Performance Comparison of Open Virtual Switch (OVS) with 4x4 switch

CS 648

Aniruddha Kushwaha - 154056001
Aditya Kusupati - 130050054

Introduction:

The main goal of the DPDK is to provide a simple, complete framework for fast packet processing in data plane applications. The framework creates a set of libraries for specific environments through the creation of an Environment Abstraction Layer (EAL), which may be specific to a mode of the Intel® architecture (32-bit or 64-bit), Linux* user space compilers or a specific platform. These environments are created through the use of make files and configuration files. Once the EAL library is created, the user may link with the library to create their own applications. Other libraries, outside of EAL, including the Hash, Longest Prefix Match (LPM) and rings libraries are also provided. These libraries can be used to:

- Receive and send packets within the minimum number of CPU cycles (usually less than 80 cycles)
- Develop fast packet capture algorithms (tcpdump-like)
- Run third-party fast path stacks

In this work, we compared the performance of OpenVSwitch with 4x4 L2 switch in terms of latency.

OVS and 4x4 L2 switch:

Open vSwitch is a production quality open source software switch designed to be used as a virtual switch (vswitch) in virtualized server environments. A vswitch forwards traffic between different VMs and also forwards traffic between VMs and the physical network. Open vSwitch supports standard management interfaces (e.g. sFlow, NetFlow, IPFIX, RSPAN, CLI), and is open to programmatic extension and control using OpenFlow and the OVSDB management protocol. [1] When DPDK is enabled along with OVS then the performance gains are significant.

4x4 L2 switch is a simple MAC based forwarding switch written over DPDK to run in real/virtualized environment on a linux machine. Its code is such that it can be scale to any number of ports given that host machine have sufficient resources (memory, network adapters etc.)

Experimental Setup

OVS:

We setup the OVS on machine running linux 16.04. OVS require support of 1G hugepages therefore for the experiment we allocated 8 1G hugepages to DPDK. We added four simple port forwarding rules.

Rule 1: Forward packets from port-1 to port-4.
Rule 2: Forward packets from port-4 to port-1.
Rule 3: Forward packets from port-2 to port-3.
Rule 4: Forward packets from port-3 to port-2.

Experimental setup is shown in fig. 1. To take the results we connected the JDSU MTS 6000 traffic generator. We pumped traffic from one port and gathered the statistics at the other port.

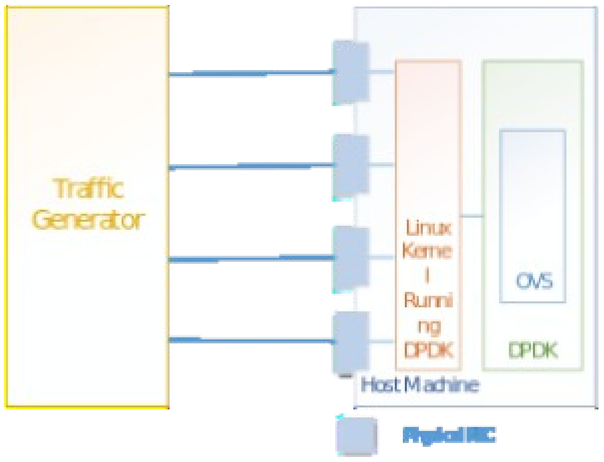

Figure 1 Experimental setup for OVS

4x4 L2 switch:

To perform this experiment DPDK require NIC support. Due to unavailability of DPDK supported NIC we performed this experiment in virtualized environment. We setup virtual box on host running Ubuntu 16.04. VM also run Ubuntu 16.04. We added four virtualized NIC to the VM which are supported by DPDK to create a 4x4 switch. We bridged these VM port to the physical port.

To perform this experiment we modified the L2 forwarding sample application provided by the DPDK which simply forward traffic from one port to other. We added hash table which take the mac address as the key and provide the output port for the MAC looked. Experimental setup is shown in fig. 2. Similar to OVS experiment we connected the JDSU MTS 6000 traffic generator at the ports. We pumped traffic from one port and gathered the statistics at the other port. We assume one hash table as one rule. We varied the number of rules by varying the number of hash tables. We emulated this by looping the hash table and take the output port after running loop for defined number of loops.

