Enabling NFV to Deliver on its Promise

Intel® Xeon® processor E5-2600 v3 product family has special features to speed up packet processing in virtualized environments.

"...the industry is beginning to embrace NFV, as seen by the availability of more interoperable solutions using software-based network functions that are decoupled from hardware through virtualization."

Executive Summary

Network functions virtualization (NFV) is paving the way for more open, flexible, and economical networking and communications equipment based on general-purpose computing platforms instead of traditional proprietary, purpose-built products. However, this transition hinges on increasing performance and minimizing the latencies associated with virtualization on general-purpose platforms, particularly for interrupt-intensive, packet processing workloads. Designed for such demanding networking applications, platforms based on the Intel® Xeon® processor E5-2600 v3 product family incorporate several technologies that can significantly reduce virtualization latency or its impact.

This paper reviews memory management features and inter-VM (virtual machine) communication schemes that can be applied to dramatically improve performance and determinism for packet processing workloads in virtualized environments.

Table of Contents

NFV Benefits

Service providers want the ability to deploy new services in hours, not weeks or months. But standing in the way are today's networks that are built with proprietary, fixed-function appliances, requiring equipment vendors to be closely involved in new services creation. Vendors must typically develop, test, and integrate the necessary software on their equipment before a service provider can purchase and install the boxes – in all, a costly and time-consuming process.

In response, the industry is beginning to embrace NFV, as seen by the availability of more interoperable solutions using software-based network functions that are decoupled from hardware through virtualization. European Telecommunications Standards Institute's (ETSI) NFV Industry Specification Group (ISG) is chartered with standardizing architecture, framework, and required protocol specifications in this area. Over the last couple of years, NFV ISG has made significant progress and delivered architectural foundational work.1,2 A major benefit is network functions are no longer tied to a particular hardware platform, allowing them to be controlled centrally and deployed dynamically throughout the network as needed. The end result is new service deployment that can be as easy as uploading software to an existing networked server – taking just minutes or hours.

To make this happen, many equipment vendors are moving away from using different architectures per major workload (application, control/data plane, and signal processing) to running all these workloads on Intel® architecture with the Data Plane Development Kit (DPDK). Consolidating these workloads onto a scalable and simplified platform makes it easier to implement multi-function and multi-vendor solutions, such as the service chaining example in Figure 1. In this usage model, a next-generation firewall, service appliance, and data plane applications (e.g., forwarding engine) share packet data on a single platform.

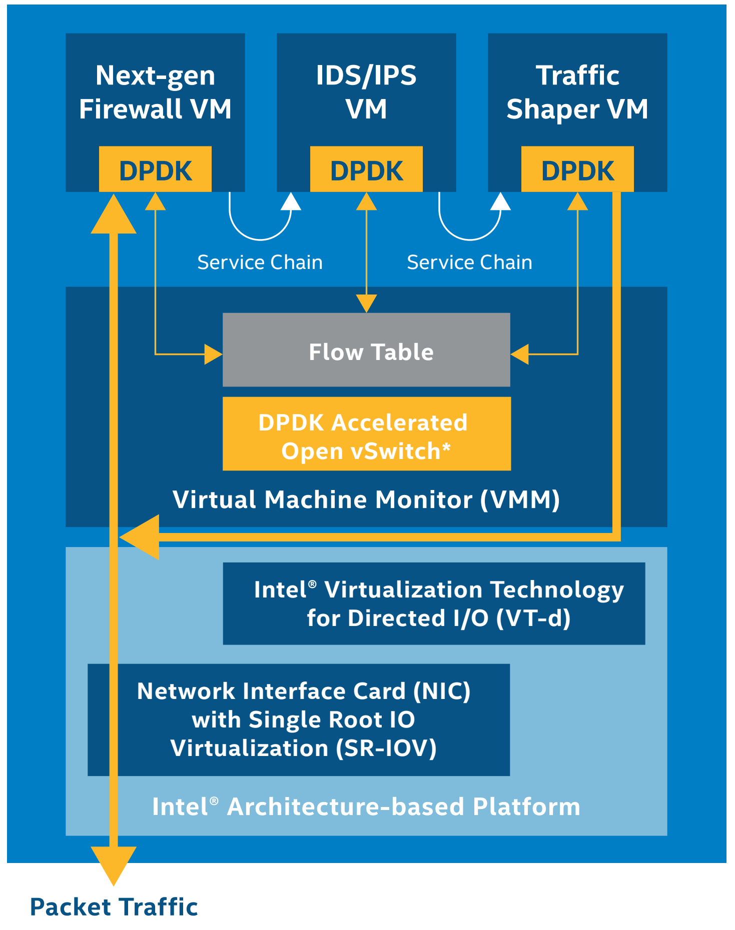

Next-gen Firewall VM
DPDK
IDS/IPS VM
DPDK
Traffic Shaper VM
DPDK
Service Chain
Service Chain
Flow Table
DPDK Accelerated Open vSwitch*
Virtual Machine Monitor (VMM)
Intel® Virtualization Technology for Directed I/O (VT-d)
Network Interface Card (NIC) with Single Root IO Virtualization (SR-IOV)
Intel® Architecture-based Platform
Packet Traffic

Figure 1.
Service Chaining Example.

The following sections describe special features that speed up packet processing in virtualized environments running on the Intel® Xeon® processor E5-2600 v3 product family.

