The Case for Intel® FM10000 in KVM Acceleration Part II

DPDK OVS to Bridge the Gap

General

Attaching a physical PCIe device, most commonly a network interface, in pass through mode to one specific VM is not a new idea. In fact, it is a basic technique, exposing the device control and data planes solely to a VM, while completely bypassing the hypervisor's networking stacks.

The benefit in pass through mode is brought by low latency and high bandwidth. However, VM that runs in pass through mode could be deprived of a long list of features:

- ***Orchestration related***
  - Suspend
  - Resume
  - Migrate
  - High availability
- ***Networking virtualization related***
  - MAC/VLAN
  - VXLAN
  - NVGRE
- ***Compute virtualization related***
  - VM snapshot
  - Compute load distribution
  - Virtual memory or CPU hot plug and play

Not just a physical PCIe device (or physical function, PF) can pass through a hypervisor straight to a VM. Virtual PCIe device, a virtual function (VF), can be configured as well in pass through mode, on the same token. Virtual function implementation of a network device includes also a rudimentary switching fabric, to forward traffic across VFs, most commonly based on MAC/VLAN forwarding. However, this inter VFs forwarding plane is separate than that of the hypervisor; so in effect, the passed through VM's networking, is isolated form that of the rest of VM's, that in turn runs on the same hypervisor.

This is a significant trade off. A VM can either run with ultimate network efficiency, isolated from the surrounding virtualization or the larger orchestration; or, on the other hand, benefit from long list of features and capabilities, but with a considerable compromise in network efficiency and scale.

Closing the Gap

Promptly after launching a series of Intel® FM10K based adapters earlier this year, Silicom embarked on a series of integration tests with FM10K technology together with OpenVSwitch, in order to demonstrate the value of offloading the data path to a hardware forwarding plane. To that end, Silicom devised a three stages PoC. Initial test with software based OpenVSwitch was done to set the base line of performance. A virtual machine running on KVM was bombarded with 10Gbps traffic and with DPDK based `l2fwd`, shot it back to OVS, which in turn forwarded it onwards to the next VM. Once again this test proved what is already known, that OpenVSwitch, regardless of CPU utilization, cannot uphold volumes of traffic bandwidth higher than several hundreds of megabytes. In fact, there was no point trying to forward traffic from the second VM in the chain, for degradation was over 90%.

Second test involved pure SR-IOV. Each VM was assigned a VF in pass through mode, with traffic injected towards first VM, only to be forwarded onwards to the next, utilizing MAC/VLAN based traffic engineering. In this test it was evident how SR-IOV brings us close to wire speed, and actually poses almost no constraints on traffic, other than those of bus bandwidth. Up to 5 VMs were forwarding 10Gbps

traffic freely among each other. Those numbers was discussed in the first part of this white paper [1].

Welcoming DPDK OVS

Next stage in the series of tests, was directed to combine OVS data path operation with that of Intel® FM10K. During test design work, however, it was discovered that OVS integration into DPDK code base and data path has reached a stable state, with quite promising tests results, compared to a standard software OpenVSwitch.

DPDK as a user space data plane implementation has taken OpenVSwitch data path into user space as well. This modification per se is a huge benefit when it comes to performance; since there is no need for tap network device to act as backend interface to VNIC. This mere fact had an immense effect of performance improvement of DPDK OVS compared to standard OpenVSwitch.

Designing the Data Path

So DPDK OVS infrastructure was chosen to work with. Instances of DPDK poll mode drivers were set above SR-IOV VFs coming out of FM10K based PCIe host network interface. On the VM side, the OpenVSwitch exposed backend VNIC that enabled the use of `virt-io` within the VM. The only missing part of this jigsaw puzzle was a flow that was added to the OpenVSwitch forwarding plane, to connect the dots, i.e., connect the DPDK based interface coming up from the VF and to the VNIC backend of the VM network interface.

Remaining with `virt-io` is a huge plus. It makes a data path the capable of supporting virtualization and orchestration capabilities (listed above).

