VSPERF DEEP DIVE: VIRTUAL SWITCH PERFORMANCE IN OPNFV

Maryam Tahhan
Why are vSwitches important and do we need to characterize their performance?

Virtual switches are a key component of Network Function virtualization (NFV) based telecoms networks and Software Defined Networking (SDN) based data centres.

- They provide similar functionality to their physical counterparts.
- One key difference between a virtual switch and a physical switch is that the vSwitch runs on the same server/platform where the VNFs are running.
- Its performance can directly affect the number of subscribers/VNFs that can be supported on a single blade.
Define, implement and execute a test suite to characterize the performance of a virtual switch in the NFVI

Drive standardization

Promote a defined platform and reuse

Establish best practice

30+ committers and contributors
VSPERF Standardization and Open Source Projects

Driving the standard platform – by doing

Feedback
VSPERF Deliverables

IETF Draft

Versions: 00 01

Network Working Group
M. Taban
M. B. O'Mahony
Intelligence
Expires: April 16, 2016

Benchmarking Virtual Switches in Open vSwitch
Draft-Open vSwitch 01

Abstract

This memo describes the progress of the Open Platform for NFV (OPNFV) project on virtual switch performance "VSWITCHPERF". This project intends to build on the current and completed work of the Benchmarking Methodology Working Group in IETF, by referencing existing literature. The Benchmarking Methodology Working Group has traditionally conducted laboratory characterization of dedicated physical implementations of networking functions. Therefore, this memo begins to describe the additional considerations when virtual switches are implemented in general-purpose hardware. The expanded tests and benchmarks are also influenced by the OPNFV mission to support virtualization of the "telco" infrastructure.

Test Specification

1. CHARACTERIZE VSWITCH PERFORMANCE FOR TELCO NFV USE CASES LEVEL TEST DESIGN

Modular Test Framework

Consumable by:

OPNFV

Consumable by:

VSPERF

VNF(s)
vSwitch

Traffic Gen

Traffic Gen Client

DUT
TEST SPECIFICATION APPROACH AND TEST COVERAGE
VSPERF test specification approach

Start with existing RFCs/Specifications that describe the testing of physical switches, and are applicable to vSwitches.

Define additional tests applicable to virtual switches, such as: noisy neighbour tests, datapath and control path coupling, CPU and memory utilization...
BMWG and IPPM specs referenced by VSPERF

- [RFC2889] Benchmarking Methodology for LAN Switching
- [RFC6201] Device Reset Characterization
- [RFC3393] IP Packet Delay Variation Metric for IPPM
- [RFC5481] Packet Delay Variation Applicability Statement
- [RFC2285] Benchmarking Terminology for LAN Switching Devices
## VSPERF Test Coverage Matrix

<table>
<thead>
<tr>
<th></th>
<th>Speed</th>
<th>Accuracy</th>
<th>Reliability</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RFC2889. AddressCachingCapacity InitialPacketProcessingLatency LatencyAndLatencyVariation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Deactivation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
KEY CONSIDERATIONS WHEN BENCHMARKING VSWITCHES
Key considerations when benchmarking vSwitch performance

Three fundamental considerations:

- vSwitches should be benchmarked in a similar way to physical switches – when possible – to allow for a direct comparison.

- Try to ensure accuracy, consistency, stability and repeatability of the results between test runs.

- It’s essential to limit and if possible eliminate any noise that may interfere with the accuracy of the metrics collected by the test.
Consistency and Stability of Test Results

Can be achieved through appropriate configuration of:

- BIOS
- The OS/Kernel.
- The system hardware.
- The system software.
Consistency and Stability Configuration
Parameters BIOS Configuration

- BIOS should be configured for performance where an explicit option exists
- sleep states should be disabled
- any virtualization optimization technologies should be enabled
- hyper threading should also be disabled
- turbo boost and overclocking should also be disabled.
Consistency and Stability Configuration
Parameters for the OS/Kernel

Boot/Grub Parameters:

- `maxcpus = n` where `n >= 0`; limits the kernel to using 'n' processors. Only use exactly what you need.

- `isolcpus`: Isolate CPUs from the general scheduler. Isolate all CPUs except one which will be used by the OS.

- Taskset/affinitize the vSwitch and the VNFs onto isolated cores.

- Note: VNFs and the vSwitch must not share the same cores. vCPUs for the VNF should be affinitized to individual cores also.
Consistency and Stability Configuration for System HW and SW

- Limit the amount of background applications that are running.
- Set OS to boot to runlevel 3. Make sure to kill any unnecessary system processes/daemons.
- Only enable hardware that you need to use for your test – to ensure there are no other interrupts on the system.
- Configure NIC interrupts to only use the cores that are not allocated to any other process (VNF/vSwitch).
Repeatability of Test Results

In addition to the previous system configuration recommendations it is critical to document all aspects of the SUT to foster repeatability, including:

- Hardware parameters
- Software parameters
- Traffic profile/parameters
Hardware Parameters to document for repeatability

- Platform details
- Processor details
- Memory information
- Number of enabled cores
- Number of cores used for the test
- Number of physical NICs.
- NIC interrupt configuration

- BIOS version, release date and any configurations that were modified
- Memory DIMM configurations (quad rank performance may not be the same as dual rank) in size, freq and slot locations
- PCI configuration parameters (payload size, early ack option...)
- Power management at all levels (ACPI sleep states, processor package, OS...)
Software Parameters to document for repeatability

- OS version
- Kernel version
- GRUB boot parameters
- Hypervisor details
- Selected vSwitch, version number
- vSwitch launch command line
- Memory allocation to the vSwitch
- Libraries or any other SW dependency versions.

