Case Study

VNF Benchmarking

spirent™
Promise. Assured.

Customer Profile

Customer: Large Network Equipment Manufacturer

Industry: Networking Equipment

Employees: 180,000 (2016)

The Challenges

How to test the performance of VNFs?

How to isolate infrastructure bottlenecks?

How to ensure performance in the context of a common, shared infrastructure?

The Move to Virtualization

Network functionality is moving to virtualized deployments, with some Tier-1 carriers already claiming to be over one-third virtualized. Formerly, network functions were implemented as dedicated appliances or chassis-based devices which were essentially self-contained and had clear boundaries at the physical interface. In such a context, performance problems could be more easily characterized and isolated. With virtualization, however, network functionality is embodied in virtual network functions, or VNFs, which can be deployed on top of a common network infrastructure. Virtualization adds value by optimizing the use of resources via on demand capacity management, but it also increases risk, since resources are drawn from a shared environment. Consequently, VNFs must be benchmarked differently from dedicated devices.

In this case study, we:

- Explain the NFV ecosystem
- Specify how VNFs should be benchmarked differently from dedicated devices
- Introduce the concept of “white box testing”
- Provide a concrete example, including takeaway insights

NFV Ecosystem

At a very high level, ETSI has defined the major components of NFV ecosystems, and standardized interfaces between those components - see Figure 1 below. Since the ETSI architecture has been described at length in other documents we will only give a brief review here. Please see Figure 1 and glossary for a quick review of the ETSI components and interfaces.

