max_rx_queue, n_dpdk_rxqs);
…
for (i = 0; i < n_rxqs; i++) {
    rte_eth_rx_queue_setup(port_id, i, RX_Q_SIZE,
                           dev->socket_id, NULL,
                           dev->dpdk_mp->mp );
}
```

We change it to setup each Rx queue on the NUMA node specified by numa_id associated with the queue index

```
n_rxq_per_vsi = MIN(max_rx_queue_per_pool, n_dpdk_rxqs);
…
n_rxqs = n_rxq_per_vsi * n_numas;
…
for (qid = 0; qid < n_rxqs; qid++) {
    numa_id = dpdk_get_numa_id_by_qindex(dev, qid);
    rte_eth_rx_queue_setup(port_id, qid, RX_Q_SIZE,
                           numa_id, NULL,
                           dev->dpdk_mp[numa_id]->mp );
}
```

MAC Address Registration

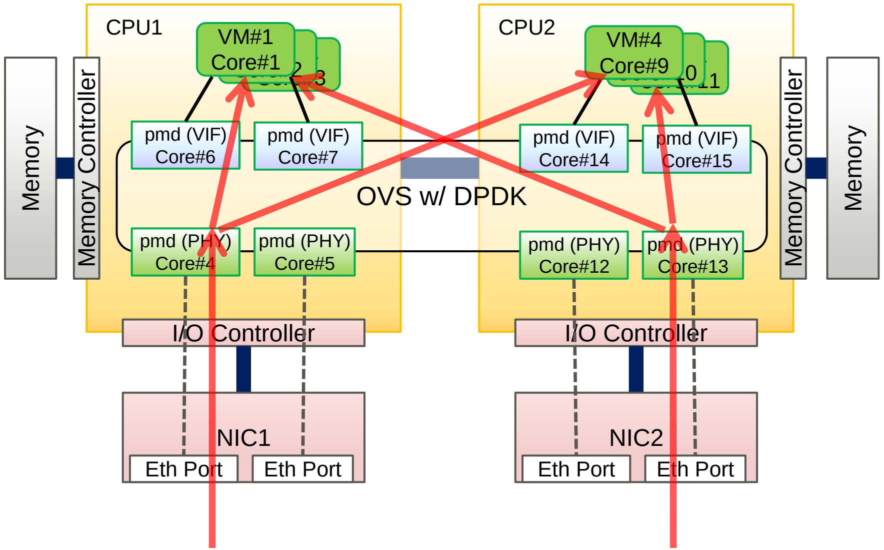
- Develop new utility: ovs-numactl
  - issue command to add / del MAC-NUMA info to bridge
- Add NUMA control features to bridge
  - manage the MAC-NUMA table by receiving command from ovs-numactl
  - set MAC-NUMA info in the table to all dpdk-ports
- Add API to netdev-dpdk
  - set MAC-NUMA info to specified dpdk-port
  - use rte_eth_dev_mac_addr_add() API to register MAC address to MAC-VLAN filter of PF



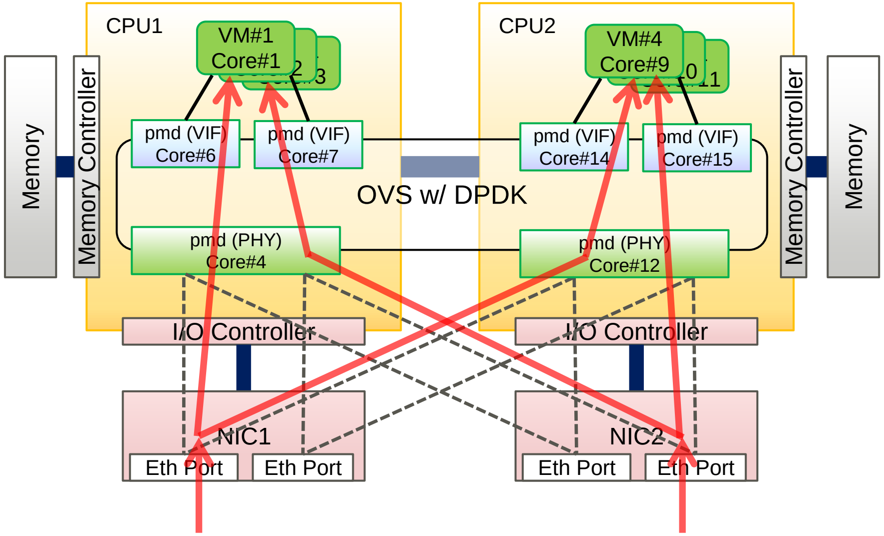
Evaluation

- Measure the total throughput of **6** VNFs (L3 forwarder w/o DPDK)
  - Traffics are inputted from a port of each NIC (maximum 20 Gbps)

