
opnfv-kvmforopnfv Documentation

Release latest

May 09, 2019

1	Overview of Documentation	1
1.1	KVM4NFV Project Overview	1
1.1.1	Project Purpose	1
1.1.2	Project Description	1
2	KVM4NFV Installation Procedure	3
2.1	Abstract	3
2.2	KVM4NFV Installation Instruction	3
2.2.1	Preparing the installation	3
2.2.2	HW requirements	3
2.2.3	Build instructions	3
2.2.4	Installation instructions	4
2.2.5	Post-installation activities	4
2.3	Release Note for KVM4NFV CICD	5
2.3.1	Abstract	5
2.3.2	Introduction	5
2.3.3	Release Data	5
2.3.4	Document version change	5
2.3.5	Reason for new version	6
2.3.5.1	Feature additions	6
2.3.6	Known issues	6
2.3.7	Workarounds	6
3	KVM4NFV Design Guide	7
3.1	KVM4NFV design description	7
3.1.1	Design Considerations	8
3.1.2	Goals and Guidelines	8
3.1.3	Test plan	9
3.1.3.1	Reference	9
4	KVM4NFV Requirements Guide	13
4.1	Kvm4nfv Requirements	13
4.1.1	Introduction	13
4.1.2	Scope and Purpose	13
4.1.3	Methods and Instrumentation	13
4.1.4	Features to be tested	14
4.1.5	Dependencies	14

4.1.6	Reference	14
5	KVM4NFV Configuration Guide	15
5.1	Configuration Abstract	15
5.2	Configuration Options	15
5.3	Scenariomatrix	16
5.3.1	Euphrates scenario overview	16
5.3.2	Scenario Naming	16
5.3.3	Installing your scenario	18
5.4	Low Latency Feature Configuration Description	18
5.4.1	Introduction	18
5.4.2	Configuration of Cyclictest	18
5.4.2.1	Pre-configuration activities	18
5.4.2.2	Hardware configuration	21
6	KVM4NFV Scenarios Overview and Description	23
6.1	Scenario Abstract	23
6.1.1	Release Features	23
6.1.2	E- Release Scenario's overview	24
6.2	KVM4NFV Scenario-Description	24
6.2.1	Abstract	24
6.2.2	Version Features	24
6.2.3	Introduction	25
6.2.4	System pre-requisites	25
6.2.5	Environment Setup	26
6.2.5.1	Enable network access after the installation	26
6.2.5.2	Configuring Proxy	26
6.2.5.3	Install redsocks	26
6.2.5.4	Network Time Protocol (NTP) setup and configuration	28
6.2.6	Scenario Testing	28
6.2.6.1	Fuel	28
6.2.6.2	Apex	31
6.2.6.3	OPNFV-Playground	34
6.2.6.4	Jenkins Project	36
7	os-nosdn-kvm_ovs_dpdk-noha Overview and Description	37
7.1	os-nosdn-kvm_ovs_dpdk-noha Description	37
7.1.1	Introduction	37
7.1.2	Scenario Components and Composition	37
7.1.3	Scenario Usage Overview	40
7.1.4	Scenario Components and Composition	41
7.1.5	Scenario Usage Overview	42
7.1.6	References	42
8	os-nosdn-kvm_ovs_dpdk-ha Overview and Description	43
8.1	os-nosdn-kvm_ovs_dpdk-ha Description	43
8.1.1	Introduction	43
8.1.2	Scenario Components and Composition	43
8.1.3	Scenario Usage Overview	46
8.1.4	Scenario Components and Composition	47
8.1.5	Scenario Usage Overview	48
8.1.6	References	48
9	os-nosdn-kvm_ovs_dpdk_bar-noha Overview and Description	49
9.1	os-nosdn-kvm_ovs_dpdk_bar-ha Description	49

9.1.1	Introduction	49
9.1.2	Scenario Components and Composition	49
9.1.3	Scenario Usage Overview	52
9.1.4	Known Limitations, Issues and Workarounds	53
9.1.5	References	53
10	os-nosdn-kvm_ovs_dpdk_bar-ha Overview and Description	55
10.1	os-nosdn-kvm_ovs_dpdk_bar-ha Description	55
10.1.1	Introduction	55
10.1.2	Scenario Components and Composition	55
10.1.3	Scenario Usage Overview	58
10.1.4	Known Limitations, Issues and Workarounds	59
10.1.5	References	59
11	KVM4NFV User Guide	61
11.1	Userguide Abstract	61
11.2	Userguide Introduction	61
11.2.1	Overview	61
11.2.2	KVM4NFV Features	61
11.2.3	General usage guidelines	62
11.2.4	Scenarios User Guide	62
11.3	Using common platform components	62
11.4	Using Euphrates Features	63
11.5	FTrace Debugging Tool	63
11.5.1	About Ftrace	63
11.5.2	Version Features	63
11.5.3	Implementation of Ftrace	63
11.5.4	Files in Ftrace:	64
11.5.5	Avaliable Tracers	64
11.5.6	Ftrace in KVM4NFV	65
11.5.7	Ftrace Usage in KVM4NFV Kernel Debugging:	66
11.5.8	Enabling Ftrace in KVM4NFV	66
11.5.9	Details of enable_trace script	66
11.5.10	BREAKTRACE	67
11.5.11	Post-execute scripts	67
11.5.12	Details of disable_trace Script	67
11.5.13	Publishing Ftrace logs:	67
11.6	KVM4NFV Dashboard Guide	68
11.6.1	Dashboard for KVM4NFV Daily Test Results	68
11.6.2	Abstract	68
11.6.3	Version Features	68
11.6.4	Installation Steps:	69
11.6.5	Configuring the Dispatcher Type:	69
11.6.5.1	Detailing the dispatcher module in verify and daily Jobs:	70
11.6.6	Understanding Kvm4nfv Grafana Dashboard	72
11.6.6.1	1. Idle-Idle Graph	74
11.6.6.2	2. CPU_Stress-Idle Graph	74
11.6.6.3	3. Memory_Stress-Idle Graph	74
11.6.6.4	4. IO_Stress-Idle Graph	74
11.6.6.5	Packet Forwarding Results	76
11.6.7	Understanding Kvm4nfv Grafana Dashboard	76
11.6.8	Future Scope	77
11.7	Low Latency Environment	77
11.7.1	Hardware Environment Description	77

11.7.1.1	CPU Features	77
11.7.1.2	CPU Topology	77
11.7.1.3	BIOS Setup	77
11.7.2	Software Environment Setup	77
11.7.2.1	Kernel Parameter	77
11.7.2.2	Run-time Environment Setup	78
11.7.3	Test cases to measure Latency	78
11.7.4	1. Cyclicttest case	78
11.7.4.1	Understanding the naming convention	78
11.7.4.2	Version Features	79
11.7.4.3	Idle-Idle test-type	79
11.7.4.4	CPU_Stress-Idle test-type	80
11.7.4.5	Memory_Stress-Idle test-type	80
11.7.4.6	IO_Stress-Idle test-type	80
11.7.4.7	CPU_Stress-CPU_Stress test-type	80
11.7.4.8	Memory_Stress-Memory_Stress test-type	80
11.7.4.9	IO_Stress-IO_Stress test type	82
11.7.5	2. Packet Forwarding Test cases	82
11.7.5.1	Packet forwarding to Host	82
11.7.5.2	Packet forwarding to Guest	82
11.7.5.3	Packet forwarding to Guest using SRIOV	82
11.8	Fast Live Migration	83
11.8.1	Current Challenges	83
11.8.2	Optimizations	85
11.8.3	Test Environment	85
11.8.4	Test Result	87
11.9	Euphrates OpenStack User Guide	88
11.9.1	OpenStack references	88
11.9.2	Connecting to the OpenStack instance	88
11.10	Packet Forwarding	88
11.10.1	About Packet Forwarding	88
11.10.2	Version Features	89
11.10.3	VSPERF	89
11.10.3.1	Installation	89
11.10.3.2	Supported Operating Systems	89
11.10.3.3	Supported vSwitches	90
11.10.3.4	Supported Hypervisors	90
11.10.3.5	Other Requirements	90
11.10.3.6	For CentOS 7	90
11.10.3.7	Working Behind a Proxy	91
11.10.4	Traffic-Generators	91
11.10.5	IXIA Setup	91
11.10.5.1	Hardware Requirements	91
11.10.5.2	Installation	91
11.10.5.3	On the CentOS 7 system	92
11.10.5.4	On the IXIA client software system	92
11.10.6	VSPERF configuration	92
11.10.6.1	Test results share	94
11.10.6.2	Cloning and building src dependencies	94
11.10.6.3	Configure the <i>./conf/10_custom.conf</i> file	94
11.10.6.4	Using a custom settings file	95
11.10.6.5	vloop_vnf	95
11.10.6.6	l2fwd Kernel Module	95
11.10.6.7	Executing tests	95

11.10.7	Testcases	96
11.10.8	VSPERF modes of operation	96
11.10.9	Packet Forwarding Test Scenarios	97
11.10.9.1	Packet Forwarding Host Scenario	97
11.10.9.2	Packet Forwarding Guest Scenario (PXP Deployment)	97
11.10.9.3	Packet Forwarding SRIOV Scenario	102
11.10.9.4	Using vfio_pci with DPDK	102
11.10.9.5	Using SRIOV support	102
11.10.9.6	Results	105
11.11	PCM Utility in KVM4NFV	105
11.11.1	Collecting Memory Bandwidth Information using PCM utility	105
11.11.2	About PCM utility	105
11.11.3	Version Features	106
11.11.3.1	Implementation of pcm-memory.x:	106
11.11.3.2	pcm-memory.x in KVM4NFV:	107
11.11.4	Future Scope	108
11.12	Low Latency Tunning Suggestion	108
11.12.1	Platform Configuration	108
11.12.2	Operating System Configuration	108
12	KVM4NFV Releasenotes	111

1.1 KVM4NFV Project Overview

1.1.1 Project Purpose

Purpose:

This document provides an overview of the areas that can be addressed to enhance the KVM Hypervisor for NFV. It is intended to capture and convey the significant changes which have been made on the KVM Hypervisor.

1.1.2 Project Description

The NFV hypervisors provide crucial functionality in the NFV Infrastructure(NFVI).The existing hypervisors, however, are not necessarily designed or targeted to meet the requirements for the NFVI.

This design focuses on the enhancement of following area for KVM Hypervisor

- **Minimal Interrupt latency variation for data plane VNFs:**
 - Minimal Timing Variation for Timing correctness of real-time VNFs
 - Minimal packet latency variation for data-plane VNFs
- Fast live migration

The detailed understanding of this project is organized into different sections-

- **installation procedure** - This will give the user instructions on how to deploy available KVM4NFV build scenario.
- **design** - This includes the parameters or design considerations taken into account for achieving minimal interrupt latency for the data VNFs.