NFV Challenges

In an NFV context, the virtualization employs a virtual machine monitor (or hypervisor) to create an additional abstraction layer between the physical hardware platform and operator network applications. Although this abstraction enables the benefits previously discussed, it also adds virtualization overheads occurring in cache, I/O, and memory, and makes it more difficult to share data between applications. As a result, developers face a variety of challenges, including:

- **Performance**
  Virtualization inherently introduces overheads (e.g., VM exits) that lead to some level of application performance degradation compared to a non-virtualized environment. The primary source of overhead is the sheer amount of additional operating environment code needed for tasks such as memory address translations performed when the CPU switches between VMs.
- **Service Chaining**
  Fundamentally, NFV simplifies the consolidation of virtual networking functions (VNFs), such as firewall, address translator, application delivery controller, and WAN optimizer, onto a single computing platform. Since each VNF performs a task on the network traffic, it is critical to have an efficient way for VNFs to send and receive packets from one another, which is called service chaining. Although there are a handful of standardization exercises in motion, to date, the industry has not ratified a standard way to chain VNFs.
- **Reliability**
  CPU cache plays a pivotal role in delivering optimum performance in the modern networking system. VNFs must share this precious resource, and if any one of them is allowed to use more than its fair share of cache, the latency, jitter, and performance of the entire system may become unpredictable. This could make the system unsuitable for time-critical workloads such as processing SIP-based voice calls.

Memory Management Features and Inter-VM Communication Schemes

The following sections describe how a variety of state-of-the-art Intel technologies can help the industry overcome implementation challenges in NFV:

A. Huge Pages

The Intel® Xeon® Processor E5-2600 v3 product family allows a virtual machine monitor (VMM) to assign larger memory pages to VMs, which helps to greatly minimize overhead. Using Linux* huge pages through the "hugetlbfs" filesystem, 2 MB and 1 GB pages are supported, in addition to the previously available 4 KB page size. Support for huge pages was first introduced in the prior generation Intel® Xeon® Processor E5-2600 v2 product family.

Memory Management Basics
Applications running in virtual machines (VM) do not have direct access to system memory for a number of reasons, one of which is to prevent them from accessing memory outside their intended domain. Instead, memory management units (MMUs) police memory requests by determining whether access is permitted and translating virtual addresses to host physical addresses. The MMU performs address translations for the CPU, while the input/output MMU (IOMMU) translates virtual addresses to physical addresses for I/O devices that send packet data to VMs, as shown in Figure 2.

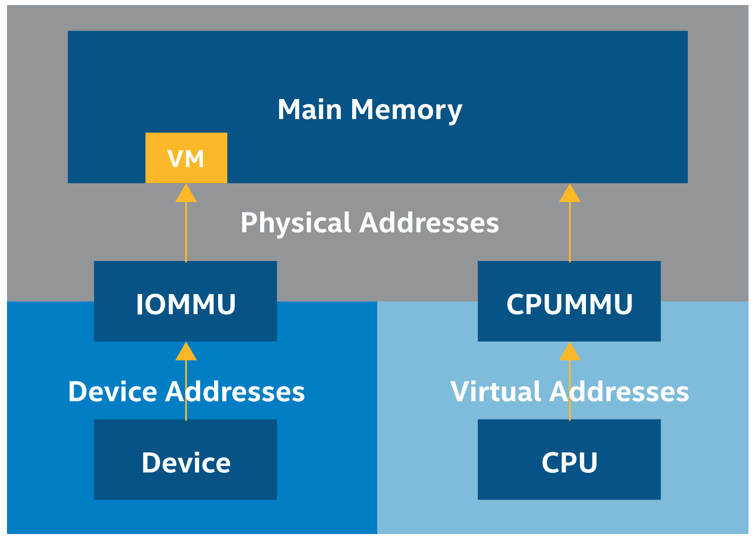


Figure 2.
Simplified IOMMU Representation[3]
Source: wikipedia.org

The IOMMU plays a large role in the performance of I/O-intensive networking applications. To greatly speed up memory accesses, it has a dedicated memory management cache, called the input/output translation look-aside buffer (IOTLB). The IOTLB caches frequently used address translations, including memory locations assigned to VMs. When a requested address is not in the IOTLB, a IOTLB-miss occurs, which triggers a high-latency process called a page walk. In this case, the processor searches through a large page table to determine the physical address associated with the virtual address, thereby incurring a large delay.

Benefits from Huge Tables

The IOTLB is used by Intel® Virtualization Technology for Directed I/O (Intel® VT-d) to deliver data from I/O device to VMs, thus it plays a key role in NFV solutions. Intel VT-d now supports 2 MB and 1 GB pages, where a 2 MB page is 512 times larger than a 4 KB page, thus incurring up to 512 fewer page cross penalties. Earlier Intel® processors only support 4 KB pages.

The address structures for 4KB and 2 MB pages are shown in Figure 3, illustrating 2 MB pages require one less memory access to get to the page data compares to 4 KB pages. Other benefits include:• With larger page sizes, a VM requires fewer entries in IOTLB, and therefore takes up less space in the IOTLB cache.

• In the event of an IOTLB miss, page walks are shorter since the page table has fewer address entries, thereby improving performance.