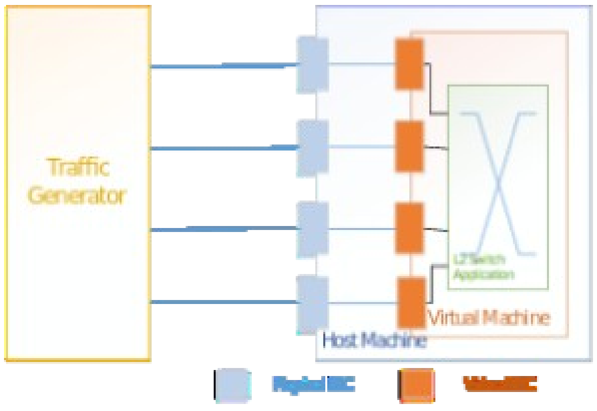

Figure 2 Experimental setup for running L2 switch on DPDK in virtual environment

Result:

Based on our experimental setup for OVS we calculated the round trip latency for packets sent by generator. Fig.3 shows the roundtrip latency with variation in the packet size. This latency increase with the increase in size of packets. We believe this increase in latency is due to the store and forward mechanism. Larger size packet take larger time to store. Packet size 2000 in fig.3 shows the latency when packet size is random. These results are obtained at the 10% of maximum traffic (100mbps). We observed that maximum throughput achieved was only 10%. Switch start dropping packets if pump traffic more than 10% of maximum traffic rate.

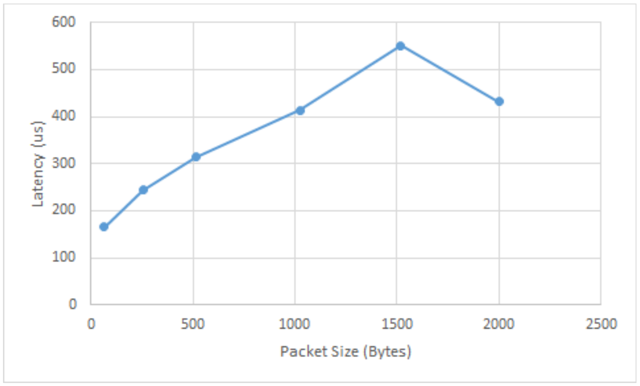

Figure 3 Round trip latency of OVS

Based on the experimental setup of L2 switch we calculated the roundtrip latency. Here we also varied the number of rules that can be applied on a packet. It is observed that with increase in the number of rules latency remain almost similar. We believe this behavior is due to the fact that it take few cpu cycles (i.e. nano seconds) to perform lookup in a hash table. So even for 128 rules lookup these latency aggregate to few *us* only. We also observed that this latency is almost double as compare to OVS. We attribute this to the fact that OVS is running over host whereas switch run over VM. Therefore there is additional delay as packet need to pass through all kernel stacks of host machine to reach virtual machine.

These experiments are taken at maximum throughput of 3-5% of maximum rate.

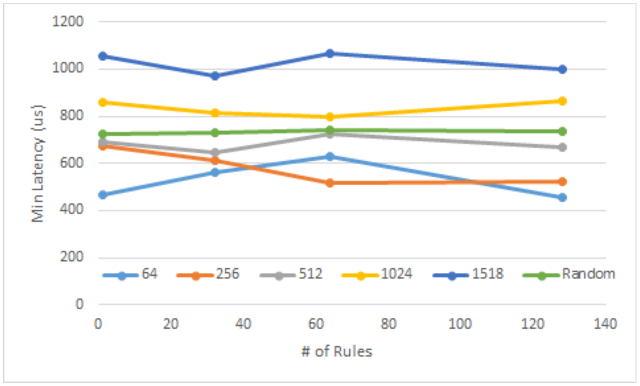

Figure 4 Round Trip Latency of L2 switch for different packet size

Limitation:

DPDK and OVS both had limitations and constraints in terms of dependencies. DPDK cannot be run on host machine as it requires the network adapters supporting DPDK. OVS cannot run over virtual machine as OVS require 1G hugepages whereas VM does not support 1G hugepages.

References:

[1] https://github.com/openvswitch/ovs/blob/master/FAQ.rst
[2] http://openvswitch.org/support/dist-docs-2.5/
[3] http://dpdk.org/doc/quick-start
[4] http://dpdk.org/doc/guides/sample_app_ug/l2_forward_real_virtual.html