- **requirements** - This includes the introduction of KVM4NFV project, specifications of how the project should work, and constraints placed upon its execution.
- **configuration guide** - This provides guidance for configuring KVM4NFV environment, even with the use of specific installer tools for deploying some components, available in the Euphrates release of OPNFV.
- **scenarios** - This includes the sceanrios that are currently implemented in the kvm4nfv project,features of each scenario and a general guide to how to deploy them.
- **userguide** - This provides the required technical assistance to the user, in using the KVM4NFV process.
- **release notes** - This describes a brief summary of recent changes, enhancements and bug fixes in the KVM4NFV project.
- **glossary** - It includes the definition of terms, used in the KVM4NFV project.

KVM4NFV Installation Procedure

2.1 Abstract

This document will give the instructions to user on how to deploy available KVM4NFV build scenario verified for the Euphrates release of the OPNFV platform.

2.2 KVM4NFV Installation Instruction

2.2.1 Preparing the installation

The OPNFV project- KVM4NFV (<https://gerrit.opnfv.org/gerrit/kvmfornfv.git>) is cloned first, to make the build scripts for Qemu & Kernel, Rpms and Debians available.

2.2.2 HW requirements

These build scripts are triggered on the Jenkins-Slave build server. Currently Intel POD10 is used as test environment for kvm4nfv to execute cyclictst. As part of this test environment Intel pod10-jump is configured as jenkins slave and all the latest build artifacts are downloaded on to it. Intel pod10-node1 is the host on which a guest vm will be launched as a part of running cyclictst through yardstick.

2.2.3 Build instructions

Builds are possible for the following packages-

kvmfornfv source code

The ./ci/build.sh is the main script used to trigger the Rpms (on 'centos') and Debians (on 'ubuntu') builds in this case.

- How to build Kernel/Qemu Rpms- To build rpm packages, build.sh script is run with -p and -o option (i.e. if -p package option is passed as "centos" or in default case). Example:

```
cd kvmfornfv/

For Kernel/Qemu RPMs,
sh ./ci/build.sh -p centos -o build_output
```

- How to build Kernel/Qemu Debians- To build debian packages, build.sh script is run with -p and -o option (i.e. if -p package option is passed as “ubuntu”). Example:

```
cd kvmfornfv/

For Kernel/Qemu Debians,
sh ./ci/build.sh -p ubuntu -o build_output
```

- How to build all Kernel & Qemu, Rpm & Debians- To build both debian and rpm packages, build.sh script is run with -p and -o option (i.e. if -p package option is passed as “both”). Example:

```
cd kvmfornfv/

For Kernel/Qemu RPMs and Debians,
sh ./ci/build.sh -p both -o build_output
```

Note: Kvm4nfv can be installed in two ways

1. As part of a [scenario deployment](#)
 2. As a [stand alone](#) component
-

For installation of kvmfornfv as part of scenario deployment use this [‘link’](#)

```
http://artifacts.opnfv.org/kvmfornfv/docs/index.html#document-scenarios/kvmfornfv.
↪scenarios.description
```

2.2.4 Installation instructions

Installation can be done in the following ways-

1. From kvmfornfv source code- The build packages that are prepared in the above section, are installed differently depending on the platform.

Please visit the links for each-

- Centos : https://www.centos.org/docs/5/html/Deployment_Guide-en-US/s1-rpm-using.html
- Ubuntu : <https://help.ubuntu.com/community/InstallingSoftware>

2. Using Fuel installer-

- Please refer to the document present at [/fuel-plugin/README.md](#)

2.2.5 Post-installation activities

After the packages are built, test these packages by executing the scripts present in ci/envs for configuring the host and guest respectively.

2.3 Release Note for KVM4NFV CICD

2.3.1 Abstract

This document contains the release notes for the Euphrates release of OPNFV when using KVM4NFV CICD process.

2.3.2 Introduction

Provide a brief introduction of how this configuration is used in OPNFV release using KVM4VFV CICD as scenario.

Be sure to reference your scenario installation instruction.

2.3.3 Release Data

Project	NFV Hypervisors-KVM
Repo/tag	kvmformfv
Release designation	
Release date	2017-10-06
Purpose of the delivery	<ul style="list-style-type: none">• Automate the KVM4VFV CICD scenario• Executing latency test cases• Collection of logs for debugging

2.3.4 Document version change

The following documents are added-

- configurationguide
- installationprocedure
- userguide
- overview
- glossary
- releasenotes

2.3.5 Reason for new version

2.3.5.1 Feature additions

JIRA REFERENCE	SLOGAN
JIRA:	NFV Hypervisors-KVMFORNFV-72
JIRA:	NFV Hypervisors-KVMFORNFV-73
JIRA:	NFV Hypervisors-KVMFORNFV-78
JIRA:	NFV Hypervisors-KVMFORNFV-86
JIRA:	NFV Hypervisors-KVMFORNFV-87
JIRA:	NFV Hypervisors-KVMFORNFV-88
JIRA:	NFV Hypervisors-KVMFORNFV-89
JIRA:	VSPERF-510
JIRA:	YARDSTICK-783
JIRA:	YARDSTICK-815

2.3.6 Known issues

JIRA TICKETS:

JIRA REFERENCE	SLOGAN

2.3.7 Workarounds

See JIRA: <https://jira.opnfv.org/projects>

For more information on the OPNFV Euphrates release, please visit <http://www.opnfv.org/euphrates>

3.1 KVM4NFV design description

This design focuses on the enhancement of following area for KVM Hypervisor

- **Minimal Interrupt latency variation for data plane VNFs:**
 - Minimal Timing Variation for Timing correctness of real-time VNFs
 - Minimal packet latency variation for data-plane VNFs
- Fast live migration

Minimal Interrupt latency variation for data plane VNFs

Processing performance and latency depend on a number of factors, including the CPUs (frequency, power management features, etc.), micro-architectural resources, the cache hierarchy and sizes, memory (and hierarchy, such as NUMA) and speed, inter-connects, I/O and I/O NUMA, devices, etc.

There are two separate types of latencies to minimize:

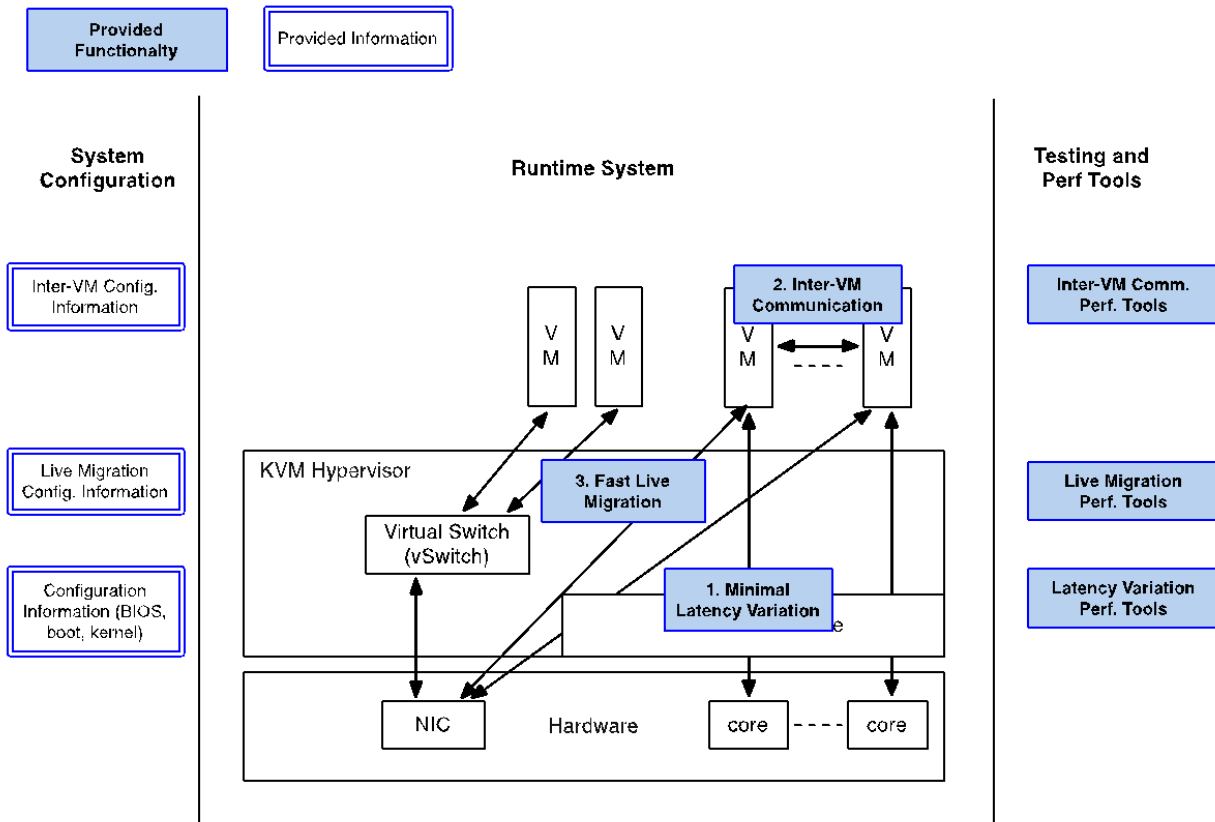
1. Minimal Timing Variation for Timing correctness of real-time VNFs – timing correctness for scheduling operations (such as Radio scheduling)
2. Minimal packet latency variation for data-plane VNFs – packet delay variation, which applies to packet processing.

For a VM, interrupt latency (time between arrival of H/W interrupt and invocation of the interrupt handler in the VM), for example, can be either of the above or both, depending on the type of the device. Interrupt latency with a (virtual) timer can cause timing correctness issues with real-time VNFs even if they only use polling for packet processing.

We assume that the VNFs are implemented properly to minimize interrupt latency variation within the VMs, but we have additional causes of latency variation on KVM:

- Asynchronous (e.g. external interrupts) and synchronous (e.g. instructions) VM exits and handling in KVM (and kernel routines called), which may have loops and spin locks
- Interrupt handling in the host Linux and KVM, scheduling and virtual interrupt delivery to VNFs

- Potential VM exit (e.g. EOI) in the interrupt service routines in VNFs
- Exit to the user-level (e.g. QEMU)



3.1.1 Design Considerations

The latency variation and jitters can be minimized with the below steps (with some in parallel):

1. Statically and exclusively assign hardware resources (CPUs, memory, caches,) to the VNFs.
2. Pre-allocate huge pages (e.g. 1 GB/2MB pages) and guest-to-host mapping, e.g. EPT (Extended Page Table) page tables, to minimize or mitigate latency from misses in caches,
3. Use the host Linux configured for hard real-time and packet latency, Check the set of virtual devices used by the VMs to optimize or eliminate virtualization overhead if applicable
4. Use advanced hardware virtualization features that can reduce or eliminate VM exits, if present, and
5. Inspect the code paths in KVM and associated kernel services to eliminate code that can cause latencies (e.g. loops and spin locks).
6. Measure latencies intensively. We leverage the existing testing methods. OSADL, for example, defines industry tests for timing correctness.