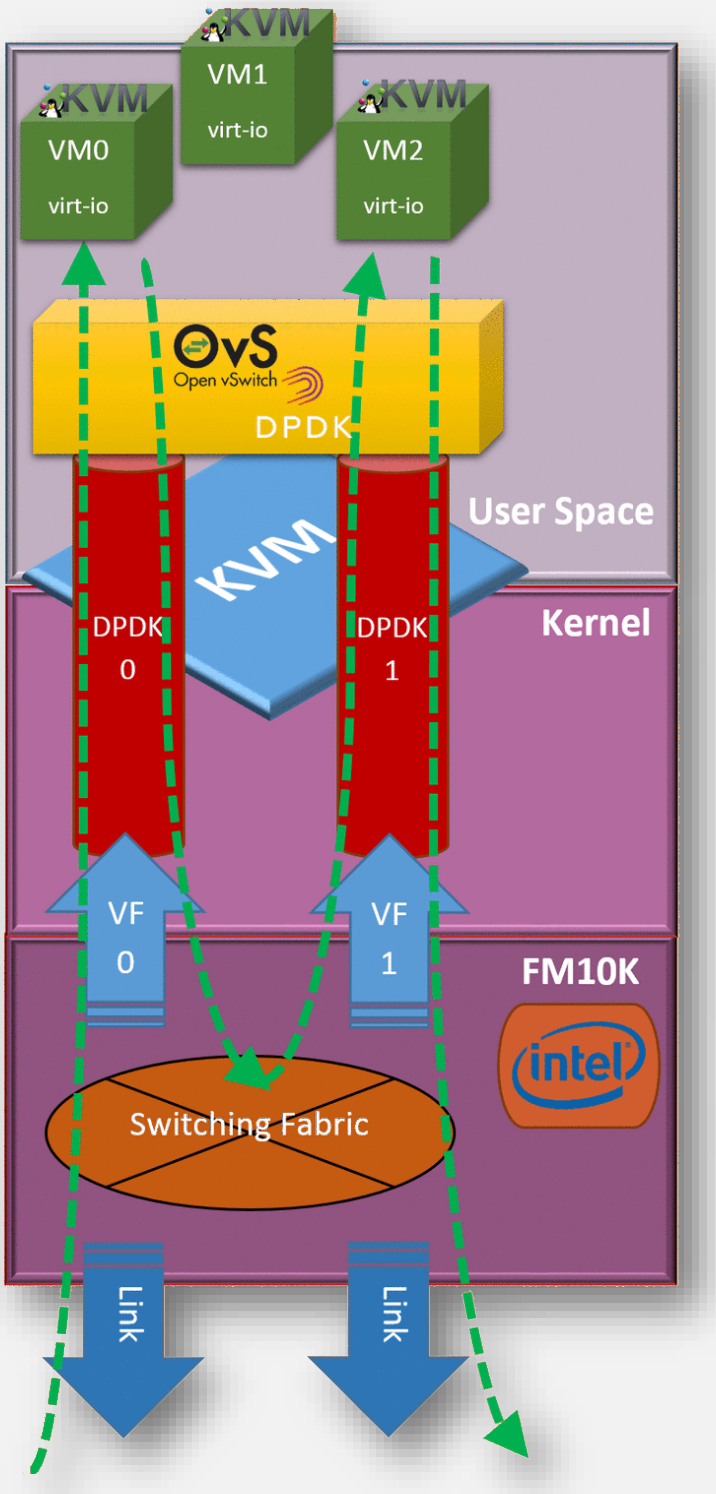

Figure1 - Intel® FM10K and OVS seamless data path

	1	2	3	4	5
OVS	523,160,704				
SR-IOV	9,621,494,968	9,618,339,880	9,613,687,488	9,608,780,080	7,739,236,808
OVS/dpdk	2,119,689,968				
OVS/dpdk SR-IOD Add-flow (2M)	9,623,167,256	9,621,062,742	8,051,743,000		

Table1 - Forwarding tests results in bits per second

Tests Results

The summary of tests results is brought herein, whereas all results can be seen at a glance. On top are the results of software OVS that were taken as reference, exhibiting less than 1 Gbps of forwarding capability. SR-IOV numbers are shown right afterwards, defining the top half of the range of forwarding performance target, with close to wire speed results in a chain of VMs. Third row of results shows once again software only forwarding of Intel® DPDK as backend interface for the OVS, where it really helped mitigate the overhead and improved results. A work very well done by Intel®.

Last row show the results of the target configuration of this test, and the results are encouraging. The FM10K NIC was configured to work in SR-IOV mode, enabling number of VFs, the same as the number of VMs. The MAC table of the NIC's internal switch was configured using `ip` utility ("`ip link set`"). The hardware forwarded packets between VFs, based on MAV/VLAN traffic engineering. All VFs were managed by DPDK and used as backend interface for the OpenVSwitch. The hardware switching and forwarding lets us use OpenVSwitch in its own "pass through" mode, that mapping backend and frontend interfaces of the OpenVSwitch. With direct map – that actually was done with "`vsctl add flow`" – OpenVSwitch was relieved of the task of performing heavy match and table lookup activities for the packet steering; and as a result yielded and measured considerably better performance numbers.

Where to Go Next

Utilizing the Switching fabric of FM10K – Inter VF forwarding in this test was performed by SR-IOV switching fabric on the MAC front of the FM10K. A treasure of rich and modern switching and forwarding fabric lies deeper within the controller, with 32K deep TCAM and FlexPipe™ filtering and forwarding engine, just waiting to be harnessed into operation.

Inter VM Forwarding at 100Gbps – The FM10K's host interface consists of single or multiple PCIe v3 x8 interface, capable of up to 50Gbps of bandwidth. Silicom offer Intel® FM10K based adapter with two such host interfaces, with nominal power to support true 100Gbps inter VM forwarding.

Keeping all the goodies – We've demonstrated how virt-io based data path is supported in a scalable manner by SR-IOV hardware forwarding. One of the next challenge would be demonstrating actual live migration from one SR-IOV to another.

Two forwarding plane as one – By implementing OVS `netdev` interface to Intel® FM10K's MAC and OVS `dpif` interface to FM10K's switching fabric, OVS control plane would gain the power to configure the data plane presented herein.

Test setup included SuperMicro X9DR3-F based sever with Xeon E5-2650 @ 2.6 GHz, and Spirent test center for traffic generation.

REFERENCES

[1] RRC-OVS-OFFLOAD-POC062016.pdf

Silicom ltd., 14 Atir Yeda St., Kefar Sava 4464323, ISRAEL
sales@silicom.co.il www.silicom-usa.com
Intel is a registered trademark of Intel Corporation. All other trademarks are the property of their respective owners.
Silicom reserves the right to make changes without further notice to any products or data herein to improve reliability, function or design. Copyright© 2016 All Right Reserved