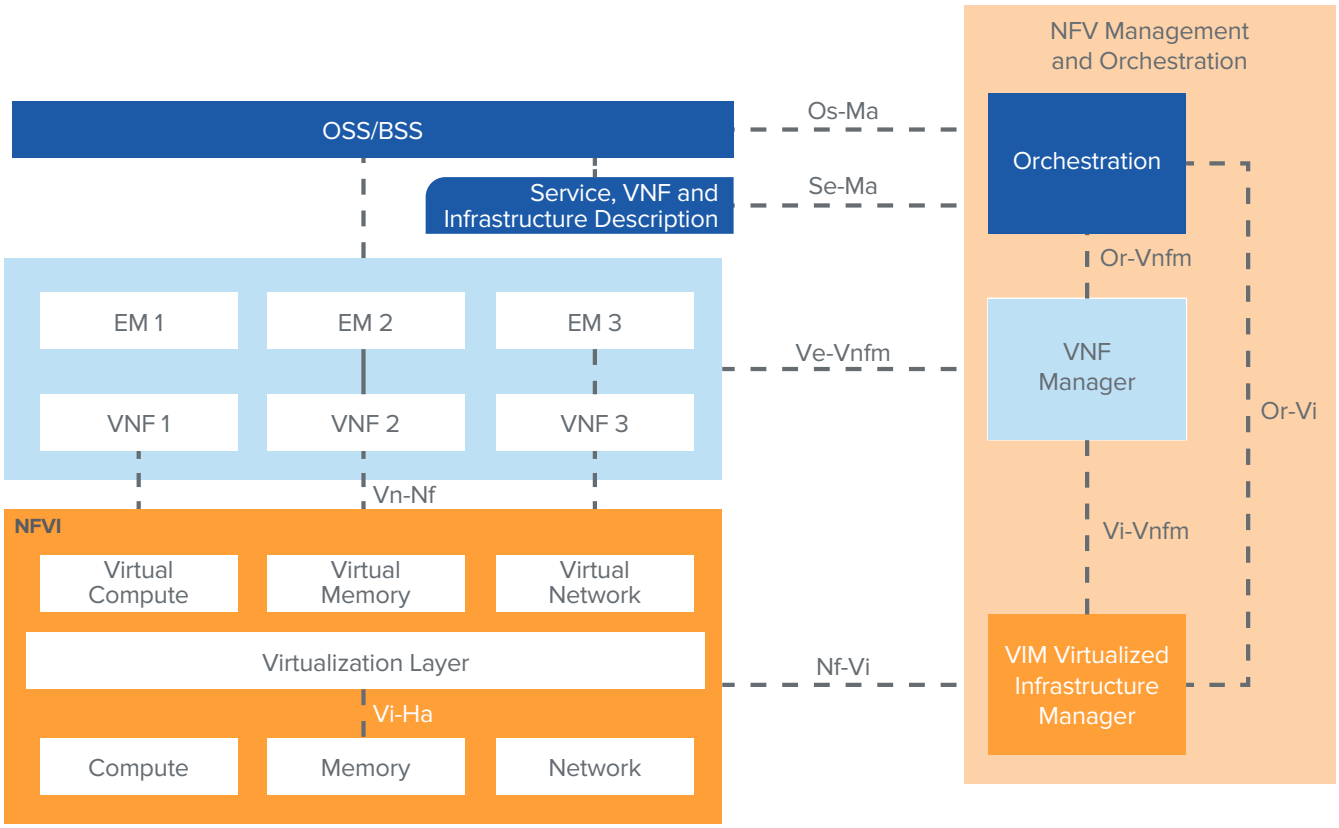

Figure 1: ETSI NFV Architecture

ETSI NFV Glossary

OSS/BSS

Operation Support System/Business Support System. OSS deals with management of configurations, faults, and the network. BSS deals with management of customers, products and orders.

EM

Element Management—responsible for managing the configuration, performance, security, etc., of VNFs.

VNF

Virtual Network Function—a virtualized network element such as a router, base station, firewall, load balancer, etc.

NFVi

Network Function Virtualization infrastructure—the environment in which VNFs run, including physical resources, virtual resources, and a virtualization layer

- **Physical Compute, Memory and Networking Resources:** Physical CPUs, memory, disk storage, and networking resources such as switches
- **Virtual Compute, Memory and Networking Resources:** Virtualized CPUs, memory, disk storage, and networking resources such as switches
- **Virtualization Layer:** Responsible for abstracting physical resources into virtual resources, e.g. CPUs into vCPUs. In practice this is the hypervisor

NFV MANO

Management and Orchestration: responsible for setting up/tearing down services, their component VNFs and managing infrastructure resources

- **NFVO NFV Orchestrator:** Generates, maintains and tears down network services, which are comprised of VNFs
- **VNFM VNF Manager:** Responsible for setting up, maintaining, and tearing down VNFs
- **VIM Virtualized Infrastructure Manager:** Responsible for controlling and managing compute, network and storage resources as well as performance measurements

From Abstraction to Real Deployments

While the ETSI definition provides a high-level conceptual abstraction of the NFV world, in practice, real-world NFV deployments must be defined in more concrete terms, for example as shown in Figure 2 below. Note that the elements in Figure 2 are not the hypothetical sub-parts of the ETSI NFV architecture, but rather real components, either software or hardware. Let's examine each of those components in more detail:

In practice, VNFs have been mostly deployed as one or more VMs. In the future, VNFs will also be implemented as one or more containers, which can be automatically deployed and managed by cluster-managers such as Kubernetes. Each VM is typically running an instance of an operating system (OS), which is usually some form of Linux or Windows. Finally, the VNF functionality is implemented as a software application running on that operating system.

OS: The operating system of the virtual machine. The OS is typically some flavor of Linux or Windows.

App: The software application which performs the network function.

Virtual Switch: A piece of software which provides connectivity to virtual and physical interfaces

vNIC: Virtual network interface controller, which provides connectivity to a virtual interface on a network

pNIC: Physical network interface controller, which provides connectivity to a physical interface on a network

Hypervisor: A piece of software which allows running one or more virtual machines on a guest machines

VM: A virtual machine—a software version of a computer with its own dedicated operating system

Container: A software version of a computer running on an operating system which may be shared by other containers.