3.1.2 Goals and Guidelines

The output of this project will provide :

1. A list of the performance goals, which will be obtained by the OPNFV members (as described above)
2. A set of comprehensive instructions for the system configurations (hardware features, BIOS setup, kernel parameters, VM configuration, options to QEMU/KVM, etc.)
3. The above features to the upstream of Linux, the real-time patch set, KVM, QEMU, libvirt, and
4. Performance and interrupt latency measurement tools

3.1.3 Test plan

The tests that need to be conducted to make sure that all components from OPNFV meet the requirement are mentioned below:

Timer test: This test utilizes the `cyclictst` (<https://rt.wiki.kernel.org/index.php/Cyclictst>) to test the guest timer latency (the latency from the time that the guest timer should be triggered to the time the guest timer is really triggered).

Device Interrupt Test: A device on the hardware platform triggers interrupt every one ms and the device interrupt will be delivered to the VNF. This test covers the latency from the interrupt happened on the hardware to the time the interrupt is handled in the VNF.

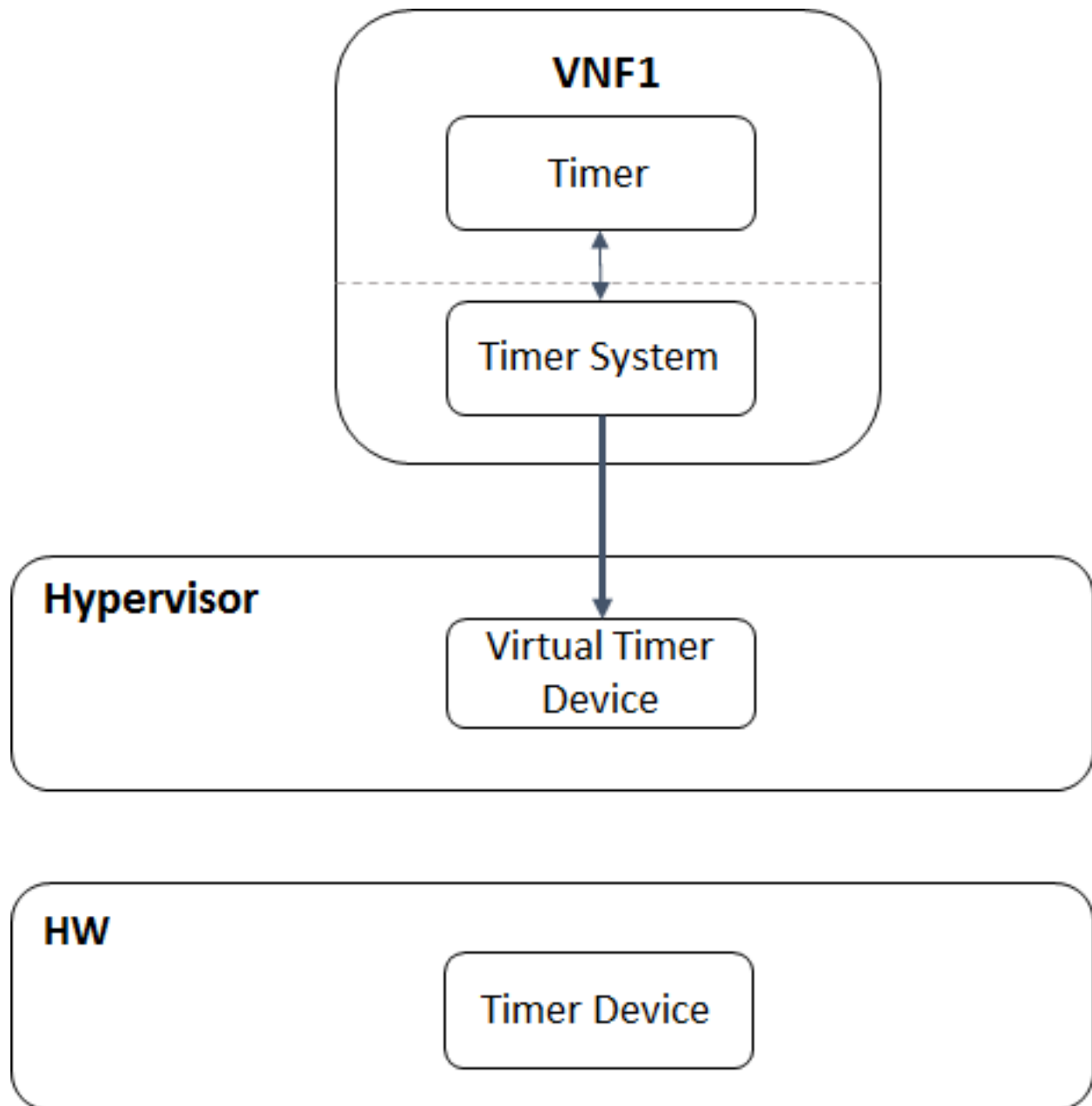
Packet forwarding (DPDK OVS): A packet is sent from TG (Traffic Generator) to a VNF. The VNF, after processing the packet, forwards the packet to another NIC and in the end the packet is received by the traffic generator. The test checks the latency from the packet is sent out by the TC to the time the packet is received by the TC.

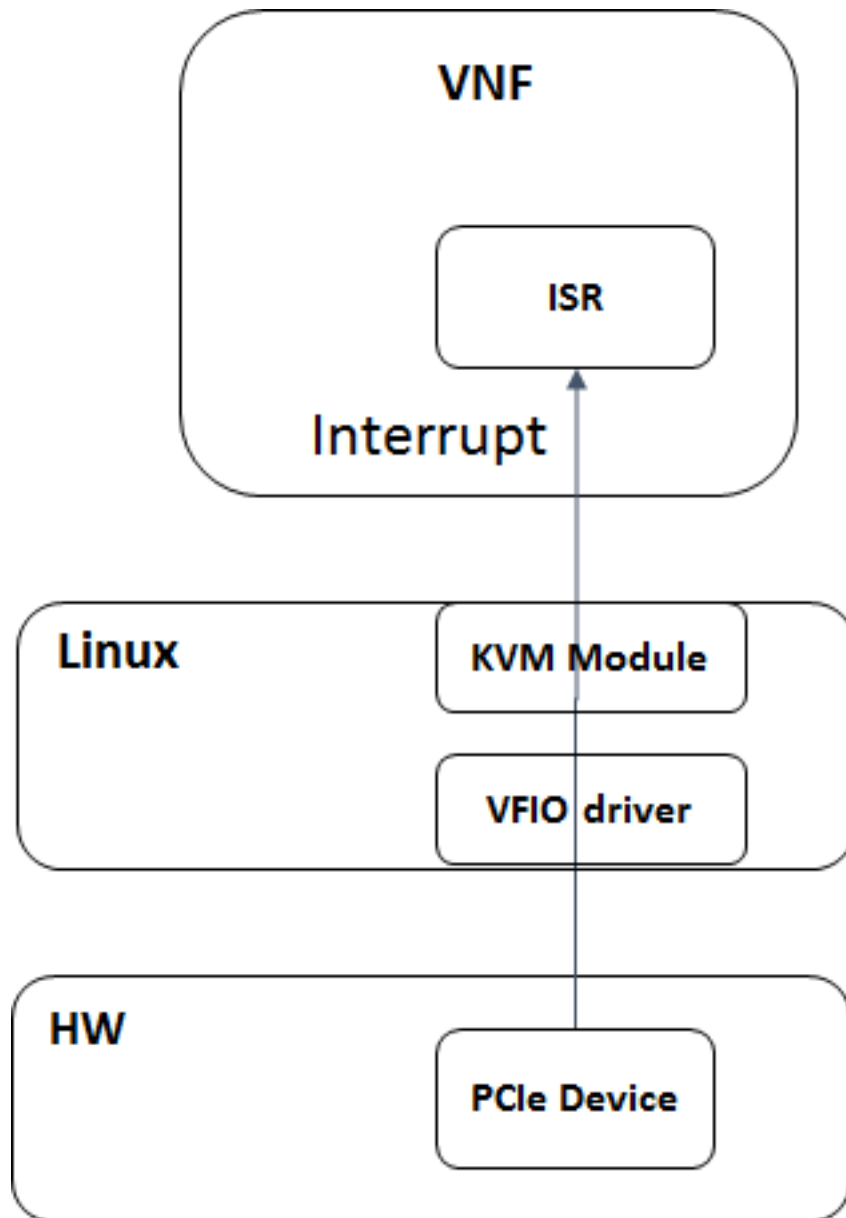
Packet Forwarding (SR-IOV): This test is similar to Packet Forwarding (DPDK OVS). However, instead of using virtio NIC devices on the guest, a PCI NIC or a PCI VF NIC is assigned to the guest for network access.

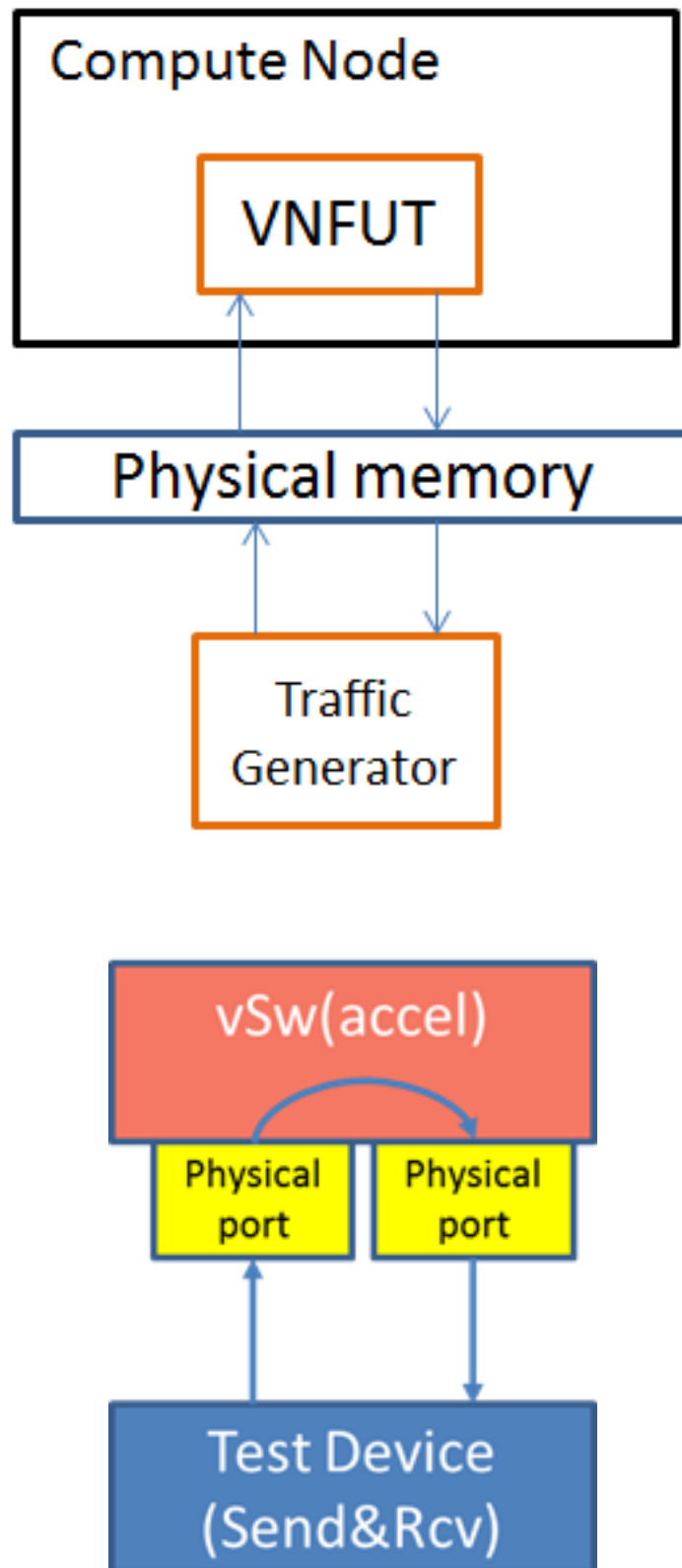
Bare-metal Packet Forwarding: This is used to compare with the above packet forwarding scenario.

