The Core Cost of Doing Business

Don Wallwork, *Broadcom Corporation*
Andy Gospodarek, *Broadcom Corporation*

Abstract

Achieving line-rate reception or transmission of flows with the Linux kernel datapath often seems trivial with technologies like LRO, GRO, and TSO. However, forwarding traffic via the Linux kernel datapath often has a greater cost per packet to make sure it arrives at its intended destination -- whether that destination is a VM, container, or a remote system. This talk will cover in detail:

- Cost of the kernel datapath forwarding operation
- Typical cost of the destination lookup of a packet -- including how flow table state impacts lookup speed
- Impact of the size of the frames on link utilization

The intent is to present a formula for estimating the minimum number of server cores needed to service traffic at particular rates and frame sizes with IPv4 and IPv6 as well and VxLAN encapsulated/tunneled traffic with the goal of understanding how the utilization of these cores impacts planning and deployment of new systems.

1. The Rise of Open vSwitch

In the decade since its first release[1], Open vSwitch has become a popular tool which has helped realize the vision of Software Defined Networking (SDN) across datacenters. The network abstraction that it provides allows datacenter operators to deliver valuable services using compute resources that are not bounded by physical location within a given datacenter. Inclusion in Linux Kernel version 3.3 was a major milestone that increased its popularity.

In addition to implementing support for the Linux kernel as a *datapath provider*, Open vSwitch added support for two more significant datapaths to increase performance: DPDK and TC Flower.

The DPDK framework was initially used to accelerate the processing of datapath frames and avoid the overhead of using the Linux kernel datapath. DPDK lacked support for features like NAT and Conntrack initially, but those were added later as that datapath matured.

A more recent addition to Open vSwitch was the TC Flower datapath. This was intended to be an alternative to the Linux kernel datapath that could be easily offloaded to hardware on devices that offer such support.

Another significant milestone for Open vSwitch was the adoption by OpenStack. Inclusion in OpenStack helped solidify Open vSwitch's place as a leader in the network virtualization space.

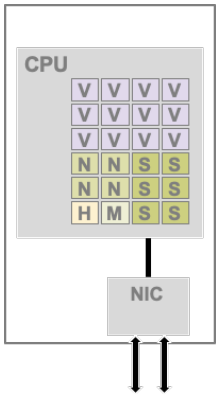
1.1 Resource Partitioning

When setting up a server to use as a hypervisor, an administrator often dedicates cores to specific tasks. This typically comes in the form of processor, memory, or storage reservation. Proper resource isolation reduces interference between independent applications and services running on the server. An important value proposition for datacenter operators is minimizing the resources dedicated to overhead functions and maximizing the resources available to applications. A typical server may have the following functions reserved or pinned to specific cores:

- VM Networking (N)
- VM Storage (S)
- Hypervisor Management (M)
- Host Networking (H)

The remainder of the cores on the system are available for VMs or Containers (V). For example, a breakdown of what this could look like on a 24 core system appears in Figure 1:

***Figure 1**: Hypervisor Core Allocation*



On this 24-core system only half of the cores are available to run customer workloads. This paper will examine the 4 cores being allocated to handle the network workload (N) generated by the applications (V) and whether there are cases where it is possible to use fewer server cores for network workloads based on hardware configuration or capabilities.

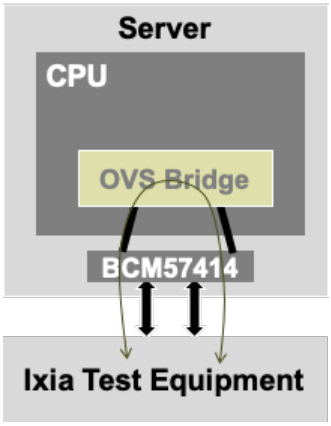
2. Open vSwitch Performance

Quantifying the performance of Open vSwitch can be a difficult task. Not only are there multiple datapaths that can be used for dataplane traffic, but there are a host of different configuration options that can alter the complexity of the

datapath and impact the performance in packets per second or in core utilization.

For these tests IPv4 UDP IMIX traffic (358 byte frames) was selected based on Broadcom customer deployment experience. To simplify test configuration, firewall and NAT features were not used. The goal was to understand the best-case performance for routing traffic by OVS between two ports.