Standard OVS: A dpdk port is mapped to only the NUMA node to which its NIC is connected

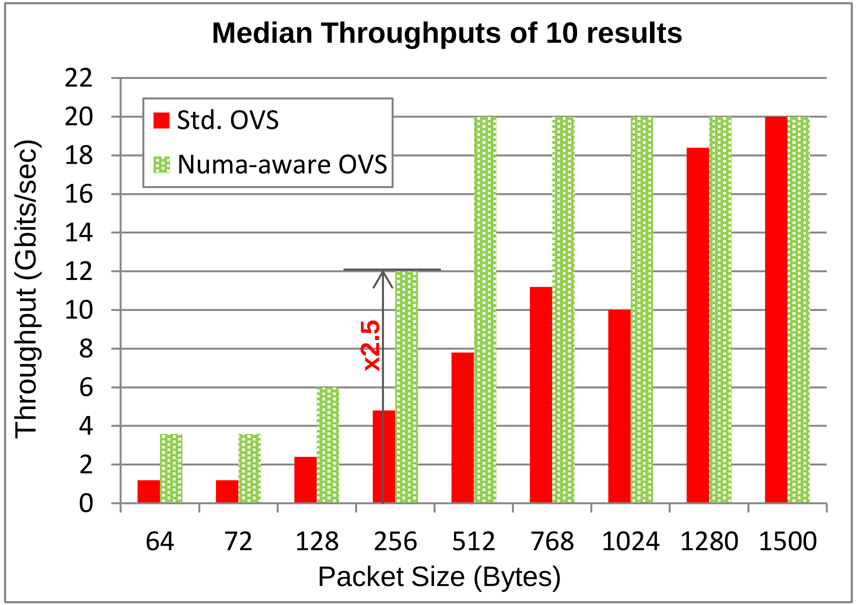


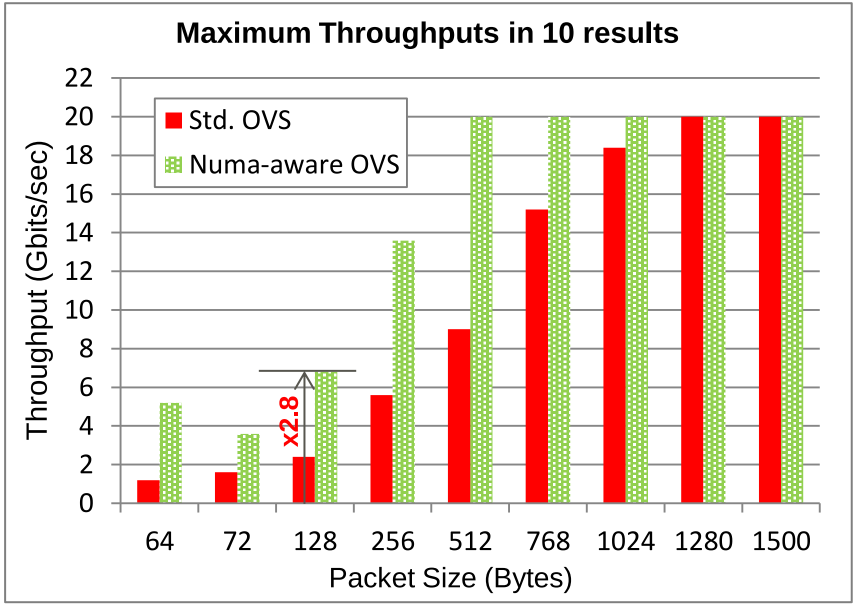
Our Approach: Each dpdk port has multple queues mapped to each NUMA node respectively



Result

- Try 10 times of max rate search at no drop condition for each size
- Our approach achieved **over 2.5 times throughput** in this test





Conclusions

- We introduced a concept and trial implementation of NUMA-aware DPDK-OVS
  - Improvement throughput by reducing remote memory access by CPU
  - Use multiple RXQs and MAC filter to forward received packets directly to the memory of the NUMA node on which its destination VM is running
    - Each NUMA node has a RXQ associated with HwQ on its own memory, and
    - NICs can write received packets directly to there by DMA according dest. MAC address
    - Implement on trial with Intel® X710 (use VMDq and MAC-VLAN filter)
- We observed significant performance improvement in our approach

Conclusions

- Limitations
  - NIC is limited to be enabled in our approach
  - a few MAC address can be registered (depends on NIC specification)
  - only destination MAC address is used to specify destination NUMA node
  - It is difficult to support a BOX using tunnel protocol in current standard NICs
  - etc.
- Need investigations to extend the applicable cases

Thank You!