3.1.3.1 Reference

<https://wiki.opnfv.org/display/kvm/>







4.1 Kvm4nfv Requirements

4.1.1 Introduction

The NFV hypervisors provide crucial functionality in the NFV Infrastructure(NFVI).The existing hypervisors, however, are not necessarily designed or targeted to meet the requirements for the NFVI.

This document specifies the list of requirements that need to be met as part of this “NFV Hypervisors-KVM” project in Euphrates release.

As part of this project we need to make collaborative efforts towards enabling the NFV features.

4.1.2 Scope and Purpose

The main purpose of this project is to enhance the KVM hypervisor for NFV, by looking at the following areas initially:

- **Minimal Interrupt latency variation for data plane VNFs:**
 - Minimal Timing Variation for Timing correctness of real-time VNFs
 - Minimal packet latency variation for data-plane VNFs
- Inter-VM communication
- Fast live migration

The output of this project would be list of the performance goals,comprehensive instructions for the system configurations,tools to measure Performance and interrupt latency.

4.1.3 Methods and Instrumentation

The above areas would require software development and/or specific hardware features, and some need just configurations information for the system (hardware, BIOS, OS, etc.).

A right configuration is critical for improving the NFV performance/latency. Even working on the same code base, different configurations can make completely different performance/latency result. Configurations that can be made as part of this project to tune a specific scenario are:

1. **Platform Configuration** : Some hardware features like Power management, Hyper-Threading, Legacy USB Support/Port 60/64 Emulation, SMI can be configured.
2. **Operating System Configuration** : Some configuration features like CPU isolation, Memory allocation, IRQ affinity, Device assignment for VM, Tickless, TSC, Idle, _RCU_NOCB_, Disable the RT throttling, NUMA can be configured.
3. **Performance/Latency Tuning** : Application level configurations like timers, Making vfio MSI interrupt as non-threaded, Cache Allocation Technology (CAT) enabling can be tuned to improve the NFV performance/latency.

4.1.4 Features to be tested

The tests that need to be conducted to make sure that latency is addressed are:

1. Timer test
2. Device Interrupt Test
3. Packet forwarding (DPDK OVS)
4. Packet Forwarding (SR-IOV)
5. Bare-metal Packet Forwarding

4.1.5 Dependencies

1. OPNFV Project: “Characterize vSwitch Performance for Telco NFV Use Cases” (VSPERF) for performance evaluation of ivshmem vs. vhost-user.
2. OPNFV Project: “Pharos” for Test Bed Infrastructure, and possibly “Yardstick” for infrastructure verification.
3. There are currently no similar projects underway in OPNFV or in an upstream project
4. The relevant upstream project to be influenced here is QEMU/KVM and libvirt.
5. In terms of HW dependencies, the aim is to use standard IA Server hardware for this project, as provided by OPNFV Pharos.

4.1.6 Reference

<https://wiki.opnfv.org/display/kvm/>

5.1 Configuration Abstract

This document provides guidance for the configurations available in the Euphrates release of OPNFV

The release includes four installer tools leveraging different technologies; Apex, Compass4nfv, Fuel and JOID, which deploy components of the platform.

This document also includes the selection of tools and components including guidelines for how to deploy and configure the platform to an operational state.

5.2 Configuration Options

OPNFV provides a variety of virtual infrastructure deployments called scenarios designed to host virtualised network functions (VNF's). KVM4NFV scenarios provide specific capabilities and/or components aimed to solve specific problems for the deployment of VNF's. KVM4NFV scenario includes components such as OpenStack, KVM etc. which includes different source components or configurations.

Note:

- Each KVM4NFV [scenario](#) provides unique features and capabilities, it is important to understand your target platform capabilities before installing and configuring. This configuration guide outlines how to configure components in order to enable the features required.
 - More details of kvm4nfv scenarios installation and description can be found in the [scenario guide](#) of kvm4nfv docs
-

5.3 Scenariomatrix





Scenarios are implemented as deployable compositions through integration with an installation tool. OPNFV supports multiple installation tools and for any given release not all tools will support all scenarios. While our target is to establish parity across the installation tools to ensure they can provide all scenarios, the practical challenge of achieving that goal for any given feature and release results in some disparity.

5.3.1 Euphrates scenario overview

The following table provides an overview of the installation tools and available scenario's in the Euphrates release of OPNFV.

Scenario status is indicated by a weather pattern icon. All scenarios listed with a weather pattern are possible to deploy and run in your environment or a Pharos lab, however they may have known limitations or issues as indicated by the icon.

Weather pattern icon legend:

Weather Icon	Scenario Status
	Stable, no known issues
	Stable, documented limitations
	Deployable, stability or feature limitations
	Not deployed with this installer

Scenarios that are not yet in a state of “Stable, no known issues” will continue to be stabilised and updates will be made on the stable/euphrates branch. While we intend that all Euphrates scenarios should be stable it is worth checking regularly to see the current status. Due to our dependency on upstream communities and code, some issues may not be resolved prior to E release.

5.3.2 Scenario Naming

In OPNFV scenarios are identified by short scenario names, these names follow a scheme that identifies the key components and behaviours of the scenario. The rules for scenario naming are as follows:

`os-[controller]-[feature]-[mode]-[option]`

Details of the fields are

- **[os]:** mandatory
 - Refers to the platform type used
 - possible value: os (OpenStack)
- **[controller]:** mandatory
 - Refers to the SDN controller integrated in the platform

- example values: nosdn, ocl, odl, onos
- **[feature]:** mandatory
 - Refers to the feature projects supported by the scenario
 - example values: nofeature, kvm, ovs, sfc
- **[mode]:** mandatory
 - Refers to the deployment type, which may include for instance high availability
 - possible values: ha, noha
- **[option]:** optional
 - Used for the scenarios those do not fit into naming scheme.
 - The optional field in the short scenario name should not be included if there is no optional scenario.

Some examples of supported scenario names are:

- **os-nosdn-kvm-noha**
 - This is an OpenStack based deployment using neutron including the OPNFV enhanced KVM hypervisor
- **os-onos-nofeature-ha**
 - This is an OpenStack deployment in high availability mode including ONOS as the SDN controller
- **os-odl_l2-sfc**
 - This is an OpenStack deployment using OpenDaylight and OVS enabled with SFC features
- **os-nosdn-kvm_ovs_dpdk-ha**
 - This is an Openstack deployment with high availability using OVS, DPDK including the OPNFV enhanced KVM hypervisor * This deployment has 3-Controller and 2-Compute nodes
- **os-nosdn-kvm_ovs_dpdk-noha**
 - This is an Openstack deployment without high availability using OVS, DPDK including the OPNFV enhanced KVM hypervisor * This deployment has 1-Controller and 3-Compute nodes
- **os-nosdn-kvm_ovs_dpdk_bar-ha**
 - This is an Openstack deployment with high availability using OVS, DPDK including the OPNFV enhanced KVM hypervisor and Barometer
 - This deployment has 3-Controller and 2-Compute nodes
- **os-nosdn-kvm_ovs_dpdk_bar-noha**
 - This is an Openstack deployment without high availability using OVS, DPDK including the OPNFV enhanced KVM hypervisor and Barometer
 - This deployment has 1-Controller and 3-Compute nodes

5.3.3 Installing your scenario

There are two main methods of deploying your target scenario, one method is to follow this guide which will walk you through the process of deploying to your hardware using scripts or ISO images, the other method is to set up a Jenkins slave and connect your infrastructure to the OPNFV Jenkins master.

For the purposes of evaluation and development a number of Euphrates scenarios are able to be deployed virtually to mitigate the requirements on physical infrastructure. Details and instructions on performing virtual deployments can be found in the installer specific installation instructions.

To set up a Jenkins slave for automated deployment to your lab, refer to the [Jenkins slave connect guide](#).

5.4 Low Latency Feature Configuration Description

5.4.1 Introduction

In KVM4NFV project, we focus on the KVM hypervisor to enhance it for NFV, by looking at the following areas initially

- **Minimal Interrupt latency variation for data plane VNFs:**
 - Minimal Timing Variation for Timing correctness of real-time VNFs
 - Minimal packet latency variation for data-plane VNFs
- Inter-VM communication,
- Fast live migration

5.4.2 Configuration of Cyclictest

Cyclictest measures Latency of response to a stimulus. Achieving low latency with the KVM4NFV project requires setting up a special test environment. This environment includes the BIOS settings, kernel configuration, kernel parameters and the run-time environment.

- For more information regarding the test environment, please visit <https://wiki.opnfv.org/display/kvm/KVM4NFV+Test++Environment> <https://wiki.opnfv.org/display/kvm/Nfv-kvm-tuning>

5.4.2.1 Pre-configuration activities

Intel POD10 is currently used as OPNFV-KVM4NFV test environment. The rpm packages from the latest build are downloaded onto Intel-Pod10 jump server from artifact repository. Yardstick running in a ubuntu docker container on Intel Pod10-jump server will configure the host(intel pod10 node1/node2 based on job type), the guest and triggers the cyclictest on the guest using below sample yaml file.

```
For IDLE-IDLE test,

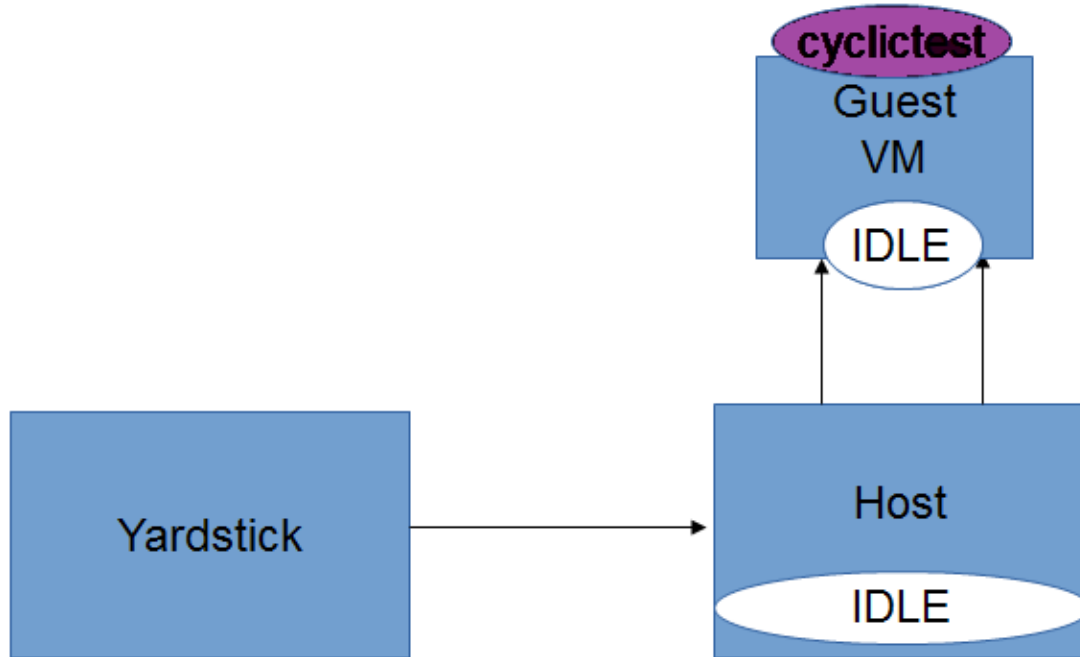
host_setup_seqs:
- "host-setup0.sh"
- "reboot"
- "host-setup1.sh"
- "host-run-qemu.sh"

guest_setup_seqs:
- "guest-setup0.sh"
```