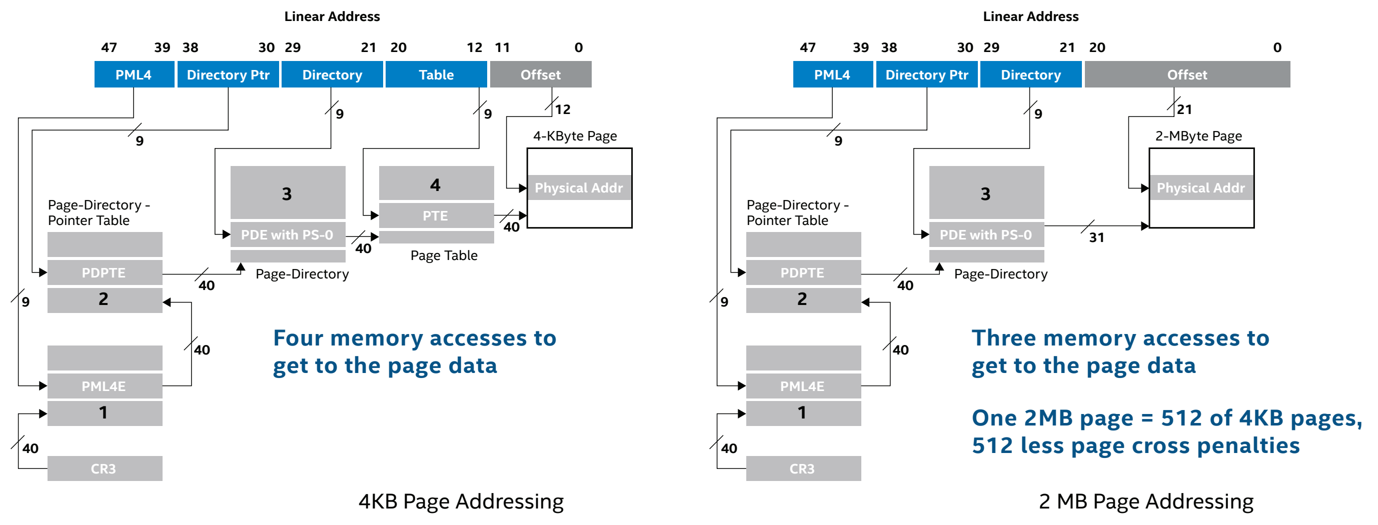

Linear Address
PML4
Directory Ptr
Directory
Table
Offset
4-KByte Page
Physical Addr
Page-Directory - Pointer Table
PDPTE
PDE with PS-0
Page-Directory
PTE
Page Table
PML4E
CR3
Four memory accesses to get to the page data
4KB Page Addressing
2-MByte Page
Three memory accesses to get to the page data
One 2MB page = 512 of 4KB pages, 512 less page cross penalties
2 MB Page Addressing

Figure 3.
Address Structures for 4KB and 2 MB Pages.

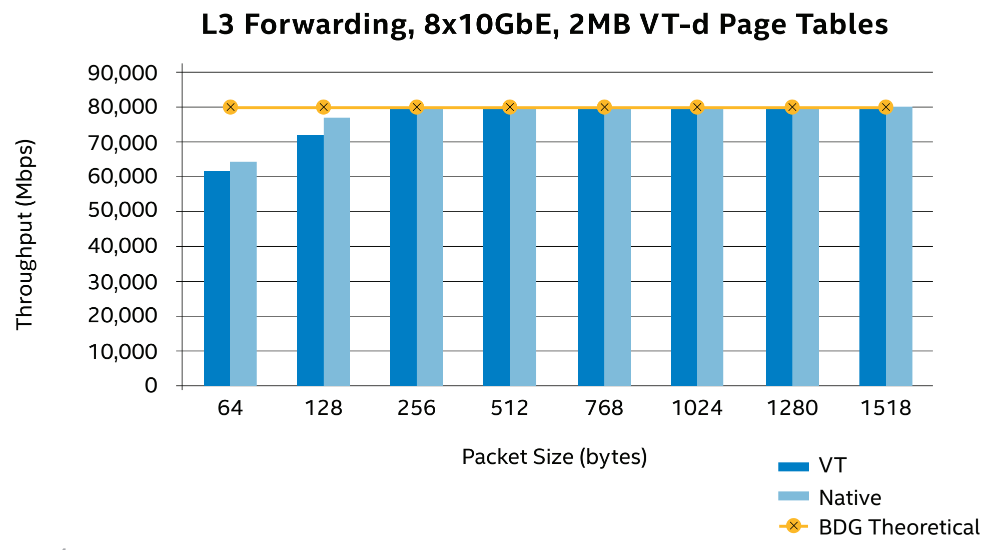


Figure 4.
L3 Forwarding Performance with Huge Pages.

Near-Native Performance[4]

Figure 4 compares the L3 packet forwarding throughput of the DPDK in virtualized and native (i.e. non-virtualized) environments with 2 MB page sizes achieved on Intel® Xeon® processor E5-2658 v2-based server platforms. For 256 byte packet size onwards, an 80G line-rate was achieved in both native and virtualized environments. For 64 and 128 bytes, the native performance had slightly higher throughput. Overall, the use of huge tables helps the Intel® platform deliver near-native packet forwarding performance in a virtualized environment.

B. Inter-VM Communication Schemes

Virtualization is being used to run software-based network functions in VMs on general purpose servers, thereby enabling a new breed of multi-function and multi-vendor solutions, as shown in Figure 1. These network functions typically collaborate with each other, necessitating a low latency inter-VM communication scheme that overcomes typical performance overheads. For instance, standard VM enter and exit overhead is considerable, so allowing the VMM to schedule the correspondences between VMs can result in significant latency penalties.

This section discusses three inter-VM communication schemes and their associated tradeoffs.

The schemes are mapped to an Intel® Xeon® processor E5-2600 v3 platform in Figure 5.