***Figure 2**: Test Server Configuration*



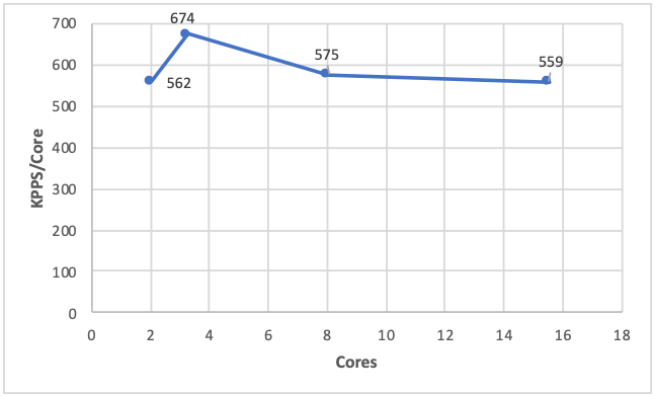
The server used was a Dell R730 with a dual socket Intel Xeon E5-2697v4@2.6GHz and a Broadcom BCM57414 (2x25Gbps) NIC. The system was tuned for maximum performance BIOS settings and IRQ affinity was set to the proper NUMA node for the NIC.

The software on the system was RHEL7.6 with 3.10.0-957.1.3.el7 and openvswitch-2.9.0-83.el7fdp.1.

2.1. Kernel Datapath

The first tests involved only the kernel datapath to determine the packet per second rate a single core could handle sending and receiving. Once this was known, the number of receive and transmit rings was altered to control how many receive/RSS queues would be handling inbound traffic. The number of receive queues was also used to dictate the number of cores used for forwarding traffic. Altering the number of queues available to the NIC provided the ability to determine whether per core performance scales linearly with the number of cores used.

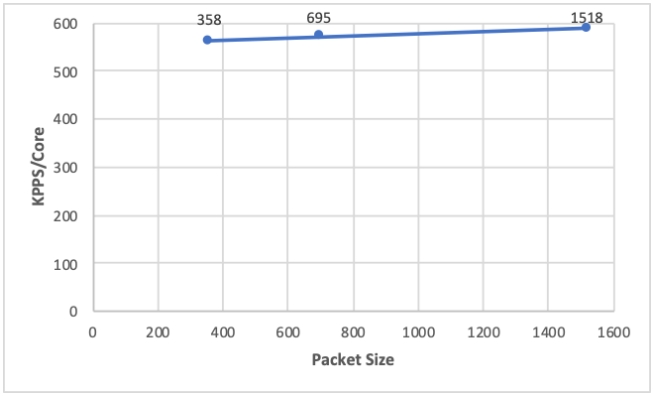
***Figure 3**: Received packets per second per core vs number of cores processing traffic*



The graph in Figure 3 shows a fairly flat line for the maximum number of packets per second each core can handle as the number of cores (and the traffic) scales up.

With a Xeon processor running at 2.6GHz, this graph tells us that we are able to handle 560kpps/core. This translates to 4600 cycles/packet or a packet time of 2µs. This high value of cycles per packet set an expectation that there would not be a significant difference in per core packet processing rate as the packet size scales towards MTU.

***Figure 4**: Received packets per second per core vs packet size of each packet*



Testing showed this hypothesis to be correct. This system can handle 560kpps/core across a variety of frame sizes and therefore consistently takes around 4600 cycles/packet.

These numbers provided a usable baseline for the performance in packets per second per core when using the OvS datapath available in the Linux kernel.

2.2. Userspace Datapath

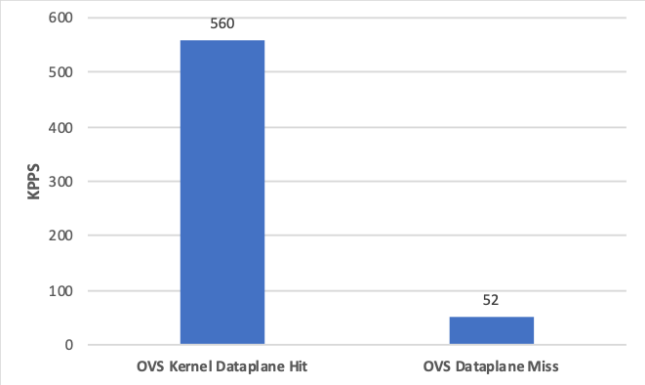
The life of a packet when there is a miss in the kernel datapath is significantly different. An *upcall* is made to `ovs-vswitchd` and the `skbuff` is passed to userspace. OpenFlow tables are queried to determine whether a matching entry exists for that flow. If so, a flow table entry is added to the kernel datapath and finally the `skbuff` is reinjected to the kernel for transmission.