(continues on next page)

(continued from previous page)

```
- "reboot"
- "guest-setup1.sh"
```



```
For [CPU/Memory/IO]Stress-IDLE tests,

host_setup_seqs:
- "host-setup0.sh"
- "reboot"
- "host-setup1.sh"
- "stress_daily.sh" [cpustress/memory/io]
- "host-run-qemu.sh"

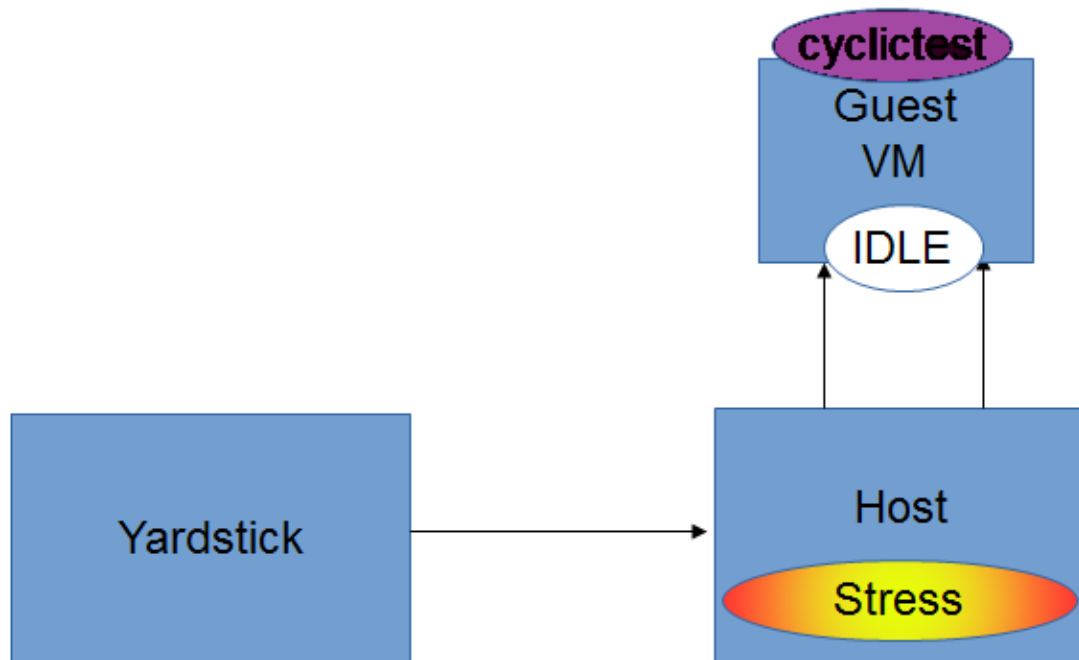
guest_setup_seqs:
- "guest-setup0.sh"
- "reboot"
- "guest-setup1.sh"
```

The following scripts are used for configuring host and guest to create a special test environment and achieve low latency.

Note: host-setup0.sh, host-setup1.sh and host-run-qemu.sh are run on the host, followed by guest-setup0.sh and guest-setup1.sh scripts on the guest VM.

host-setup0.sh: Running this script will install the latest kernel rpm on host and will make necessary changes as following to create special test environment.

- Isolates CPUs from the general scheduler
- Stops timer ticks on isolated CPUs whenever possible
- Stops RCU callbacks on isolated CPUs



- Enables intel iommu driver and disables DMA translation for devices
- Sets HugeTLB pages to 1GB
- Disables machine check
- Disables clocksource verification at runtime

host-setup1.sh: Running this script will make the following test environment changes.

- Disabling watchdogs to reduce overhead
- Disabling RT throttling
- Reroute interrupts bound to isolated CPUs to CPU 0
- Change the iptable so that we can ssh to the guest remotely

stress_daily.sh: Scripts gets triggered only for stress-idle tests. Running this script make the following environment changes.

- Triggers stress_script.sh, which runs the stress command with necessary options
- CPU,Memory or IO stress can be applied based on the test type
- Applying stress only on the Host is handled in D-Release
- For Idle-Idle test the stress script is not triggered
- Stress is applied only on the free cores to prevent load on qemu process

Note:

- On Numa Node 1: 22,23 cores are allocated for QEMU process
- 24-43 are used for applying stress

host-run-qemu.sh: Running this script will launch a guest vm on the host. Note: download guest disk image from artifactory.

guest-setup0.sh: Running this scripct on the guest vm will install the latest build kernel rpm, cyclictst and make the following configuration on guest vm.

- Isolates CPUs from the general scheduler
- Stops timer ticks on isolated CPUs whenever possible
- Uses polling idle loop to improve performance
- Disables clocksource verification at runtime

guest-setup1.sh: Running this script on guest vm will do the following configurations.

- Disable watchdogs to reduce overhead
- Routes device interrupts to non-RT CPU
- Disables RT throttling

5.4.2.2 Hardware configuration

Currently Intel POD10 is used as test environment for kvm4nfv to execute cyclictst. As part of this test environment Intel pod10-jump is configured as jenkins slave and all the latest build artifacts are downloaded on to it.

- For more information regarding hardware configuration, please visit <https://wiki.opnfv.org/display/pharos/Intel+Pod10> <https://build.opnfv.org/ci/computer/intel-pod10/> http://artifacts.opnfv.org/octopus/brahmaputra/docs/octopus_docs/opnfv-jenkins-slave-connection.html

KVM4NFV Scenarios Overview and Description

6.1 Scenario Abstract

This chapter includes detailed explanation of various scenarios files deployed as part of kvm4nfv E-Release.

6.1.1 Release Features

Scenario Name	Colorado	Danube	Euphrates
<ul style="list-style-type: none">os-nosdn-kvm-ha	Y	Y	
<ul style="list-style-type: none">os-nosdn-kvm_ovs_dpdk-noha		Y	Y
<ul style="list-style-type: none">os-nosdn-kvm_ovs_dpdk-ha		Y	Y
<ul style="list-style-type: none">os-nosdn-kvm_ovs_dpdk_bar-noha		Y	
<ul style="list-style-type: none">os-nosdn-kvm_ovs_dpdk_bar-ha		Y	

6.1.2 E- Release Scenario's overview

Scenario Name	No of Controllers	No of Computes	Plugin Names	DPDK	OVS
• os-nosdn-kvm_ovs_dpdk-noha	1	1	KVM	Y	Y
• os-nosdn-kvm_ovs_dpdk-ha	3	2	KVM	Y	Y

6.2 KVM4NFV Scenario-Description

6.2.1 Abstract

This document describes the procedure to deploy/test KVM4NFV scenarios in a nested virtualization environment. This has been verified with os-nosdn-kvm-ha, os-nosdn-kvm-noha,os-nosdn-kvm_ovs_dpdk-ha, os-nosdn-kvm_ovs_dpdk-noha and os-nosdn-kvm_ovs_dpdk_bar-ha test scenarios.

6.2.2 Version Features

Release	Features
Colorado	<ul style="list-style-type: none"> Scenario Testing feature was not part of the Colorado release of KVM4NFV
Danube	<ul style="list-style-type: none"> High Availability/No-High Availability deployment configuration of KVM4NFV software suite using Fuel Multi-node setup with 3 controller and 2 compute nodes are deployed for HA Multi-node setup with 1 controller and 3 compute nodes are deployed for NO-HA Scenarios os-nosdn-kvm_ovs_dpdk-ha, os-nosdn-kvm_ovs_dpdk_bar-ha, os-nosdn-kvm_ovs_dpdk-noha, os-nosdn-kvm_ovs_dpdk_bar-noha are supported
Euphrates	<ul style="list-style-type: none"> High Availability/No-High Availability deployment configuration of KVM4NFV software suite using Apex Multi-node setup with 3 controller and 2 compute nodes are deployed for HA Multi-node setup with 1 controller and 1 compute node are deployed for NO-HA Scenarios os-nosdn-kvm_ovs_dpdk-ha, os-nosdn-kvm_ovs_dpdk-noha, are supported

6.2.3 Introduction

The purpose of os-nosdn-kvm_ovs_dpdk-ha,os-nosdn-kvm_ovs_dpdk_bar-ha and os-nosdn-kvm_ovs_dpdk-noha,os-nosdn-kvm_ovs_dpdk_bar-noha scenarios testing is to test the High Availability/No-High Availability deployment and configuration of OPNFV software suite with OpenStack and without SDN software.

This OPNFV software suite includes OPNFV KVM4NFV latest software packages for Linux Kernel and QEMU patches for achieving low latency and also OPNFV Barometer for traffic, performance and platform monitoring.

When using Fuel installer, High Availability feature is achieved by deploying OpenStack multi-node setup with 1 Fuel-Master,3 controllers and 2 computes nodes. No-High Availability feature is achieved by deploying OpenStack multi-node setup with 1 Fuel-Master,1 controllers and 3 computes nodes.

When using Apex installer, High Availability feature is achieved by deploying Openstack multi-node setup with 1 undercloud, 3 overcloud controllers and 2 overcloud compute nodes. No-High Availability feature is achieved by deploying Openstack multi-node setup with 1 undercloud, 1 overcloud controller and 1 overcloud compute nodes.

KVM4NFV packages will be installed on compute nodes as part of deployment. The scenario testcase deploys a multi-node setup by using OPNFV Fuel and Apex deployer.

