SmartSilc
Solutions by Silicom Ltd.

Red Rock Canyon Technology

Accelerates

KVM

June 2015

Under active packet processing scenario, a standard x86 based server is often set to perform a myriad of tasks to which it suits well, while different type of tasks, although doable and implemented, are far from the server's forté. Tasks that fit the most for general purpose x86 CPU are **compute** and **data processing** tasks, such as running applications and services, whether data bases, or mail and web servers, or even security applications.

On the other hand, packet switching and forwarding tasks, that include sub tasks such as lookup table operations, or memory copy (aggravated by QPI bus use in case of memory fetch across CPU sockets), cause in many cases considerable penalty in CPU cycles, bringing the system in general to underutilization. Let's take, for example, a case of a cache service that runs on the server, where multiple instances of it operate with affinity to specific CPU core. Each processing core is busy performing its business logic, but there are several cores that are busy doing packet forwarding and switching, be it on:

- Hypervisor, with its bridging software;
- DPDK with implemented forwarding engine;
- Standard kernel with layer 2 stack operating in kernel.

Since dual or quad CPU sockets servers are the de-facto standard platform, CPU core affinity for data path as well as data processing or data consumption, become an important aspect for best performance. For a given packet stream, it is best if same CPU core, or a core from the same CPU socket, will implement both data path and actual data processing.

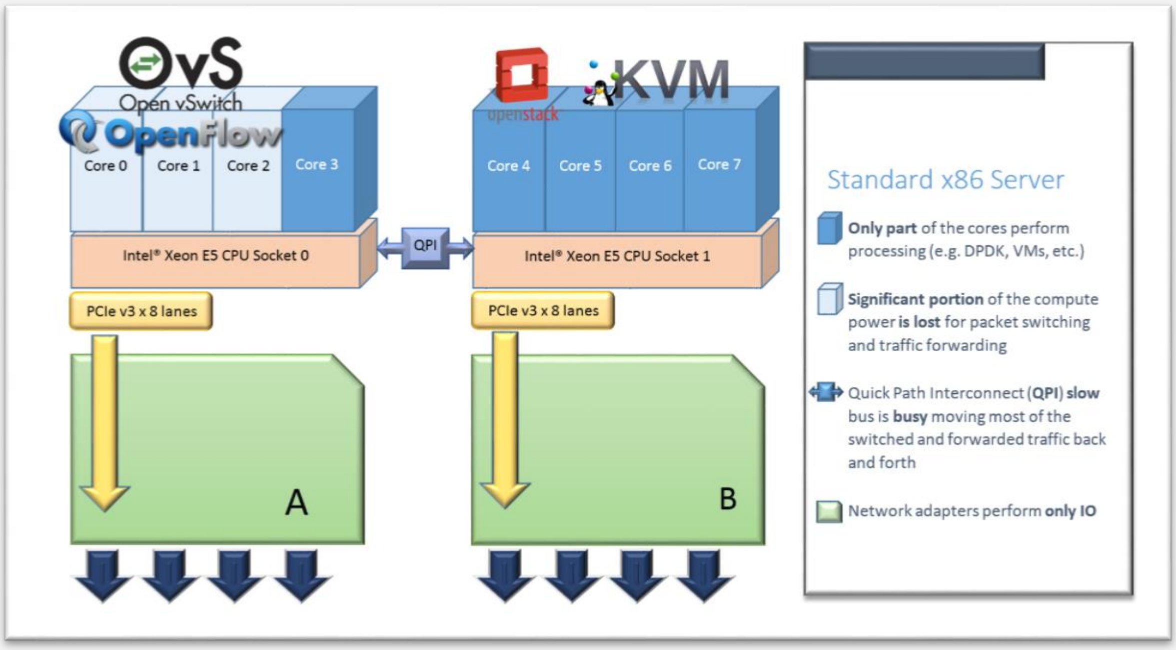
OvS
Open vSwitch
OpenFlow
Core 0
Core 1
Core 2
Core 3
openstack
KVM
Core 4
Core 5
Core 6
Core 7
QPI
Intel® Xeon E5 CPU Socket 0
Intel® Xeon E5 CPU Socket 1
PCIe v3 x 8 lanes
PCIe v3 x 8 lanes
A
B
Standard x86 Server
Only part of the cores perform processing (e.g. DPDK, VMs, etc.)
Significant portion of the compute power is lost for packet switching and traffic forwarding
Quick Path Interconnect (QPI) slow bus is busy moving most of the switched and forwarded traffic back and forth
Network adapters perform only IO