1. SR-IOV and Layer 2 Switch Inside the NIC

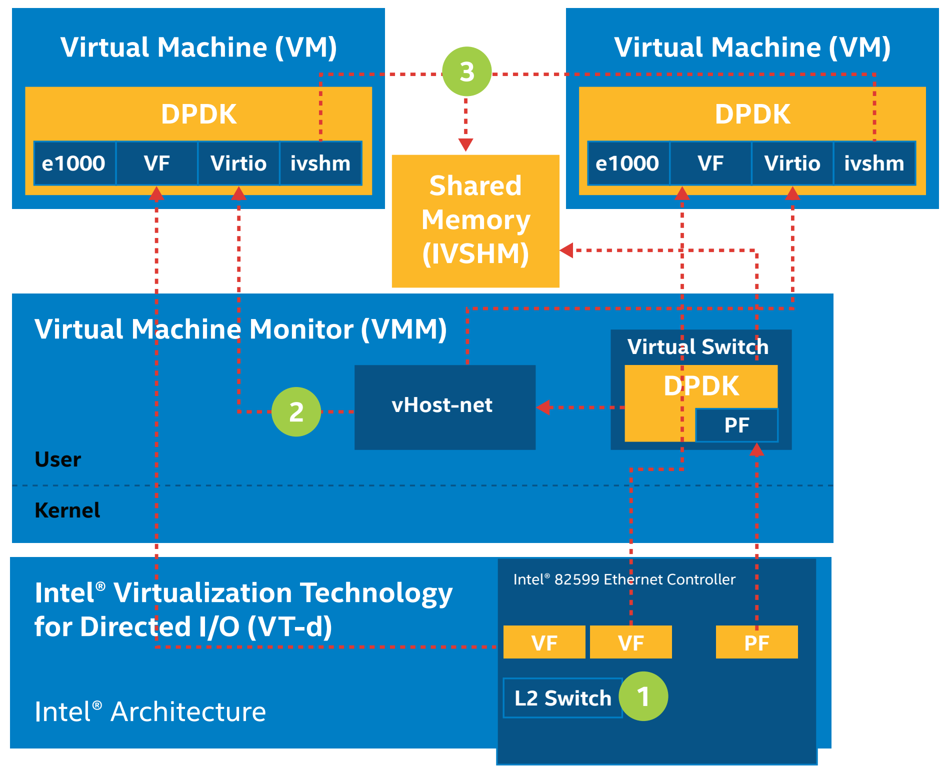
The PCI-SIG developed the Single Root I/O Virtualization and Sharing (SR-IOV) specification to define a standard approach to creating and managing native-shared I/O devices in a virtualized environment. The objective was to enable data movement without VMM involvement, thus minimizing overhead. The specification provides a mechanism, called virtual function (VF), that is implemented in physical network interface cards (NICs) and other I/O devices for transferring data either between VMs, as shown in Path 1 of Figure 5, or to/from the network, which is discussed in more detail in the following text.

SR-IOV Basics

SR-IOV enables multiple VMs, system images, and guests to directly access subset portions of physical I/O resources for performance data movement and to natively share underlying hardware resources. An SR-IOV device presents physical functions (PFs), which are standard PCI Express* (PCIe*) functions. Each PF can have multiple VFs, which are "light-weight" PCIe functions with enough resources for major data movement, as well as unique requester identifiers (RIDs) to index the IOMMU page table for address translation.

SR-IOV extends this by enabling multiple direct communication channels for each physical I/O port on the device. The method employed by SR-IOV provides unique memory space,

Figure 5.
Three Inter-VM Communication Schemes.



work queues, interrupts, and command processing for each interface exposed while utilizing common shared resources behind the host interface. The SR-IOV specification defines a standardized mechanism to create natively shared devices. This also provides for security to be maintained between VMs, as each memory domain is isolated.

Solution Tradeoffs

PRO: Since NICs (and not the VMM) are used for switching, amount of CPU required for implementing switching function is saved.

CON: The PCI bus, which connects the CPU to the Layer 2 switch in the NIC, is used to pass packets between VMs. This approach can stress PCI bandwidth, weighing in on overall system performance for long VM service chains.

2. Virtual Switch Support

A virtual switch (vSwitch) is logical switching software built into a VMM to network VMs together. One version is called DPDK Accelerated Open vSwitch, an open source solution expressly developed for inter-VM communication and dramatically sped up by the DPDK from Intel.

Open vSwitch Basics[5]

Open vSwitch can operate both as a soft switch running within the VMM and as the control stack for switching silicon. It has been ported to multiple virtualization platforms and switching chipsets. It is the default switch in XenServer 6.0, the Xen Cloud Platform*, and also supports Xen*, KVM*, Proxmox VE*, and VirtualBox*. It has been integrated into many virtual management systems including OpenStack*, openQRM*, OpenNebula*, and oVirt*. The kernel datapath is distributed with Linux*, and packages are available for Ubuntu*, Debian*, and Fedora*. Open vSwitch* is also supported on FreeBSD* and NetBSD*. The Open vSwitch release in development has been ported to the DPDK.

Virtual Switch Configuration

Path 2 in Figure 5 shows two VMs running on a VMM with the DPDK Accelerated Open vSwitch. The virtio Poll Mode Driver (PMD) inside the VM DPDK instance, coupled with user space vhost back-end for vSwitch, allows VMs to send and receive packets between themselves.

Solution Tradeoffs

PRO: This open source software solution is highly cost-effective and supports live VM migration.

CON: Since Open vSwitch is implemented in the VMM, there is a higher load on the VMM and additional inter-VM communications latency, as compared to SR-IOV. However, throughput can be significantly improved with the DPDK Accelerates Open VSwitch, which uses optimizations such as huge pages and software pre-fetch techniques.