6.2.4 System pre-requisites

- RAM - Minimum 16GB
- HARD DISK - Minimum 500GB
- Linux OS installed and running
- Nested Virtualization enabled, which can be checked by,

```
$ cat /sys/module/kvm_intel/parameters/nested
Y

$ cat /proc/cpuinfo | grep vmx
```

Note: If Nested virtualization is disabled, enable it by,

```
For Ubuntu:
$ modprobe kvm_intel
$ echo Y > /sys/module/kvm_intel/parameters/nested
$ sudo reboot

For RHEL:
$ cat << EOF > /etc/modprobe.d/kvm_intel.conf
options kvm-intel nested=1
options kvm-intel enable_shadow_vmcs=1
options kvm-intel enable_apicv=1
options kvm-intel ept=1
EOF
$ cat << EOF > /etc/sysctl.d/98-rp-filter.conf
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.all.rp_filter = 0
EOF
$ sudo reboot
```

6.2.5 Environment Setup

6.2.5.1 Enable network access after the installation

For **CentOS**., Login as “root” user. After the installation complete, the Ethernet interfaces are not enabled by the default in Centos 7, you need to change the line “ONBOOT=no” to “ONBOOT=yes” in the network interface configuration file (such as ifcfg-enp6s0f0 or ifcfg-em1 ... whichever you want to connect) in /etc/sysconfig/network-scripts sub-directory. The default BOOTPROTO is dhcp in the network interface configuration file. Then use following command to enable the network access:

```
systemctl restart network
```

6.2.5.2 Configuring Proxy

For **Ubuntu**., Create an apt.conf file in /etc/apt if it doesn’t exist. Used to set proxy for apt-get if working behind a proxy server.

```
Acquire::http::proxy "http://<username>:<password>@<proxy>:<port>/" ;
Acquire::https::proxy "https://<username>:<password>@<proxy>:<port>/" ;
Acquire::ftp::proxy "ftp://<username>:<password>@<proxy>:<port>/" ;
Acquire::socks::proxy "socks://<username>:<password>@<proxy>:<port>/" ;
```

For **CentOS**., Edit /etc/yum.conf to work behind a proxy server by adding the below line.

```
$ echo "proxy=http://<username>:<password>@<proxy>:<port>/" >> /etc/yum.conf
```

6.2.5.3 Install redsocks

For **CentOS**., Since there is no redsocks package for CentOS Linux release 7.2.1511, you need build redsocks from source yourself. Using following commands to create “proxy_redsocks” sub-directory at /root:

```
cd ~
mkdir proxy_redsocks
```

Since you can’t download file at your Centos system yet. At other Centos or Ubuntu system, use following command to download redsocks source for Centos into a file “redsocks-src”;

```
wget -O redsocks-src --no-check-certificate https://github.com/darkk/redsocks/zipball/
↪master
```

Also download libevent-devel-2.0.21-4.el7.x86_64.rpm by:

```
wget ftp://fr2.rpmfind.net/linux/centos/7.2.1511/os/x86_64/Packages/libevent-devel-2.
↪0.21-4.el7.x86_64.rpm
```

Copy both redsock-src and libevent-devel-2.0.21-4.el7.x86_64.rpm files into ~/proxy_redsocks in your Centos system by “scp”.

Back to your Centos system, first install libevent-devel using libevent-devel-2.0.21-4.el7.x86_64.rpm as below:

```
cd ~/proxy_redsocks
yum install -y libevent-devel-2.0.21-4.el7.x86_64.rpm
```

Build redsocks by:

```
cd ~/proxy_redsocks
unzip redsocks-src
cd darkk-redsocks-78a73fc
yum -y install gcc
make
cp redsocks ~/proxy_redsocks/.
```

Create a redsocks.conf in ~/proxy_redsocks with following contents:

```
base {
log_debug = on;
log_info = on;
log = "file:/root/proxy.log";
daemon = on;
redirector = iptables;
}
redsocks {
local_ip = 0.0.0.0;
local_port = 6666;
// socks5 proxy server
ip = <proxy>;
port = 1080;
type = socks5;
}
redudp {
local_ip = 0.0.0.0;
local_port = 8888;
ip = <proxy>;
port = 1080;
}
dnstc {
local_ip = 127.0.0.1;
local_port = 5300;
}
```

Start redsocks service by:

```
cd ~/proxy_redsocks
./redsocks -c redsocks.conf
```

Note The redsocks service is not persistent and you need to execute the above-mentioned commands after every reboot.

Create intc-proxy.sh in ~/proxy_redsocks with following contents and make it executable by “chmod +x intc-proxy.sh”:

```
iptables -t nat -N REDSOCKS
iptables -t nat -A REDSOCKS -d 0.0.0.0/8 -j RETURN
iptables -t nat -A REDSOCKS -d 10.0.0.0/8 -j RETURN
iptables -t nat -A REDSOCKS -d 127.0.0.0/8 -j RETURN
iptables -t nat -A REDSOCKS -d 169.254.0.0/16 -j RETURN
iptables -t nat -A REDSOCKS -d 172.16.0.0/12 -j RETURN
iptables -t nat -A REDSOCKS -d 192.168.0.0/16 -j RETURN
iptables -t nat -A REDSOCKS -d 224.0.0.0/4 -j RETURN
iptables -t nat -A REDSOCKS -d 240.0.0.0/4 -j RETURN
iptables -t nat -A REDSOCKS -p tcp -j REDIRECT --to-ports 6666
iptables -t nat -A REDSOCKS -p udp -j REDIRECT --to-ports 8888
iptables -t nat -A OUTPUT -p tcp -j REDSOCKS
iptables -t nat -A PREROUTING -p tcp -j REDSOCKS
```

Enable the REDSOCKS nat chain rule by:

```
cd ~/proxy_redsocks
./intc-proxy.sh
```

Note These REDSOCKS nat chain rules are not persistent and you need to execute the above-mentioned commands after every reboot.

6.2.5.4 Network Time Protocol (NTP) setup and configuration

Install ntp by:

```
$ sudo apt-get update
$ sudo apt-get install -y ntp
```

Insert the following two lines after “server ntp.ubuntu.com” line and before “# Access control configuration; see [link](#) for” line in /etc/ntp.conf file:

```
server 127.127.1.0
fudge 127.127.1.0 stratum 10
```

Restart the ntp server to apply the changes

```
$ sudo service ntp restart
```

6.2.6 Scenario Testing

There are three ways of performing scenario testing,

- 1 Fuel
- 2 Apex
- 3 OPNFV-Playground
- 4 Jenkins Project

6.2.6.1 Fuel

1 Clone the fuel repo :

```
$ git clone https://gerrit.opnfv.org/gerrit/fuel.git
```

2 Checkout to the specific version of the branch to deploy by:

The default branch is master, to use a stable release-version use the below.,

3 Building the Fuel iso :

```
$ cd ~/fuel/ci/
$ ./build.sh -h
```

Provide the necessary options that are required to build an iso. Create a customized iso as per the deployment needs.

```
$ cd ~/fuel/build/
$ make
```

(OR) Other way is to download the latest stable fuel iso from [here](#).

```
http://artifacts.opnfv.org/fuel.html
```

4 Creating a new deployment scenario

(i). Naming the scenario file

Include the new deployment scenario yaml file in ~/fuel/deploy/scenario/. The file name should adhere to the following format:

```
<ha | no-ha>_<SDN Controller>_<feature-1>_..._<feature-n>.yaml
```

(ii). Meta data

The deployment configuration file should contain configuration metadata as stated below:

```
deployment-scenario-metadata:
  title:
  version:
  created:
```

(iii). “stack-extensions” Module

To include fuel plugins in the deployment configuration file, use the “stack-extensions” key:

```
Example:
  stack-extensions:
    - module: fuel-plugin-collectd-ceilometer
      module-config-name: fuel-barometer
      module-config-version: 1.0.0
      module-config-override:
        #module-config overrides
```

Note: The “module-config-name” and “module-config-version” should be same as the name of plugin configuration file.

The “module-config-override” is used to configure the plugin by overriding the corresponding keys in the plugin config yaml file present in ~/fuel/deploy/config/plugins/.

(iv). “dea-override-config” Module

To configure the HA/No-HA mode, network segmentation types and role to node assignments, use the “dea-override-config” key.

```
Example:
dea-override-config:
  environment:
    mode: ha
    net_segment_type: tun
  nodes:
    - id: 1
      interfaces: interfaces_1
      role: mongo,controller,opendaylight
    - id: 2
      interfaces: interfaces_1
```

(continues on next page)

(continued from previous page)

```

    role: mongo,controller
  - id: 3
    interfaces: interfaces_1
    role: mongo,controller
  - id: 4
    interfaces: interfaces_1
    role: ceph-osd,compute
  - id: 5
    interfaces: interfaces_1
    role: ceph-osd,compute
settings:
  editable:
    storage:
      ephemeral_ceph:
        description: Configures Nova to store ephemeral volumes in RBD.
        This works best if Ceph is enabled for volumes and images, too.
        Enables live migration of all types of Ceph backed VMs (without
↪this
        option, live migration will only work with VMs launched from
        Cinder volumes).
        label: Ceph RBD for ephemeral volumes (Nova)
        type: checkbox
        value: true
        weight: 75
      images_ceph:
        description: Configures Glance to use the Ceph RBD backend to
↪store
        images.If enabled, this option will prevent Swift from
↪installing.
        label: Ceph RBD for images (Glance)
        restrictions:
        - settings:storage.images_vcenter.value == true: Only one Glance
        backend could be selected.
        type: checkbox
        value: true
        weight: 30

```

Under the “dea-override-config” should provide atleast {environment:{mode:'value'},{net_segment_type:'value'}} and {nodes:1,2,... } and can also enable additional stack features such ceph,heat which overrides corresponding keys in the dea_base.yaml and dea_pod_override.yaml.

(v). “dha-override-config” Module

In order to configure the pod dha definition, use the “dha-override-config” key. This is an optional key present at the ending of the scenario file.

(vi). Mapping to short scenario name

The scenario.yaml file is used to map the short names of scenario’s to the one or more deployment scenario configuration yaml files. The short scenario names should follow the scheme below:

```

[os]-[controller]-[feature]-[mode]-[option]

[os]: mandatory
possible value: os

```

Please note that this field is needed in order to select parent jobs to list and do blocking relations between them.

```
[controller]: mandatory
example values: nosdn, ocl, odl, onos

[mode]: mandatory
possible values: ha, noha

[option]: optional
```

Used for the scenarios those do not fit into naming scheme. Optional field in the short scenario name should not be included if there is no optional scenario.

```
Example:
1. os-nosdn-kvm-noha
2. os-nosdn-kvm_ovs_dpdn_bar-ha
```

Example of how short scenario names are mapped to configuration yaml files:

```
os-nosdn-kvm_ovs_dpdn-ha:
  configfile: ha_nfv-kvm_nfv-ovs-dpdn_heat_ceilometer_scenario.yaml
```

Note:

- (-) used for separator of fields. [os-nosdn-kvm_ovs_dpdn-ha]
- (_) used to separate the values belong to the same field. [os-nosdn-kvm_ovs_bar-ha].

5 Deploying the scenario

Command to deploy the os-nosdn-kvm_ovs_dpdn-ha scenario:

```
$ cd ~/fuel/ci/
$ sudo ./deploy.sh -f -b file:///tmp/opnfv-fuel/deploy/config -l devel-pipeline -p_
↪default \
-s ha_nfv-kvm_nfv-ovs-dpdn_heat_ceilometer_scenario.yaml -i file:///tmp/opnfv.iso
```

where, -b is used to specify the configuration directory

- f is used to re-deploy on the existing deployment
- i is used to specify the image downloaded from artifacts.
- l is used to specify the lab name
- p is used to specify POD name
- s is used to specify the scenario file

Note:

```
Check $ sudo ./deploy.sh -h for further information.
```

6.2.6.2 Apex

Apex installer uses CentOS as the platform.