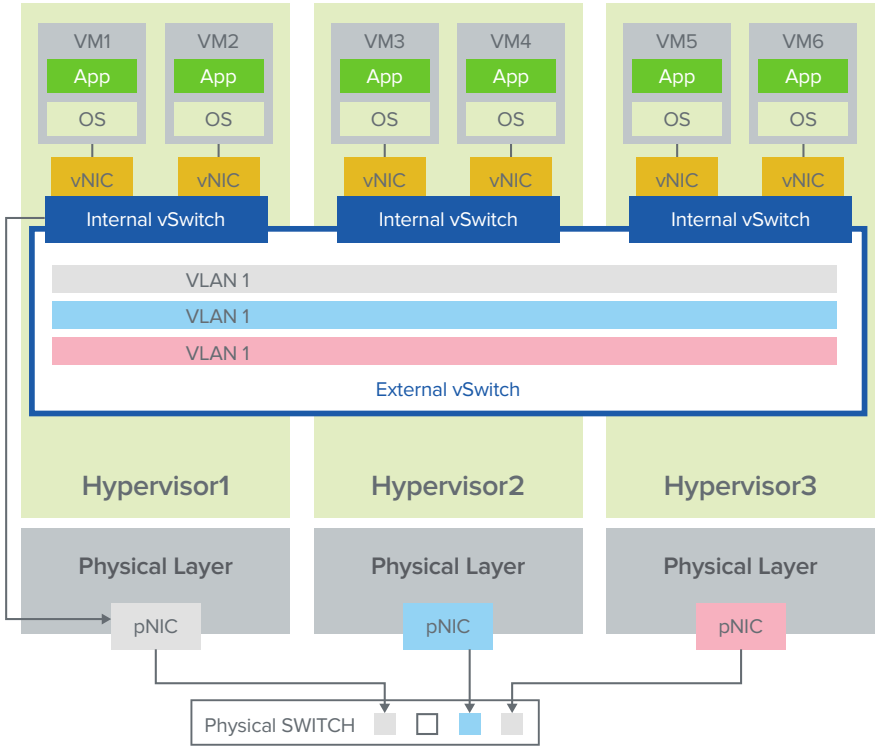

Figure 2: Example of Virtual Deployment

Benchmarking VNFs

Traditional Benchmarking

In some ways, benchmarking a VNF should be the same as benchmarking a traditional device. For example, a virtualized router should be benchmarked similarly to a traditional router, by measuring its forwarding performance using RFC 2544, as well as by measuring the scale and performance of key routing protocols such as BGP, IS-IS and OSPF. Traditional benchmarking should also be applied to virtual BNGs, virtual CPEs, and virtual firewalls/IPSs. For a (virtual) load balancer, which is the example for this case study, benchmarking typically means assessing the maximum rate of HTTP requests the load balancer can sustain. Such benchmarking has been well-defined for several years, and is now embodied in products such as Spirent MethodologyCenter, which offers scenario-based methodologies for rapid benchmarking of both traditional and virtualized devices.

Challenges Unique to VNF Benchmarking

Beyond traditional benchmarking, VNF benchmarking is different from traditional benchmarking in three distinct ways: complexity, abstraction, and concurrency.

Complexity Virtual environments are more complex since they include more components, such as hypervisors, virtual switches, etc. These additional components introduce potential areas of weakness and must be considered while benchmarking VNFs.

Abstraction Virtualization means that the data plane and control plane have been abstracted from the executing hardware. Abstraction introduces three challenges when benchmarking VNFs:

- The impact of the quantity and type of resources which have been allocated to the VNF. The variety of virtual resources continues to expand, for example, with virtual FPGAs and virtual GPUs now being available.
- The impact of certain technologies, such as DPDK, SR-IOV and CPU pinning, which are meant to address performance concerns associated with virtualization
- The time to spin up, or instantiate, a VNF (in a traditional network element, the device simply exists and does not need to be created)

Concurrency In real-world deployments, VNFs will often co-exist and possibly interact with other VNFs. Other VNFs will have their own performance constraints, and furthermore may also be competing for the same resources.

How to Test

Conquering Complexity: Testing components in isolation

Virtual switches can be benchmarked according to the methodologies described in the IETF draft document "Benchmarking Virtual Switches in OPNFV" (https://tools.ietf.org/html/draft-vsperf-bmwg-vswitch-opnfv-00). This same set of methodologies has been implemented as a suite of tests in Spirent MethodologyCenter.

In this case study (see below), we performed a basic form of isolation by evaluating the performance of the virtual environment before introducing the VNF.

Conquering Abstraction: White box testing

Historically, performance measurement of network devices could be described as "black box testing", meaning that the testing was conducted at the external interfaces of the devices, without much thought into the inner workings of the device itself, other than some basic statistics and configuration issues. In the virtual world, the boundaries are much blurrier: VNFs are not standalone devices, but rather components in a complex ecosystem where resources can be shared across multiple consumers. Consequently, performance testing in virtual environments should best be thought of as "white box testing", where measurements are made for each component in the ecosystem.

The challenge of **abstraction** really means performing benchmarking with different quantities and types of resources allocated to the VNF. For example, does doubling the amount of virtual memory double the scale of a routing protocol in a virtual router? Abstraction challenges include the benefits and pitfalls of technologies such as DPDK, which can accelerate forwarding performance, but may also increase packet delay, especially for long packets.

- We can retrieve useful and informative metrics from multiple places in the virtual infrastructure.
- From the hypervisor: CPU Utilization
- From the virtual switch: System stats, virtual NIC stats, physical NIC stats

From the **VNF** or Operating System instance: CPU Utilization, Memory Utilization, Disk I/O Rate, Memory access Latency, Cache read/write latency, Instructions processed per second, Throughput for memory access to enable such white-box visibility while performing VNF benchmarking, Spirent MethodologyCenter now includes a unique "NFVi Statistics" feature that provides OpenStack metrics on the underlying hypervisor, virtual switch and VNF or operating system.