3.Shared Memory Mechanism

There are a number of mechanisms that enable VMs to share a portion of host memory, as shown in Path 3 of Figure 5, and by doing so, deliver very high performance. However, security implications must be carefully evaluated since these mechanisms essentially provide a window into host processes, making them unsuitable for untrusted guests. Moreover, developers need to implement safeguards against data corruption (e.g., stale data), given the possible negative impact on the host and all the VMs sharing that particular memory.

Nahanni/IVSHMEM and KVM

The kernel-based virtual machine, commonly called KVM, is a full virtualization solution that supports multiple VMs running unmodified Linux or Windows* images on x86 hardware. A KVM-based memory sharing mechanism, called Nahanni/IVSHMEM, was developed by Cam Macdonell, Xiaodi Ke, Adam Wolfe Gordon, and Paul Lu from the University of Alberta.[6] It provides a user-level library for data movement and creates a virtual PCI device in QEMU that accesses memory on the host, as illustrated in Figure 6.

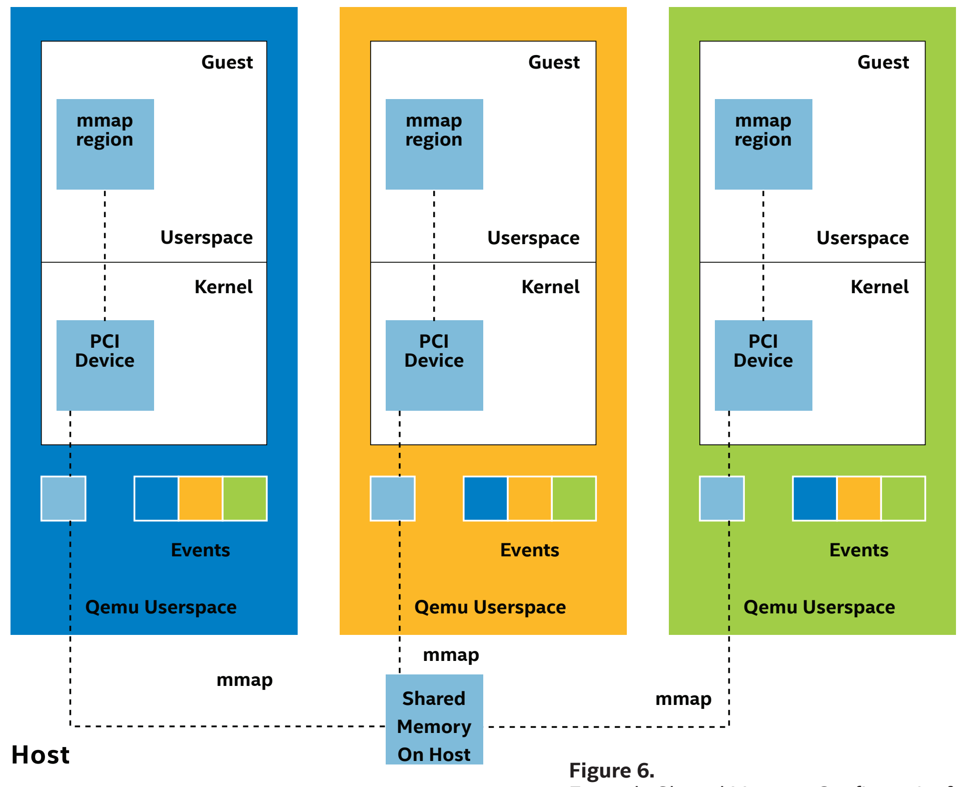
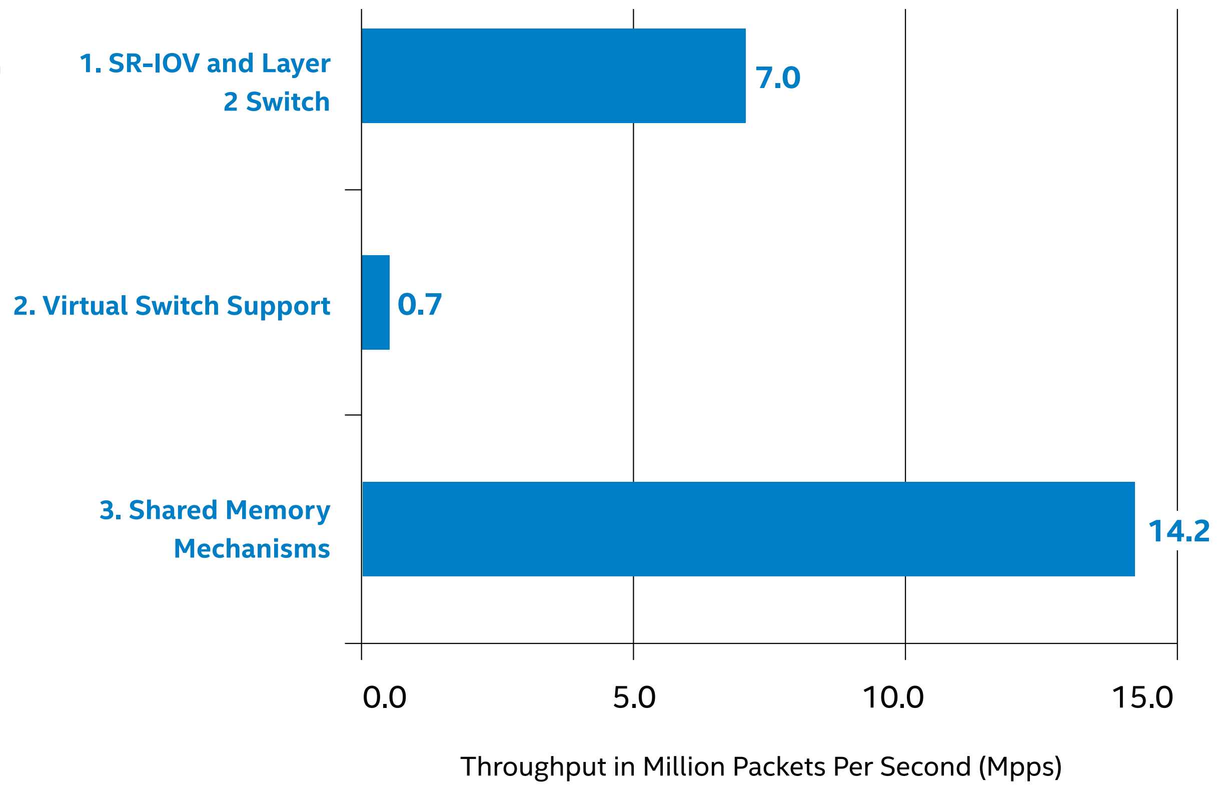