1 Install Packages :

Install necessary packages by following:

```
cd ~
yum install -y git rpm-build python-setuptools python-setuptools-devel
yum install -y epel-release gcc
curl -O https://bootstrap.pypa.io/get-pip.py
um install -y python3 python34
/usr/bin/python3.4 get-pip.py
yum install -y python34-devel python34-setuptools
yum install -y libffi-devel python-devel openssl-devel
yum -y install libxslt-devel libxml2-devel
```

Then you can use “dev_deploy_check.sh” in Apex installer source to install the remaining necessary packages by following:

```
cd ~
git clone https://gerrit.opnfv.org/gerrit/p/apex.git
export CONFIG=$(pwd)/apex/build
export LIB=$(pwd)/apex/lib
export PYTHONPATH=$PYTHONPATH:$(pwd)/apex/lib/python
cd ci
./dev_deploy_check.sh
yum install -y python2-oslo-config python2-debtcollector
```

2 Create ssh key :

Use following commands to create ssh key, when asked for passphrase, just enter return for empty passphrase:

```
cd ~
ssh-keygen -t rsa
```

Then prepare the authorized_keys for Apex scenario deployment:

```
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

3 Create default pool :

Use following command to default pool device:

```
cd ~
virsh pool-define /dev/stdin <<EOF
<pool type='dir'>
  <name>default</name>
  <target>
    <path>/var/lib/libvirt/images</path>
  </target>
</pool>
EOF
```

Use following commands to start and set autostart the default pool device:

```
virsh pool-start default
virsh pool-autostart default
```

Use following commands to verify the success of the creation of the default pool device and starting and setting autostart of the default pool device:

```
virsh pool-list
virsh pool-info default
```


4 Get Apex source code :

Get Apex installer source code:

```
git clone https://gerrit.opnfv.org/gerrit/p/apex.git
cd apex
```

5 Modify code to work behind proxy :

In “lib” sub-directory of Apex source, change line 284 “if ping -c 2 www.google.com > /dev/null; then” to “if curl www.google.com > /dev/null; then” in “common-functions.sh” file, since we can’t ping www.google.com behind Intel proxy.

6 Setup build environment :

Setup build environment by:

```
cd ~
export BASE=$(pwd) /apex/build
export LIB=$(pwd) /apex/lib
export PYTHONPATH=$PYTHONPATH:$(pwd) /apex/lib/python
export IMAGES=$(pwd) /apex/.build
```

7 Build Apex installer :

Build undercloud image by:

```
cd ~/apex/build
make images-clean
make undercloud
```

You can look at the targets in ~/apex/build/Makefile to build image for specific feature. Following show how to build vanilla ODL image (this can be used to build the overcloud image for basic (nosdn-nofeature) and opendaylight test scenario:

```
cd ~/apex/build
make overcloud-opendaylight
```

You can build the complete full set of images (undercloud, overcloud-full, overcloud-opendaylight, overcloud-onos) by:

```
cd ~/apex/build
make images
```

8 Modification of network_settings.yaml :

Since we are working behind proxy, we need to modify the network_settings.yaml in ~/apex/config/network to make the deployment work properly. In order to avoid checking our modification into the repo accidentally, it is recommend that you copy “network_settings.yaml” to “intc_network_settings.yaml” in the ~/apex/config/network and do following modification in intc_network_settings.yaml:

Change dns_nameservers settings from

```
dns_servers: ["8.8.8.8", "8.8.4.4"]
```

to

```
dns_servers: ["<ip-address>"]
```

Also, you need to modify `deploy.sh` in `apex/ci` from `"ntp_server="pool.ntp.org"` to `"ntp_server="<ip-address>"` to reflect that fact we couldn't reach outside NTP server, just use local time.

9 Commands to deploy scenario :

Following shows the commands used to deploy `os-nosdn-kvm_ovs_dpdk-noha` scenario behind the proxy:

```
cd ~/apex/ci
./clean.sh
./dev_deploy_check.sh
./deploy.sh -v --ping-site <ping_ip-address> --dnslookup-site <dns_ip-address> -n \
~/apex/config/network/intc_network_settings.yaml -d \
~/apex/config/deploy/os-nosdn-kvm_ovs_dpdk-noha.yaml
```

10 Accessing the Overcloud dashboard :

If the deployment completes successfully, the last few output lines from the deployment will look like the following:

```
INFO: Undercloud VM has been setup to NAT Overcloud public network
Undercloud IP: <ip-address>, please connect by doing 'opnfv-util undercloud'
Overcloud dashboard available at http://<ip-address>/dashboard
INFO: Post Install Configuration Complete
```

11 Accessing the Undercloud and Overcloud through command line :

At the end of the deployment we obtain the Undercloud ip. One can login to the Undercloud and obtain the Overcloud ip as follows:

```
cd ~/apex/ci/
./util.sh undercloud
source stackrc
nova list
ssh heat-admin@<overcloud-ip>
```

6.2.6.3 OPNFV-Playground

Install OPNFV-playground (the tool chain to deploy/test CI scenarios in `fuel@opnfv`,):

```
$ cd ~
$ git clone https://github.com/jonasbjurel/OPNFV-Playground.git
$ cd OPNFV-Playground/ci_fuel_opnfv/
```

- Follow the `README.rst` in this `~/OPNFV-Playground/ci_fuel_opnfv` sub-holder to complete all necessary installation and setup. - Section “RUNNING THE PIPELINE” in `README.rst` explain how to use this `ci_pipeline` to deploy/test CI test scenarios, you can also use

```
./ci_pipeline.sh --help ##to learn more options.
```

1 Downgrade `paramiko` package from `2.x.x` to `1.10.0`

The `paramiko` package `2.x.x` doesn't work with OPNFV-playground tool chain now, Jira ticket FUEL - 188 has been raised for the same.

Check `paramiko` package version by following below steps in your system:

```
$ python
Python 2.7.6 (default, Jun 22 2015, 17:58:13) [GCC 4.8.2] on linux2 Type "help",
->"copyright",
```

(continues on next page)

(continued from previous page)

```
"credits" or "license" for more information.
```

```
>>> import paramiko
>>> print paramiko.__version__
>>> exit()
```

You will get the current paramiko package version, if it is 2.x.x, uninstall this version by

```
$ sudo pip uninstall paramiko
```

Ubuntu 14.04 LTS has python-paramiko package (1.10.0), install it by

```
$ sudo apt-get install python-paramiko
```

Verify it by following:

```
$ python
>>> import paramiko
>>> print paramiko.__version__
>>> exit()
```

2 Clone the fuel@opnfv

Check out the specific version of specific branch of [fuel@opnfv](#)

```
$ cd ~
$ git clone https://gerrit.opnfv.org/gerrit/fuel.git
$ cd fuel
By default it will be master branch, in-order to deploy on the Colorado/Danube branch,
→ do:
$ git checkout stable/Danube
```

3 Creating the scenario

Implement the scenario file as described in 3.1.4

4 Deploying the scenario

You can use the following command to deploy/test os-nosdn kvm_ovs_dpdk-(no)ha and os-nosdn-kvm_ovs_dpdk_bar-(no)ha scenario

```
$ cd ~/OPNFV-Playground/ci_fuel_opnfv/
```

For os-nosdn-kvm_ovs_dpdk-ha :

```
$ ./ci_pipeline.sh -r ~/fuel -i /root/fuel.iso -B -n intel-sc -s os-nosdn-kvm_ovs_
→dpdk-ha
```

For os-nosdn-kvm_ovs_dpdk_bar-ha:

```
$ ./ci_pipeline.sh -r ~/fuel -i /root/fuel.iso -B -n intel-sc -s os-nosdn-kvm_ovs_
→dpdk_bar-ha
```

The “ci_pipeline.sh” first clones the local fuel repo, then deploys the os-nosdn-kvm_ovs_dpdk-ha/os-nosdn-kvm_ovs_dpdk_bar-ha scenario from the given ISO, and run Functest and Yaststick test. The log of the deployment/test (ci.log) can be found in ~/OPNFV-Playground/ci_fuel_opnfv/artifact/master/YYYY-MM-DD—HH.mm, where YYYY-MM-DD—HH.mm is the date/time you start the “ci_pipeline.sh”.

Note:

Check \$./ci_pipeline.sh -h for further information.

6.2.6.4 Jenkins Project

os-nosdn-kvm_ovs_dpdk-(no)ha and os-nosdn-kvm_ovs_dpdk_bar-(no)ha scenario can be executed from the jenkins project :

HA scenarios:

1. “fuel-os-nosdn-kvm_ovs_dpdk-ha-baremetal-daily-master” (os-nosdn-kvm_ovs_dpdk-ha)
2. “fuel-os-nosdn-kvm_ovs_dpdk_bar-ha-baremetal-daily-master” (os-nosdn-kvm_ovs_dpdk_bar-ha)
3. “apex-os-nosdn-kvm_ovs_dpdk-ha-baremetal-master” (os-nosdn-kvm_ovs_dpdk-ha)

NOHA scenarios:

1. “fuel-os-nosdn-kvm_ovs_dpdk-noha-virtual-daily-master” (os-nosdn-kvm_ovs_dpdk-noha)
2. “fuel-os-nosdn-kvm_ovs_dpdk_bar-noha-virtual-daily-master” (os-nosdn-kvm_ovs_dpdk_bar-noha)
3. “apex-os-nosdn-kvm_ovs_dpdk-noha-baremetal-master” (os-nosdn-kvm_ovs_dpdk-noha)

os-nosdn-kvm_ovs_dpdk-noha Overview and Description

7.1 os-nosdn-kvm_ovs_dpdk-noha Description

7.1.1 Introduction

The purpose of os-nosdn-kvm_ovs_dpdk-noha scenario testing is to test the No High Availability deployment and configuration of OPNFV software suite with OpenStack and without SDN software. This OPNFV software suite includes OPNFV KVM4NFV latest software packages for Linux Kernel and QEMU patches for achieving low latency. When deployed using Fuel, No High Availability feature is achieved by deploying OpenStack multi-node setup with 1 controller and 3 compute nodes and using Apex the setup is with 1 controller and 1 compute.

KVM4NFV packages will be installed on compute nodes as part of deployment. This scenario testcase deployment is happening on multi-node by using OPNFV Fuel and Apex deployer.

Using Fuel Installer

7.1.2 Scenario Components and Composition

This scenario deploys the No High Availability OPNFV Cloud based on the configurations provided in no-ha_nfv-kvm_nfv-ovs-dpdk_heat_ceilometer_scenario.yaml. This yaml file contains following configurations and is passed as an argument to deploy.py script

- `scenario.yaml`: This configuration file defines translation between a short deployment scenario name(os-nosdn-kvm_ovs_dpdk-noha) and an actual deployment scenario configuration file(no-ha_nfv-kvm_nfv-ovs-dpdk_heat_ceilometer_scenario.yaml)
- `deployment-scenario-metadata`: Contains the configuration metadata like title,version,created,comment.