Special care was taken when testing kernel datapath miss performance. Rather than simulating a DDoS attack by sending a new flow per packet, a duration of 8 packets per flow was selected to emulate short duration traffic streams. This configuration was chosen based on anaylsis of Broadcom customer data. In these tests, the first packet in a flow would get send to userspace but most of the packets had the chance to be forwarded by the kernel datapath.

Prior to testing, OpenFlow tables were populated with rules that would match the test traffic. Using a specific rate of new flows and a known Open vSwitch flow aging time meant that the approximate total flows in the system could be predicted. For example, with a flow add rate of 1000 flows per second and an aging interval of 10 seconds, the system should stabilize at around 10,000 active flows. Results showed that system behavior became unstable when

the rate of new flows exceeded more than a few tens of thousands of flows per second.

***Figure 5**: Received packets per second per core for kernel dataplane hit and kernel miss cases*



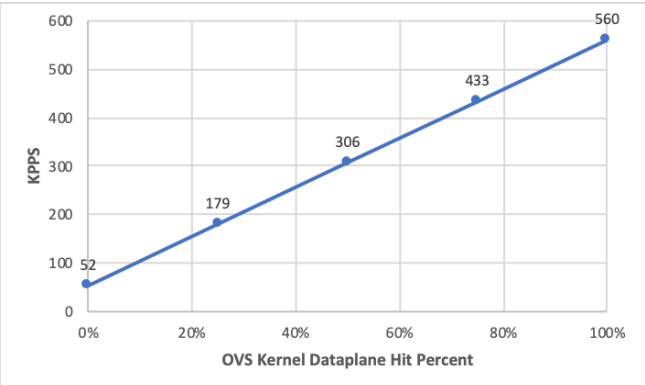
Testing showed that a kernel datapath miss reduces the datapath forwarding capacity to 52kpps. This was a 10.8x decrease in performance over the kernel datapath hit forwarding results. This showed that a kernel miss would cost roughly 50,000 cycles per packet or 20µs on the system used for testing.

2.3. Mix of Kernel and Userspace Datapath

Datacenter operators that offer application hosting services require both high performance packet forwarding as well as high flow setup and teardown rates. Typical system performance measurements focus on forwarding rates, but these tests also sought to measure the impact of a varying rates of flow churn.

Armed with the knowledge that a kernel datapath hit takes 4600 cycles and a kernel datapath miss takes 50,000 cycles, Figure 6 was created to chart the relationship between the percentage of traffic that was a hit in the kernel datapath vs a miss in the kernel datapath.

***Figure 6**: Received packets per second per core based on OVS kernel dataplane hit percentage.*



This chart highlights the dramatic impact that datapath misses have on single-core system throughput. At 75% hit rate for the kernel datapath there are only 13,000 new flows that are learned per second per core.

2.4 Application of Amdhal's Law

The implication of the extremely high cost of a datapath miss relative to that of a datapath hit is that system performance becomes limited by the higher cost workload. This is described by Amdhal's Law[2]. The basic idea is that if components of a program or application can be parallelized, then the maximum speedup can be calculated based on the workload that can be made parallel. Amdhal's Law is expressed as follows:

$$S_{latency}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

where

- $S_{latency}$ is the theoretical speedup of the execution of the task
- s is the speedup of the part of the task that benefits from improved system resources
- p is the proportion of the execution time that that original task occupied

This can be applied to serial programs as well when a component of a program can be optimized.

In the 75% kernel dataplane hit case, 433k total packets are processed. In that case, 420k are processed by the kernel datapath and 13k are processed in userspace by `ovs-vswitchd`. In this case $p = T_{hit} / (T_{hit} + T_{miss}) = 0.75$. If there is a 10x speedup in the performance of the kernel hit case, the total speedup would be:

$$S_{latency}(s) = \frac{1}{1 - 0.75 + \frac{0.75}{10}} = 3.1$$

This demonstrates that a 10x performance in datapath miss processing will only result in a 3x total throughput increase.

3. Options to Increase Performance

The first section of this document covered the fact that there is more than just a single datapath provider available for use with OvS. Both hardware acceleration via TC Flower offload and a PMD-based datapath were used to discover what gains were available there.