Figure 1 – Sustaining Performance Penalty

While it is quite straight forward to set CPU core affinity to an application (data processing), it is less easy to set same affinity for data path. For example, taking the case of the system that appear in Figure 1, it can be seen how cores 0 to 2 that are running packet forwarding mechanism (software switch), are required serve CPU cores on the other socket, in CPU1, thus hitting the QPI and sustaining degradation; let alone the fact that for a general purpose CPU, forwarding table lookup and memory copy are less efficient tasks; let alone the fact that a **software switch exhibit non-linear performance drop** as lookup table increase [1].

Silicom approaches this problem with a standard, yet true out of the box thinking solution:

- Two standard NICs operate as one (see Figure 2).
- Forwarding is offloaded to NICs (see Figure 3).

Chaining of NICS – By simply PCIe daisy chaining two NICs (each NIC is targeted to connect to different CPU socket), Silicom created a standard yet powerful detour to QPI bus, at least for network traffic.

Forwarding offload – Having a powerful switching silicon to perform the data path forwarding, switching and filtering, brings an enormous value to the system as a whole, while freeing the formerly busy CPU cores, to perform even more compute task, which fit them best.

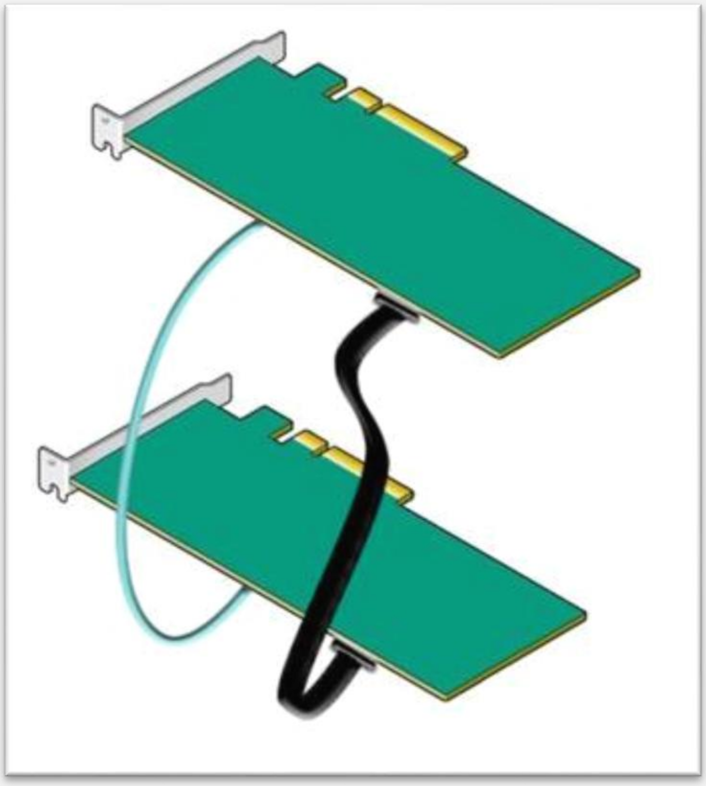

Figure 2 – Daisy Chaining Two Standard NICs

In KVM environment, the above scheme may bring added value almost instantly. OVS offload through its `netdev` interface is quickly achievable, and instantly KVM is relieved of forwarding tasks.