- Memory allocation to a VM
- VM launch commandline
- VM storage type
- Number of VMs
- Number of Virtual NICs (vNICs), versions, type, driver and interrupt configuration.
- Number of virtual CPUs and their core affinity on the host
Software Parameters to document for repeatability cont.

- Thread affinitization for the applications (including the vSwitch itself) on the host and VNF.
- Details of Resource isolation, such as CPUs designated to Host Kernel (isolcpus) and CPUs designated for specific processes (taskset).
Traffic Parameters to document for repeatability

- Traffic type - UDP, TCP, IMIX / Other
- Transmission duration.
- Packet Sizes
- Deployment Scenario
- Number of flows.
BENCHMARKING METHODOLOGY
VSPERF Typical System Under Test (SUT)
VSPERF Benchmarking Methodology

- Benchmark platform forwarding capability.
- Benchmark VNF forwarding capability.
- Benchmarking the vSwitch:
  - With isolated resources alone, leaving some resources unused.
  - With isolated resources alone, with other resources (both HW&SW) disabled.
  - With Isolated resources and all resources occupied.
Baseline Platform forwarding capability

- Run RFC 2889 Maximum forwarding rate test
- Transmit traffic at line rate/max forwarding rate (whichever is higher) for at least 72 hours, measure the forwarding rate (fps) and latency.
- Traffic should be bidirectional.
- Alternative approach is to run an RFC2544 0% frame loss throughput test.
Benchmarking the VNF forwarding capability

- Run RFC 2889 Maximum forwarding rate test
- Transmit traffic at line rate/max forwarding rate (whichever is higher) for at least 72 hours, measure the forwarding rate (fps) and latency.
- Traffic should be bidirectional.
- Alternative is to run an RFC2544 0% frame loss throughput test.

Note: The VNF configuration used for this test, is the configuration that should be used for all subsequent tests.
Benchmarking the vSwitch

Essentially executing the full test specification on a Typical SUT.

The baseline deployment scenarios to be tested by VSPERF to date, include:

- Physical to physical (phy2phy).
- Single VM loopback framework (PVP).
- Two VM loopback framework (PVVP).
- VM to VM (VM2VM)..
Physical to Physical (Phy2Phy)

Physical port → vSwitch → Physical port.

Note: the vSwitch runs on the host.
VM Loopback (PVP)

Physical port → vSwitch → VNF → vSwitch → Physical Port

Note: the vSwitch runs on the host
Two VM Loopback (PVVP)

Physical port → vSwitch → VNF → vSwitch → VNF → vSwitch → Physical Port

Note: the vSwitch runs on the host
VM To VM (VM2VM)

Hasn’t been implemented yet

Concerns around time synchronization between VMs and clock accuracy.

Recommendation under consideration: Test must include an external HW traffic generator to act as the tester/traffic source and sink.
VSPERF TEST FRAMEWORK
VSPERF Framework
Overview

A Python based test framework for characterizing the performance of virtual switches.

As of today, capable of conducting the shown tests on Stock OVS and OVS with DPDK

Supported deployment scenarios: Baseline Platform benchmark, Phy2Phy, PVP and PVVP.

Supported Traffic gens: Ixia, Spirent and Dummy.
VSPERF Framework
Details
Runs on the System Under Test (SUT), looks after:

- vSwitch setup
- VNF setup
- Traffic Generator setup through a traffic gen client.
- Collecting the results from the traffic generator
Deployment scenarios: Phy2Phy (no vSwitch)

Physical port → Testpmd (no vSwitch) → Physical port.
Deployment scenarios: Phy2Phy

Physical port → vSwitch → Physical port.

Note: the vSwitch runs on the host
Deployment scenarios: PVP

Physical port → vSwitch→ VNF→ vSwitch → Physical Port

Note: the vSwitch runs on the host

The VM runs DPDK testpmd/L2fwd (!DPDK)/Linux Bridge to forward traffic.

Traffic consists of uniform UDP packets.
Deployment scenarios: PVVP

Physical port → vSwitch → VNF → vSwitch → VNF → vSwitch → Physical Port

Note: the vSwitch runs on the host

The VM runs DPDK testpmd/L2fwd (!DPDK)/Linux Bridge to forward traffic.

Traffic consists of uniform UDP packets.
Selection of Loopback applications in the Guest

**DPDK testpmd**: set to forward traffic.

**L2FWD**: A Kernel Module that provides OSI Layer 2 IPv4 termination or forwarding with support for Destination Network Address Translation (DNAT) for both the MAC and IP addresses.