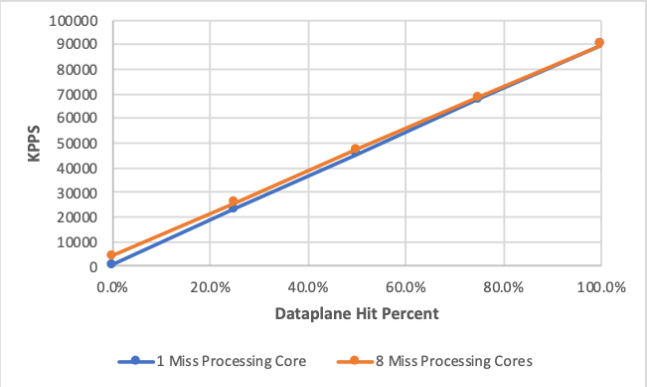
3.1. OvS with Hardware Acceleration

Hardware acceleration capabilities available when using the TC Flower datapath provide near-zero Xeon utilization when flows are completely offloaded to hardware. Unfortunately, the *upcalls* to userspace made when traffic does not match the hardware datapath are just as expensive as they are when the kernel datapath is missed.

Below is a graph of the throughput in packets per second when offloading known datapaths to hardware. It has an identical slope to the kernel datapath case, but the maximum number of packets per second that can be handled

is much greater than the kernel datapath. In this case the limitation in total packets for the datapath hit case is based on hardware capabilities not cores available for processing datapath hits.

***Figure 7**: Received packets per second per core based on hardware dataplane hit percentage.*



In addition to the blue line in Figure 7 (which shows the forwarding ability of a single core), an orange line was added to demonstrate the impact of improved userspace datapath lookup performance and assignment of more cores to the task. This orange line represents two changes to system performance:

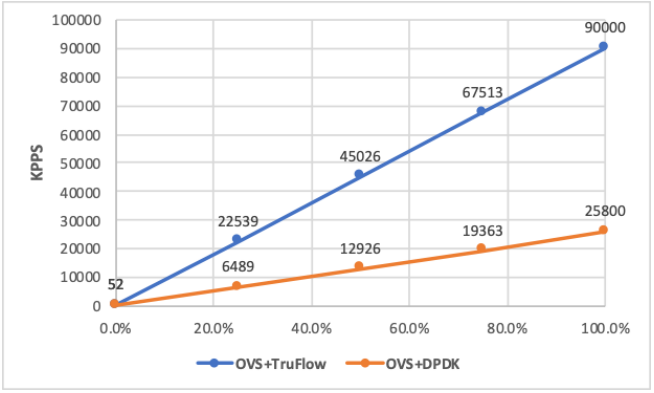
- Hypothetical 10x optimization of miss case processing
- Dedication of 8 server cores for datapath miss processing of hardware accelerated frames.

The fact that these two lines are so close to each other highlights the fact that the datapath miss lookup performance is the limiting factor for dataplane performance – even with an 80x improvement over single core miss processing.

3.2. OvS with DPDK

The next set of tests used DPDK as the datapath provider. These tests provide a comparison between OvS+DPDK forwarding results and OvS with hardware acceleration.

***Figure 8**: Received packets per second per core based on DPDK dataplane hit percentage.*



As shown in Figure 8, OvS + DPDK did not perform as well as the hardware accelerated dataplane. The interprocess communication, lookup and creation of the flow between `ovs-vswitchd` and a DPDK-based datapath provider is still expensive enough that that only 67kbpps can be processed per core. This was 29% performance increase over the lookup rate when using the kernel datapath (52k), but still not significant when the performance gap between hit and miss lookup is go expensive. An administrator would still need to allocate multiple Xeon cores to handle processing of packets for both the datapath hit and miss cases to maintains high throughput for both cases.

Without a way to offload the processing of misses to the kernel datapath, hardware datapath, or DPDK datapath, there is not going to be a way to demonstrably reduce the cost of running Open vSwitch on a server.

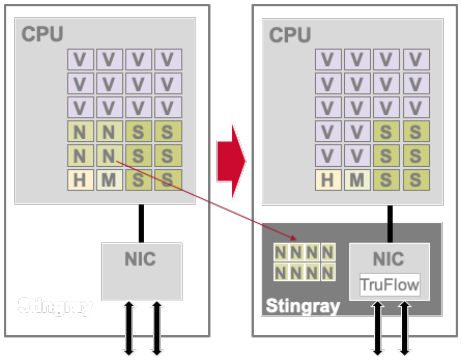
3.3. Full OvS Offload to Broadcom Stingray SmartNIC

The latest generation of *SmartNIC* products not only offer the option to accelerate the datapath for known flows, but also offer the option to offload datapath miss processing to the NIC. These SmartNICs possess datapath acceleration capabilities via the TC Flower datapath and have general purpose ARM cores capable of running applications that can process datapath misses like `ovs-vswitchd`.

In the case of the Broadcom Stingray SmartNIC this means the NIC itself has eight ARMv8 (A72) cores as well as storage for an operating system and memory that operate independently from the server. This *server in your server* runs Linux and has already demonstrated the ability to perform any of the tasks a server is capable of performing. This eliminates the need to dedicate Xeon cores for network workload on the server where the cards are installed.