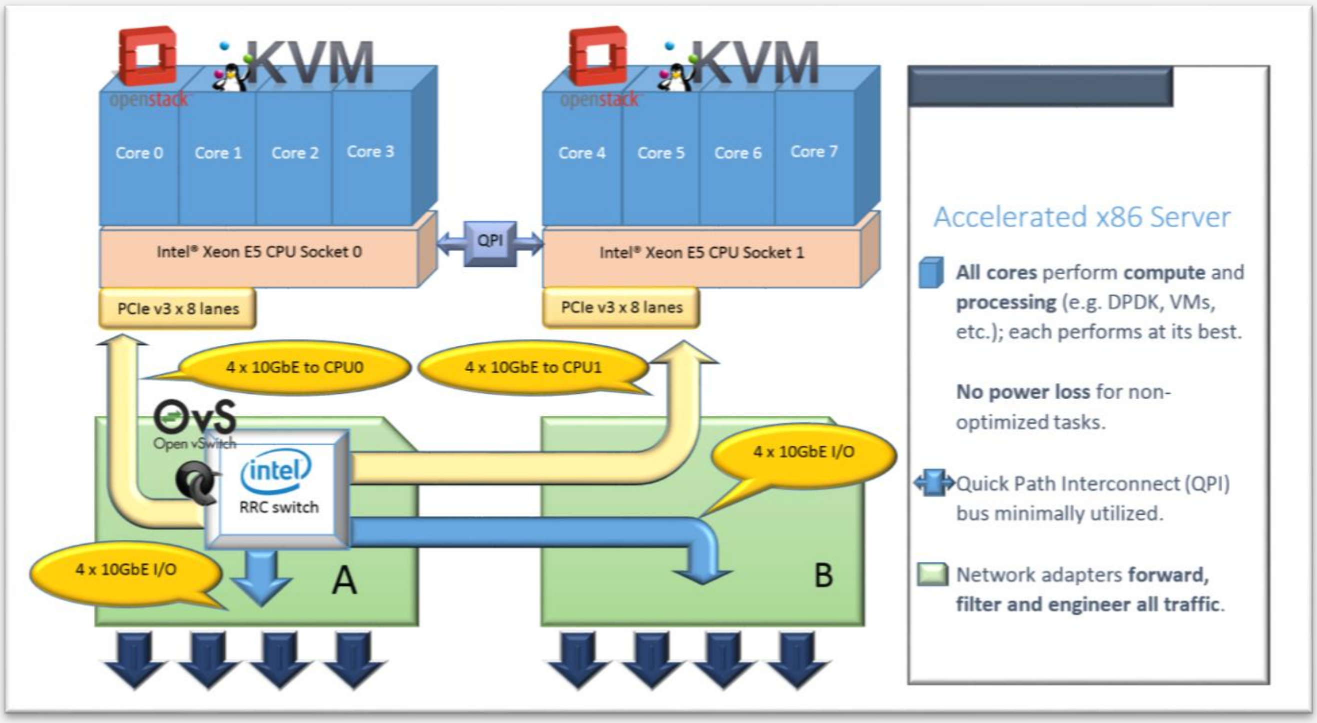
KVM
openstack
Core 0
Core 1
Core 2
Core 3
KVM
openstack
Core 4
Core 5
Core 6
Core 7
Intel® Xeon E5 CPU Socket 0
QPI
Intel® Xeon E5 CPU Socket 1
PCIe v3 x 8 lanes
PCIe v3 x 8 lanes
4 x 10GbE to CPU0
4 x 10GbE to CPU1
OvS Open vSwitch
intel RRC switch
4 x 10GbE I/O
4 x 10GbE I/O
A
B
Accelerated x86 Server
All cores perform compute and processing (e.g. DPDK, VMs, etc.); each performs at its best.
No power loss for non-optimized tasks.
Quick Path Interconnect (QPI) bus minimally utilized.
Network adapters forward, filter and engineer all traffic.

Figure 3 – Intel® RRC10420 Full Forwarding Offload and Acceleration.

The math is simple. If a hefty 25% of CPUs cores in a typical use case are busy forwarding traffic (not the best suited task for a general purpose core) for the other 75% busy processing cores, then adding these underutilized cores to the processing cores pool will automatically add 33% more power to the system; giving even more optimization options with advanced filtering capabilities of the switching silicon itself.

REFERENCES

[1] Removing Roadblock from SDN: OpenFlow Software Switch Performance on Intel DPDK,

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6680560&tag=1