Figure 6.
Example Shared Memory Configuration[2].

Figure 7.
Performance of Three Inter-VM Communication Schemes.



Grant Table and Xen*

Utilizing the Xen hypervisor, grant tables provide a generic mechanism to memory sharing between domains. Each domain has a dedicated grant table, which is a data structure that enables the domain to tell Xen what kind of permissions other domains have on its pages. Entries in the grant table are identified by grant references, which are integer indexes into the grant table that allow a grantee to perform operations on the granter's memory.[7]

Solution Tradeoffs

PRO: This is a very high performance solution since VMs communicate over shared memory using mechanisms with negligible overhead.

CON: The VMs need to be part of a trusted environment and employ mechanisms (e.g., handshaking or spinlocks) to maintain data integrity and coherency.

Intel measured the performance of these schemes using standard benchmarking methods to generate 64-byte traffic and measure the throughput through two VMs, which is shown in Figure 7. Shared memory mechanisms delivered the highest performance at approximately 14.2 million packets per second (Mpps), which was over twice as fast as SR-IOV virtual/physical function support and twenty times faster than virtual switch support.[4]

C. Cache Monitoring and Allocation

The performance of most CPUs is highly dependent upon the availability of data and instructions to the execution unit. To lower data latency, the execution unit is surrounded by small pieces of high speed static RAM (SRAM) known as cache memory. This approach minimizes the need for the CPU to fetch data from high-latency system memory, avoiding substantial delays.

The Intel® Xeon® Processor E5-2600 v3 product family implements a large shared last level (L3) cache (Figure 8), which improves the performance of VMs running on the processor. However, when VMs contend for L3 cache space,

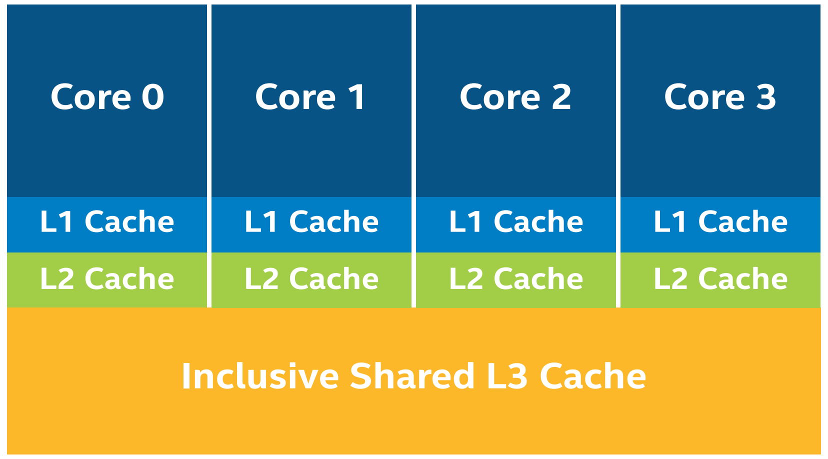


Figure 8.
Shared L3 Cache.

Intel Cache Monitoring Technology	Intel Cache Allocation Technology
All Intel® Xeon® processor E5-2600 v3 product family members	· Intel® Xeon® processor E5-2658 v3 · Intel® Xeon® processor E5-2648L v3 · Intel® Xeon® processor E5-2628L v3 · Intel® Xeon® processor E5-2618L v3 · Intel® Xeon® processor E5-2608L v3

Table 1.
Shared v Cache.

there could be a significant drop in performance and determinism. To avoid this situation, developers using select processor SKUs (see Table 1) can take advantage of cache monitoring and allocation features that are described in the following.

Intel Cache Monitoring Technology

This technology allows an operating system or virtual machine monitor (VMM) to determine how much L3 cache each software thread (application) is using. This is valuable information because it identifies which applications are consuming large amounts of L3 cache and potentially degrading the performance of other applications. For example, Figure 9 shows a low priority application (orange) that is using a lot of cache, making less cache available to speed up a higher priority application (green). In the virtualized environment, where multiple VMs would be sharing this L3 cache, such "Noisy Neighbor" behavior can adversely affect the performance of other VMs, bringing down overall system performance.

There are several ways to take advantage of cache monitoring to optimize the overall system performance:

1. Move L3 cache-intensive applications to another socket (processor).
2. Schedule L3 cache-intensive applications when time-critical applications are not running to optimize the amount of shared cache available to each application at any given time.
3. Generate performance histories to correlate available cache space and application performance.

This information can be used to implement cache-aware scheduling to ensure applications have the necessary cache available to meet performance targets.