hyoudou.kazuki@jp.fujitsu.com

shaping tomorrow with you

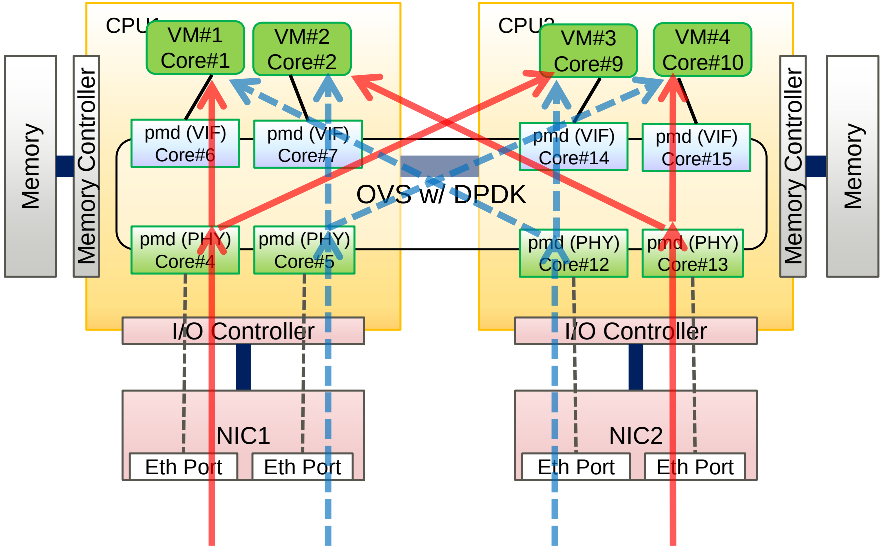
BACKUP

Evaluation (Bi-direction, input from 4 ports)

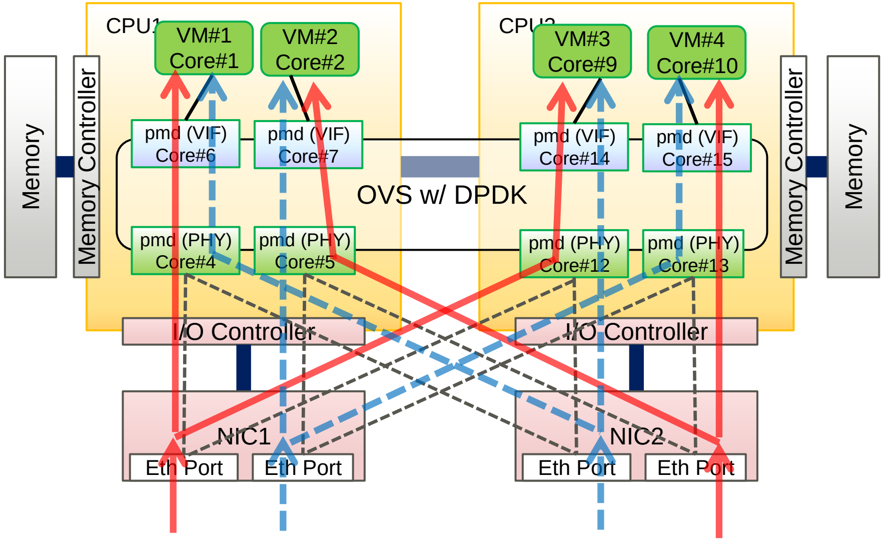
FUJITSU

- Measure the total bi-directional throughput of **4** VNFs (dpdk l2fwd)
  - Each CPU has a dual 10GbE NIC (X710-DA2), four 10GbE ports are used for this test

Standard OVS: A dpdk port is mapped to only the NUMA node to which its NIC is connected

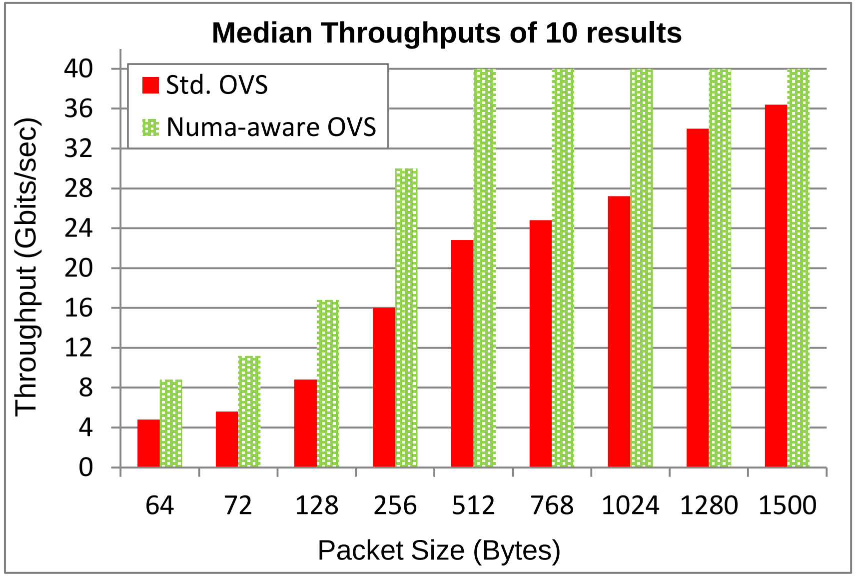


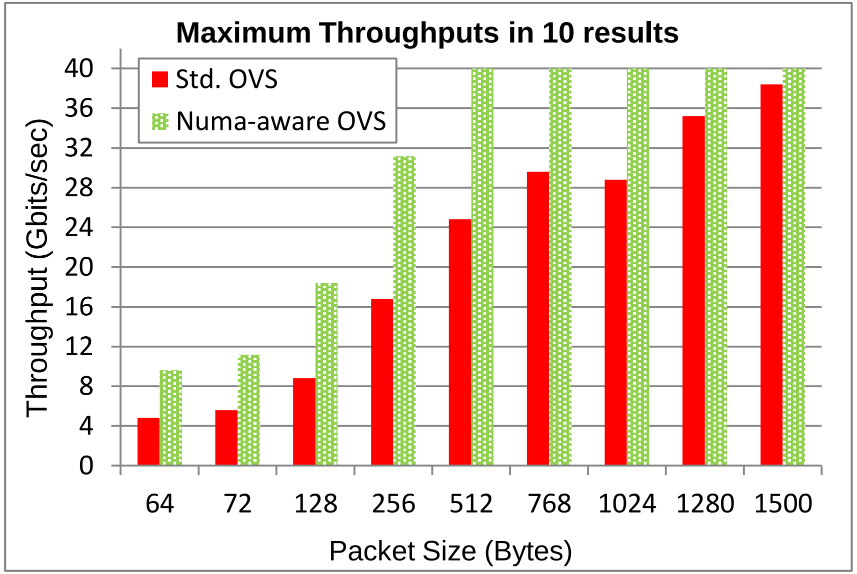