**Linux Bridge**
VNF and Traffic Gens

VSPERF uses a VM called vloop_vnf for looping traffic in the PVP and PVVP deployment scenarios.

- The image can be found @ http://artifacts.opnfv.org/vswitchperf/vloop-vnf-ubuntu-14.04_20151216.qcow2
- Alternatively you can use your own QEMU image.

Supported traffic gens: IXIA, Spirent and Dummy.

- Dummy: allows you to run your own traffic gen, VSPERF will setup the vswitch and the VNF, the end user then sets up the traffic gen manually.
VSPERF Directories

conf: contains the main configuration for testcases, vSwitches, traffic generators and VNFs.

core: contains the controllers for vSwitches, traffic generators and VNFs. These controllers look after the configuration and launching of deployment scenarios for vSwitches, traffic generators and VNFs. Implementation details of vSwitches, traffic generators and VNFs are abstracted from the controllers.

Systems: contains the script that look after installing system dependencies and setting up the python environment for vsperf.

src: looks after cloning and building the libraries/utilities we depend on including OVS, DPDK and QEMU (v2.3 for CentOS 7).

testcases: contains the base implementation of a test case.

tools: contains the implementation of packet generators, collectors and load generators.

vnfs: contain the implementation that allow us to launch a VM with a particular hypervisor.

vswitches: contains the implementation of virtual switches.
VSPERF Documentation

http://artifacts.opnfv.org/vswitchperf/docs/index.html

Includes:

User guides
Design guides
Test Specification
Release Information
Running Tests

Most of the instructions to run a test can be found in the documentation and an example @ https://wiki.opnfv.org/get_started/pod_3_characterize_vswitch_performance#running_tests

A summary of steps:

- Clone vswitchperf
- Install the required packages.
- cd src and make
- cd ../conf, copy and modify the 10_custom.conf with appropriate settings.
- cd .. and ./vsperf –list
- ./vsperf –conf-file=$HOME/10_custom.conf phy2phy_cont
VSPERF CI - OPNFV

https://build.opnfv.org/ci/view/vswitchperf/

Uses ci/build-vsperf.sh (vsperf) and Jenkins Job builder (jjb) + job description vswitchperf.yml file (releng)

Has daily, merge and verify jobs.

For more info on writing and using jjbs:
https://wiki.opnfv.org/octopus/jenkins_wow?s[]=jjb
build-vsperf.sh
### VSPERF CI – OPNFV Output

```
/tmp/results_2016-01-19_17-04-26/result_phy2phy_tput_p2p.md
```

- **Test ID:** phy2phy_tput
- **Description:** LTD.Throughput.RFC2544.PacketLossRatio
- **Deployment:** p2p
- **Traffic type:** rfc2544
- **Bidirectional:** True

<table>
<thead>
<tr>
<th>Metric</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput_rx_fps</td>
<td>22626032.812</td>
</tr>
<tr>
<td>throughput_rx_mbps</td>
<td>11584.529</td>
</tr>
<tr>
<td>tx_rate_percent</td>
<td>76.023</td>
</tr>
<tr>
<td>throughput_rx_percent</td>
<td>76.023</td>
</tr>
<tr>
<td>min_latency_ns</td>
<td>5260.000</td>
</tr>
<tr>
<td>max_latency_ns</td>
<td>138000.000</td>
</tr>
<tr>
<td>avg_latency_ns</td>
<td>7392.000</td>
</tr>
<tr>
<td>type</td>
<td>rfc2544</td>
</tr>
<tr>
<td>packet_size</td>
<td>64</td>
</tr>
<tr>
<td>traffic_type</td>
<td>udp</td>
</tr>
</tbody>
</table>
Additional Info

Gerrit: https://gerrit.opnfv.org/gerrit/#/q/vswitchperf

List of Candidate Work Items

- List of candidate work items for "vSwitch Performance"
- vSwitchPerf New Test Etherpad

Demo: https://prezi.com/tlriipoafn5_/vsperf-demo
Summary

- It’s important to understand the performance of vSwitches as they are a key component in the NFV infrastructure.
- vSwitches should be benchmarked in a similar way to physical switches – when possible.
- Consistency, stability and repeatability can be achieved through appropriate configuration and documentation of the SUT.
- Today, VSPERF describes the baseline scenarios to test the maximum achievable performance of the vSwitch. More realistic scenarios are WIP.
- VSPERF provides a python framework that implements a subset of the VSPERF test specification.
VSPERF Project Future Work

- Integrating multiple traffic gens with the Test Framework: Moongen and Xena
- VM Baseline performance using PCI Passthrough/SR-IOV
- Methodology extensions: Iterations for the short trial tests
- Prove out and refine methodology and tests through the framework
- Add more tests to the Test Spec and the framework, an initial list:
  - Scalability Tests
  - Overlay Networking Tests
  - Match action performance testing
  - Classifying L2, L3 and L4 traffic Profile Tests