Understanding The Performance of DPDK as a Computer Architect

XIAOBAN WU *, PEILONG LI *, YAN LUO *, LIANG-MIN (LARRY) WANG †, MARC PEPIN †, AND JOHN MORGAN †

* UNIVERSITY OF MASSACHUSETTS LOWELL
† INTEL CORPORATION

DPDK US Summit - San Jose - 2016
Outlines

- Background & Motivations
- Introductions to OvS arch and memory hierarchy
- Experiment setup and test methodology
- OvS versus OvS-DPDK performance evaluation
- Multi-socket platform impact analysis
- Conclusion
Open vSwitch (OvS): key connectivity component in cloud/datacenter to provide network of virtualized machines. E.g. OpenStack, and OpenNebula.

Line rate increases (10G → 40G → 100G): OvS is hard to keep up.

DPDK accelerated OvS (OvS-DPDK): known to have higher performance. But why?

We explain why OvS-DPDK has better performance over vanilla OvS from computer architecture’s perspective. E.g. cache behaviors, context switches, etc.
Introduction: OvS Application Scenario

- A typical application scenario of OvS in cloud/datacenter.

- Two communication scenarios:
  - VM → vNIC → VM (Same host)
  - VM → pNIC → VM (Different hosts)
OvS data path:

Virtual Machine

vNIC

User

Kernel

Data Path

ovs-vswitchd

If rule is not cached in kernel

Netlink

ovs_mod.ko

OvS-DPDK data path:

Virtual Machine

DPDK vSwitch

Forwarding

User

Kernel

Data Path

ovs-vswitchd

pNIC

vNIC-DPDK

IVSHM

PMQ Drivers
For a typical Intel Skylake processor

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 Peak Bandwidth (bytes/cycle)</td>
<td>2x32 Load 1x32 Store</td>
</tr>
<tr>
<td>L2 Data Access (cycles)</td>
<td>12</td>
</tr>
<tr>
<td>L2 Peak Bandwidth (bytes/cycle)</td>
<td>64</td>
</tr>
<tr>
<td>Shared L3 Access (cycles)</td>
<td>44</td>
</tr>
<tr>
<td>L3 Peak Bandwidth (bytes/cycle)</td>
<td>32</td>
</tr>
<tr>
<td>Memory Access (cycles)</td>
<td>~ 140 (for 2.0 GHz)</td>
</tr>
</tbody>
</table>

Experiment Setup Overview

- **Guest-Guest (VM2VM)**

  - **Experiment 1**
    - Guest iperf Server
    - VM 0
    - vNIC
    - OVS
    - Host
    - Guest iperf Client
    - VM 1
    - vNIC
    - Experiment 3

- **Guest-Host (VM2Host)**

  - **Experiment 2**
    - Guest iperf Server
    - VM 0
    - vNIC
    - OVS-DPDK
    - Host
    - Guest iperf Client
    - VM 1
    - vNIC

  - **Experiment 3**
    - Guest iperf Client
    - VM 0
    - vNIC
    - OVS
    - Host
    - pNIC
    - iperf Server

  - **Experiment 4**
    - Guest iperf Client
    - VM 0
    - vNIC
    - OVS-DPDK
    - Host
    - pNIC
    - iperf Server

Test Platform Specifications

- **Hardware** - Intel SuperMicro Server
  - Intel Xeon D-1540, 8 Cores @ 2.0 GHz.
  - **L1i**: 32 KB, **L1d**: 32 KB, **L2**: 256 KB, **LLC**: 12 MB, **Memory**: 64 GB.
  - **NIC**: Intel 82599ES 10-Gigabit SFI/SFP+

- **OS**: Ubuntu 16.04; **OvS** version: 2.5.0; **DPDK** version: 16.04

- All the VMs are created by KVM and emulated by QEMU.

- Run **Iperf** (version 2.0.5) test on the provided environment.

- Processor performance profiling tools:
  - Linux **Perf** version: 4.4.13
  - Intel **VTune Amplifier XE** version: 2016 Update 4
### Iperf Test Setup

- **Experiment 1 (VM-OvS-VM)**
  - On VM0 (Iperf Server)
    - `sudo iperf -s -w 512k -l 128k -p 1005 | grep SUM`
  - On VM1 (Iperf Client)
    - `iperf -c 10.0.0.1 -p 1005 -w 512k -l 128k -i2 -t60 -P4 | grep SUM`

- **Experiment 2 (VM-OvSDPDK-VM)**
  - Same as experiment 1, but use OvS-DPDK

- **Experiment 3 (Host-OvS-VM)**
  - Same as experiment 1, but use another host machine as server

- **Experiment 4 (Host-OvSDPDK-VM)**
  - Same as experiment 3, but use OvS-DPDK.
Throughput and IPC comparison for 4 different scenarios:

- **5.5x** throughput increase for VM2VM scenario
- **3x** throughput increase for the VM2HOST scenario
- OvS-DPDK scenarios render better IPC (ideal IPC is 4.0 for 4-issue arch) with pipeline.

**5.5x higher throughput, IPC > 1.0**
Evaluation 2

Cache behavior comparison:

- OvS-DPDK achieves 7x and 8x more cache references for VM2VM and VM2HOST scenarios respectively.
- L1 data cache miss rate is less for both scenarios with OvS-DPDK. Cache miss reduced by 50% for the VM2HOST case with OvS-DPDK.

More cache refs, fewer L1 cache misses (SW prefetching)
Last level cache and table lookaside buffer (TLB) behaviors

- Last level cache has $3 \sim 6$ times more accesses with OvS-DPDK than with vanilla OvS.
- TLB miss rate is near perfect 0.0% if using OvS-DPDK.

More LLC accesses, 0% TLB miss (huge page)
Modern datacenter racks employ multi-socket platform design to scale up performance with the power budget.

How OvS and OvS-DPDK behave on such multi-socket platform?

Our multi-socket test platform:

- 2-socket server
- 2 Intel Xeon E5-2643 v3 Processors, 6 cores @ 3.4 GHz each socket
- \( L1i \): 32 KB; \( L1d \): 32 KB; \( L2 \): 256 KB
- \( LLC \) (L3): 20 MB.
- NUMA Mem0: 8.0 GB; Mem1: 16 GB
Across Socket Experiment Setup

- Within/Across socket with either OvS or OvS-DPDK: 4 different configurations.
- Run Iperf benchmark for each configuration.
Throughput comparison and cache behaviors.

- Throughput difference:
  - OvS: **1.33x** if with same socket
  - OvS-DPDK: **1.1x** if with same socket
- LLC references: >10% less LLC references if running VMs across different socket.

**Same socket: higher throughput, better LLC behavior**
Context switches comparison.

- If comparing OvS vs. OvS-DPDK:
  - Context switches drop dramatically if using OvS-DPDK
- If comparing Same/Diff socket:
  - Not big difference
  - Across socket communication is not the root cause of context switches

This work conducts a thorough performance analysis of vanilla OvS and OvS-DPDK from a computer architect’s perspective.

OvS-DPDK improves system performance by:
- Increasing IPC, cache references;
- Decreasing cache misses (*software prefetching*), TLB misses (*huge pages*), and context switches (*user-space driver*).

A multi-socket platform may lead to:
- Lower system throughput and less LLC accesses.
- Across socket, however, is not the root cause of context switches.
Questions?

Dr. Peilong Li
Peilong_Li@uml.edu
UMass Lowell ACANETS Lab
http://acanets.uml.edu