Fault Isolation—identifying bottlenecks

While gaining visibility into resource utilization is helpful, the volume of metrics can also be overwhelming —there is a myriad of statistics available from the virtual infrastructure. Furthermore, since packets pass through many components on their virtual journey, identifying the location of a bottleneck can be like looking for a needle in a haystack.

Fortunately, the process of correlating large quantities of results can be automated through analytics. Analytics can be used to:

- Isolate correlations across a broad range of stimuli and responses
- Determine whether any resources are operating on "the edge" of over-utilization

Again, Spirent MethodologyCenter implements fault isolation by analyzing the OpenStack metrics specified above and identifying correlations which may assist with localizing root causes to poor performance.

Conquering Concurrency: Testing with ***concurrency*** means that additional VNFs should be deployed while the primary VNF is benchmarked. The additional VNFs should also be presented with realistic workloads so as to model actual deployments as closely as possible. This scenario is referred to as ***Noisy Neighbor Testing***. In practice, the effect of additional VNFs can be simulated with Spirent CloudStress, which generates synthetic workloads on CPU, memory, storage and network.

Case Study: DUT Profile and Test Scenarios

For this case study, our test environment included the following elements:

- A commercial VNF implementation
- OpenStack distributions from Wind River and Canonical
- OS (Ubuntu) and Hypervisor from Canonical
- Instantiation via libvirt from RedHat
- Underlying hardware from HPE
- Orchestration using Rift.io and OpenStack Tacker

Testing the Virtual Environment in Isolation

Before we tested the VNF, it was important to determine the performance of the virtual environment. Since, as explained above, the environment itself can be a cause of poor performance, this step is critical. To do this, we conducted two experiments:

Workload was input to the system via physical interfaces using a physical traffic generator (Spirent TestCenter).

Workload was input to the system via virtual interfaces using a virtual traffic generator (Spirent TestCenter virtual). For this scenario, we used Spirent TestCenter virtual in two separate modes: with DPDK turned off, and with DPDK turned on. The resources allocated to the Spirent TestCenter virtual ports were isolated by pinning CPUs.

Testing the VNF in the Virtual Environment

We loaded the VNF using simulated traffic, generated by the Spirent TestCenter traffic generator. Traffic was both (fully stateful) SIP and HTTP. As in the isolation experiments above, tests were performed with physical Spirent TestCenter ports, and with virtual Spirent Test DPDK turned off and turned on.

Testing with Concurrency

We also evaluated the performance of the VNF with concurrent VNFs, we used the Spirent CloudStress product to generate synthetic workloads to stress compute, memory and networking resources. This is sometimes referred to as the "noisy neighbor" scenario. In this case, the VNF under test was loaded using stateful traffic simulated by Spirent TestCenter virtual ports.

Key Takeaways

Our experiments revealed multiple interesting findings:

- For the baseline experiment, the performance of Spirent TestCenter Virtual ports was approximately ten times greater with DPDK turned on versus with DPDK turned off
- The Wind River OpenStack distribution proved to be reliable, scalable, and in general high quality.
- Closed systems (such as Wind River) in general had far fewer issues compared with open-source systems
- In spite of existing ETSI draft standards for MANO (available here: https://docbox.etsi.org/ISG%2FNFV%2FOpen%2FDrafts%2F), instantiation using orchestration tools is challenging and inconsistent across implementations

About Spirent Communications

Spirent Communications (LSE: SPT) is a global leader with deep expertise and decades of experience in testing, assurance, analytics and security, serving developers, service providers, and enterprise networks.

We help bring clarity to increasingly complex technological and business challenges.

Spirent's customers have made a promise to their customers to deliver superior performance. Spirent assures that those promises are fulfilled.

For more information, visit: www.spirent.com

Spirent Test Services for VNF Benchmarking

Spirent Communications offers a full range of testing services and products to ensure that your transition to virtual networking is successful. Our industry experts are fully capable and will accelerate your VNF journey. As your trusted partner, Spirent can own the entire VNF benchmarking process, from defining the test plan, test creation, test execution, and report writing. Alternatively, if you prefer to work more independently or a more flexible arrangement, we can set up and train your team with the correct Spirent test products to efficiently perform any or all of these steps on your own.

AMERICAS 1-800-SPIRENT
+1-800-774-7368
sales@spirent.com

US Government & Defense
info@spirentfederal.com
spirentfederal.com

EUROPE AND THE MIDDLE EAST
+44 (0) 1293 767979
emeainfo@spirent.com

ASIA AND THE PACIFIC
+86-10-8518-2539
salesasia@spirent.com

Spirent Promise. Assured.

Rev B | 08/18