The transition of the network workload (N) from the Xeon cores to the ARMv8 cores. A visualization of this change appears in Figure 9.

***Figure 9**: Hypervisor core allocation before and after moving OvS from Xeon cores to SmartNIC ARM cores.*



Not only are 4 cores now freed from the task of processing OvS traffic, but moving the workload to as many as 8 ARMv8 cores actually provides an increase in performance over 4 Xeon cores running the same workload. Since the Broadcom Stingray SmartNIC also runs a standard Linux kernel and tools, the hardware datapath can also be acceler-

ated by `ovs-switchd` running on the SmartNIC ARM cores.

Figure 11: Hypervisor core allocation before and after moving OvS from Xeon cores to SmartNIC ARM cores and then accelerating OvS dataplane to TruFlow hardware.

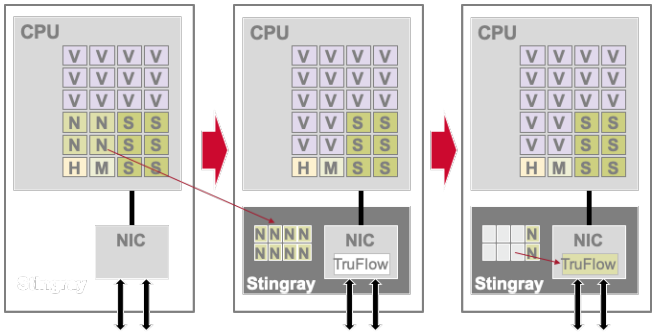
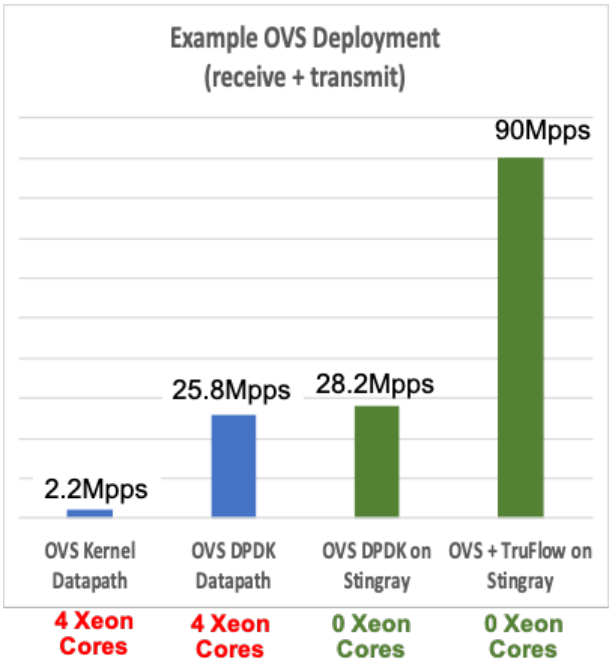


Figure 11 shows how hardware acceleration of the datapath could free the 8 ARM cores on the Stingray SmartNIC for doing nothing but processing datapath misses. The performance benefit of this case is the same as the benefit when accelerating the kernel datapath via TC flower on the Xeon cores, but now the Xeon cores are free from the burden of processing datapath misses and can be used for application workloads.

Figure 12: Comparison of OvS datapath performance in packets per second on Xeon Server with Stingray SmartNIC



The chart in Figure 12 shows (in blue) data presented in Figure 5 and Figure 8. This represents the throughput in packet per second for the OvS kernel datapath and OvS DPDK datapath in the 100% datpath hit case when dedicating 4 Xeon cores to the task of forwarding packets.

The chart shows (in green) the performance in packets per second when *zero Xeon cores* are used for any dataplane processing. In the first case the Stingray SmartNIC is running OvS with the DPDK datapath and the second graph shows a hardware accelerated datapath.

3. Conclusions

The performance of the ARMv8 processors combined with the high throughput between the ARM CPU subsystem and the networking interfaces has made this an attractive solution for full offload of OvS datapth. Doing this frees up Xeon server cores and pushes the load to the Broadcom Stingray SmartNIC.

Acknowledgments

Thanks to Karen Schramm, Cedell Alexander, and John Carney for support during this project.

References

1. "A complete list of Open vSwitch releases". http://www.openvswitch.org/releases/ . (Retrieved April 2, 2014)

2. “Amdahl’s Law” https://en.wikipedia.org/wiki/Amdahl%27s_law (retrieved 5 March 2019)