Our Approach: Each dpdk port has multple queues mapped to each NUMA node respectively



Result (Bi-direction)

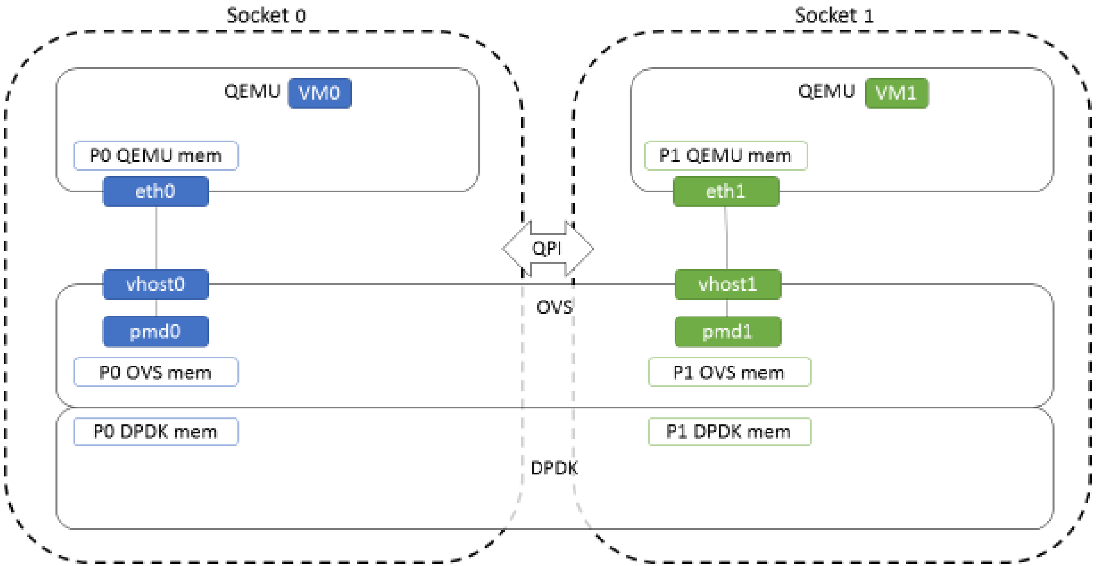
- Our approach achieved **up to 2 times throughput** compared with Std. OVS.





vhost-user NUMA Awareness in OVS 2.6

- Memory for structure & mbuf pool for **vhost** can be allocated on NUMA node on which connected VM is running



Refer to
https://software.intel.com/en-us/articles/vhost-user-numa-awareness-in-open-vswitch-with-dpdk?language=ru

Example of Connection Model for Our Products

FUJITSU

- All VNFs need to transmit / receive to / from all physical ports