Intel Cache Allocation Technology

After measuring L3 cache usage at the application level with Intel Cache Monitoring Technology, developers can use Intel Cache Allocation Technology to intelligently partition the L3 cache. This is shown in Figure 10, where a low priority application (orange) is assigned a relatively small amount of L3 cache, giving a higher priority application (green) access to more cache.

Developers can use cache allocation technology to increase determinism by prioritizing L3 cache access:

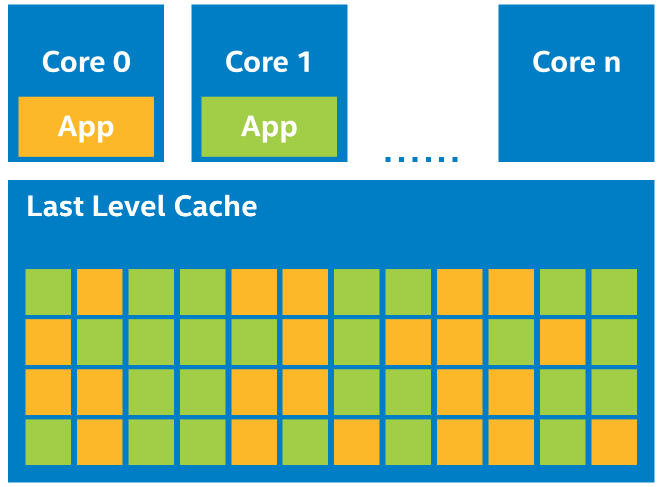


Figure 9.
Intel Cache Monitoring Technology Measures Cache Usage.

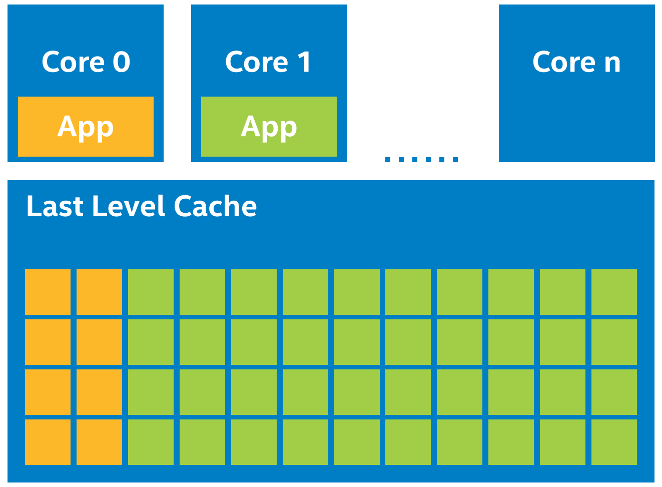


Figure 10.
Intel Cache Allocation Technology Assigns Cache Partitions to Applications.

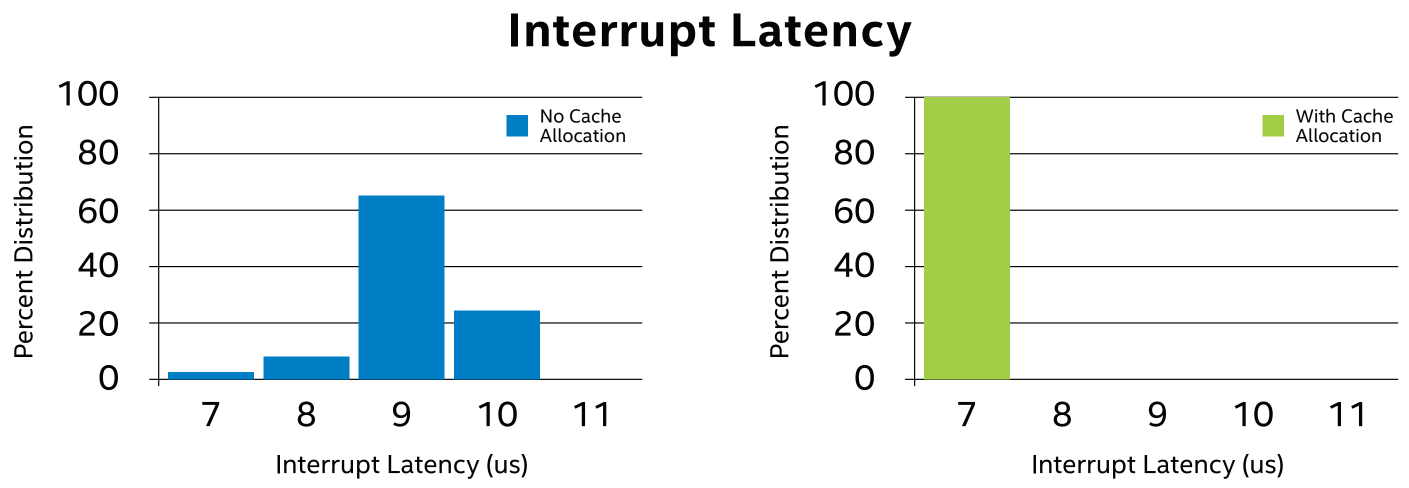


Figure 11.
Intel Cache Allocation Technology Improves Interrupt Latency.

1. Assign high-priority applications enough dedicated L3 cache to avoid having their data and instructions evicted by other applications.
2. Isolate low-priority applications by limiting their access to L3 cache.
3. Avoid unnecessary cache evictions that reduce performance.

In a study by Wind River*, Intel Cache Allocation Technology dramatically improved interrupt determinism, as seen in Figure 11. The left side shows interrupt latency without the technology ranged from 7 to 10 microseconds, but with the technology, the right side shows the interrupt latency for all samples was approximately 7 microseconds.[3]

The benefits of Intel Cache Allocation Technology can also be seen in a virtualized packet processing application. The sample application used in this example is the QoS sample application included in DPDK. The QoS sample application implements a basic packet processing pipeline consisting of a packet classification and scheduling stage, which results in the selection of a high priority or a low priority queue. In this 2 x 10 Gbps port configuration, the platform delivers 11 million packets per second (Mpps) of 64 byte packet throughput as depicted in the leftmost pane of Figure 12.

When an aggressor VM is introduced (middle pane), which is a "Noisy Neighbor" because it hogs substantial L3 cache, DPDK QoS sample application performance drops to 4 Mpps. The rightmost pane depicts application of cache allocation, thereby limiting the aggressor VM's access to L3 cache, after which DPDK QoS sample application performance goes back to the original 11 Mpps.[4]

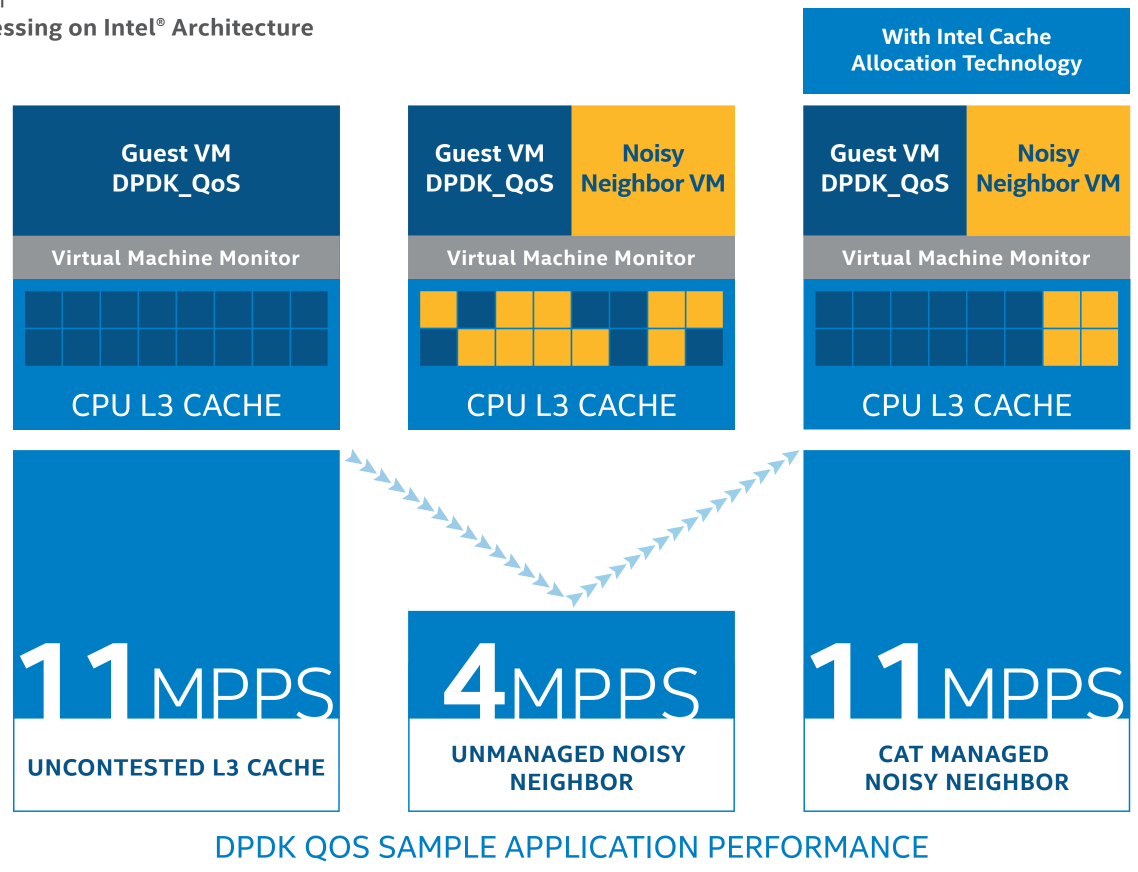


Figure 12.
Packet Processing with Intel Cache Allocation Technology.

Summary

Network functions virtualization (NFV) is having a profound impact on the design and deployment of next-generation networking and communications equipment. Fundamental to this transition is the ability of general-purpose processors to deliver high packet processing performance in a virtualized environment. Addressing this challenge, Intel designed Intel® Xeon® Processor E5-2600 v3 product family with technologies that can significantly reduce virtualization latency or its impact, thus making it particularly well-suited for demanding networking applications.

For more information about Intel solutions for NFV, visit **http://www.intel.com/content/www/us/en/communications/network-infrastructure-products-and-technologies.html**.

[1] Source: ETSI, "Network Functions Virtualizaion (NFV) Architectural Framework," http://docbox.etsi.org/ISG/NFV/Open/Published/gs_NFV002v010201p%20-%20Architectural%20Framework.pdf.
[2] Source: ETSI NFV ISG, http://docbox.etsi.org/ISG/NFV/Open/Published.
[3] Source: http://en.wikipedia.org/wiki/IOMMU.
[4] Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit Intel Performance Benchmark Limitations.
[5] Source: http://openvswitch.org.
[6] Source: Cam Macdonell, "Nahanni – a shared memory interface for KVM," slide 6, www.linux-kvm.org/wiki/images/e/e8/0.11.Nahanni-CamMacdonell.pdf.
[7] Source: http://wiki.xen.org/wiki/Grant_Table.