```
deployment-scenario-metadata:
  title: NFV KVM and OVS-DPDK NOHA deployment
  version: 0.0.1
```

(continues on next page)

(continued from previous page)

```
created: Dec 20 2016
comment: NFV KVM and OVS-DPDK
```

- **stack-extensions:** Stack extensions are opnfv added value features in form of a fuel-plugin. Plugins listed in stack extensions are enabled and configured. os-nosdn-kvm_ovs_dpkg-noha scenario currently uses KVM-1.0.0 plugin.

```
stack-extensions:
- module: fuel-plugin-kvm
  module-config-name: fuel-nfvkvm
  module-config-version: 1.0.0
  module-config-override:
    # Module config overrides
```

- **dea-override-config:** Used to configure the NO-HA mode, network segmentation types and role to node assignments. These configurations overrides corresponding keys in the dea_base.yaml and dea_pod_override.yaml. These keys are used to deploy multiple nodes (1 controller, 3 computes) as mention below.

- **Node 1:**

- * This node has MongoDB and Controller roles
 - * The controller node runs the Identity service, Image Service, management portions of Compute and Networking, Networking plug-in and the dashboard
 - * Uses VLAN as an interface

- **Node 2:**

- * This node has compute and Ceph-osd roles
 - * Ceph is a massively scalable, open source, distributed storage system
 - * By default, Compute uses KVM as the hypervisor
 - * Uses DPDK as an interface

- **Node 3:**

- * This node has compute and Ceph-osd roles
 - * Ceph is a massively scalable, open source, distributed storage system
 - * By default, Compute uses KVM as the hypervisor
 - * Uses DPDK as an interface

- **Node 4:**

- * This node has compute and Ceph-osd roles
 - * Ceph is a massively scalable, open source, distributed storage system
 - * By default, Compute uses KVM as the hypervisor
 - * Uses DPDK as an interface

The below is the dea-override-config of the no-ha_nfv-kvm_nfv-ovs-dpkg_heat_ceilometer_scenario.yaml file.

```

dea-override-config:
  fuel:
    FEATURE_GROUPS:
      - experimental
  environment:
    net_segment_type: vlan
  nodes:
    - id: 1
      interfaces: interfaces_vlan
      role: mongo,controller
    - id: 2
      interfaces: interfaces_dpdk
      role: ceph-osd,compute
      attributes: attributes_1
    - id: 3
      interfaces: interfaces_dpdk
      role: ceph-osd,compute
      attributes: attributes_1
    - id: 4
      interfaces: interfaces_dpdk
      role: ceph-osd,compute
      attributes: attributes_1

  attributes_1:
    hugepages:
      dpdk:
        value: 1024
      nova:
        value:
          '2048': 1024

  network:
    networking_parameters:
      segmentation_type: vlan
    networks:
      - cidr: null
        gateway: null
        ip_ranges: []
        meta:
          configurable: false
          map_priority: 2
          name: private
          neutron_vlan_range: true
          notation: null
          render_addr_mask: null
          render_type: null
          seg_type: vlan
          use_gateway: false
          vlan_start: null
          name: private
          vlan_start: null

  settings:
    editable:
      storage:
        ephemeral_ceph:
          description: Configures Nova to store ephemeral volumes in RBD. This works,
↪best if Ceph

```

(continues on next page)

(continued from previous page)

```

    is enabled for volumes and images, too. Enables live migration of all types.
↳of Ceph
    backed VMs (without this option, live migration will only work with VMs
↳launched from
    Cinder volumes).
    label: Ceph RBD for ephemeral volumes (Nova)
    type: checkbox
    value: true
    weight: 75
    images_ceph:
    description: Configures Glance to use the Ceph RBD backend to store images.
↳If enabled,
    this option will prevent Swift from installing.
    label: Ceph RBD for images (Glance)
    restrictions:
    - settings:storage.images_vcenter.value == true: Only one Glance backend
↳could be selected.
    type: checkbox
    value: true
    weight: 30

```

- `dha-override-config`: Provides information about the VM definition and Network config for virtual deployment. These configurations overrides the pod dha definition and points to the controller, compute and fuel definition files. The `no-ha_nfv-kvm_nfv-ovs-dpdk_heat_ceilometer_scenario.yaml` has no dha-config changes i.e., default configuration is used.
- `os-nosdn-kvm_ovs_dpdk-noha` scenario is successful when all the 4 Nodes are accessible, up and running.

Note:

- In `os-nosdn-kvm_ovs_dpdk-noha` scenario, OVS is installed on the compute nodes with DPDK configured
- Hugepages for DPDK are configured in the `attributes_1` section of the

`no-ha_nfv-kvm_nfv-ovs-dpdk_heat_ceilometer_scenario.yaml`

- Hugepages are only configured for compute nodes
- This results in faster communication and data transfer among the compute nodes

7.1.3 Scenario Usage Overview

- The high availability feature is disabled and deployment is done by `deploy.py` with `noha_nfv-kvm_nfv-ovs-dpdk_heat_ceilometer_scenario.yaml` as an argument.
- Install Fuel Master and deploy OPNFV Cloud from scratch on Hardware Environment:

Command to deploy the `os-nosdn-kvm_ovs_dpdk-noha` scenario:

```

$ cd ~/fuel/ci/
$ sudo ./deploy.sh -f -b file:///tmp/opnfv-fuel/deploy/config -l devel-pipeline -p
↳default \
-s no-ha_nfv-kvm_nfv-ovs-dpdk_heat_ceilometer_scenario.yaml -i file:///tmp/opnfv.iso

```

where, `-b` is used to specify the configuration directory

`-i` is used to specify the image downloaded from artifacts.

Note:


```
Check $ sudo ./deploy.sh -h for further information.
```

- os-nosdn-kvm_ovs_dpdk-noha scenario can be executed from the jenkins project “fuel-os-nosdn-kvm_ovs_dpdk-noha-baremetal-daily-master”
- This scenario provides the No High Availability feature by deploying 1 controller, 3 compute nodes and checking if all the 4 nodes are accessible (IP, up & running).
- Test Scenario is passed if deployment is successful and all 4 nodes have accessibility (IP, up & running).

Using Apex Installer

7.1.4 Scenario Components and Composition

This scenario is composed of common OpenStack services enabled by default, including Nova, Neutron, Glance, Cinder, Keystone, Horizon. Optionally and by default, Tacker and Congress services are also enabled. Ceph is used as the backend storage to Cinder on all deployed nodes.

The os-nosdn-kvm_ovs_dpdk-noha.yaml file contains following configurations and is passed as an argument to deploy.sh script.

- `global-params`: Used to define the global parameter and there is only one such parameter exists, i.e., `ha_enabled`

```
global-params:
  ha_enabled: false
```

- `deploy_options`: Used to define the type of SDN controller, configure the tacker, congress, service functioning chaining support (sfc) for ODL and ONOS, configure ODL with SDNVPN support, which dataplane to use for overcloud tenant networks, whether to run the kvm real time kernel (rt_kvm) in the compute node(s) to reduce the network latencies caused by network function virtualization and whether to install and configure fdio functionality in the overcloud

```
deploy_options:
  sdn_controller: false
  tacker: true
  congress: true
  sfc: false
  vpn: false
  rt_kvm: true
  dataplane: ovs_dpdk
```

- `performance`: Used to set performance options on specific roles. The valid roles are ‘Compute’, ‘Controller’ and ‘Storage’, and the valid sections are ‘kernel’ and ‘nova’

```
performance:
  Controller:
    kernel:
      hugepages: 1024
      hugepagesz: 2M
  Compute:
    kernel:
      hugepagesz: 2M
      hugepages: 2048
      intel_iommu: 'on'
      iommu: pt
  ovs:
```

(continues on next page)

(continued from previous page)

```
socket_memory: 1024
pmd_cores: 2
dpdk_cores: 1
```

7.1.5 Scenario Usage Overview

- The high availability feature can be achieved by executing `deploy.sh` with `os-nosdn-kvm_ovs_dpdk-noha.yaml` as an argument.
- Build the undercloud and overcloud images as mentioned below:

```
cd ~/apex/build/
make images-clean
make images
```

- Command to deploy `os-nosdn-kvm_ovs_dpdk-noha` scenario:

```
cd ~/apex/ci/
./clean.sh
./dev_dep_check.sh
./deploy.sh -v --ping-site <ping_ip-address> --dnslookup-site <dns_ip-address> -n \
~/apex/config/network/intc_network_settings.yaml -d ~/apex/config/deploy/os-nosdn-kvm_
↪ ovs_dpdk-noha.yaml
```

where, `-v` is used for virtual deployment `-n` is used for providing the network configuration file `-d` is used for providing the scenario configuration file

7.1.6 References

For more information on the OPNFV Euphrates release, please visit <http://www.opnfv.org/Euphrates>

os-nosdn-kvm_ovs_dpdk-ha Overview and Description

8.1 os-nosdn-kvm_ovs_dpdk-ha Description

8.1.1 Introduction

The purpose of os-nosdn-kvm_ovs_dpdk-ha scenario testing is to test the High Availability deployment and configuration of OPNFV software suite with OpenStack and without SDN software. This OPNFV software suite includes OPNFV KVM4NFV latest software packages for Linux Kernel and QEMU patches for achieving low latency. High Availability feature is achieved by deploying OpenStack multi-node setup with 3 controllers and 2 compute nodes.

KVM4NFV packages will be installed on compute nodes as part of deployment. This scenario testcase deployment is happening on multi-node by using OPNFV Fuel and Apex deployer.

Using Fuel Installer

8.1.2 Scenario Components and Composition

This scenario deploys the High Availability OPNFV Cloud based on the configurations provided in ha_nfv-kvm_nfv-ovs-dpdk_heat_ceilometer_scenario.yaml. This yaml file contains following configurations and is passed as an argument to deploy.py script

- `scenario.yaml`: This configuration file defines translation between a short deployment scenario name(os-nosdn-kvm_ovs_dpdk-ha) and an actual deployment scenario configuration file(ha_nfv-kvm_nfv-ovs-dpdk_heat_ceilometer_scenario.yaml)
- `deployment-scenario-metadata`: Contains the configuration metadata like title,version,created,comment.

```
deployment-scenario-metadata:
  title: NFV KVM and OVS-DPDK HA deployment
  version: 0.0.1
  created: Dec 20 2016
  comment: NFV KVM and OVS-DPDK
```

- **stack-extensions**: Stack extensions are opnfv added value features in form of a fuel-plugin. Plugins listed in stack extensions are enabled and configured. os-nosdn-kvm_ovs_dpdk-ha scenario currently uses KVM-1.0.0 plugin.

```
stack-extensions:
- module: fuel-plugin-kvm
  module-config-name: fuel-nfvkvm
  module-config-version: 1.0.0
  module-config-override:
    # Module config overrides
```

- **dea-override-config**: Used to configure the HA mode, network segmentation types and role to node assignments. These configurations overrides corresponding keys in the dea_base.yaml and dea_pod_override.yaml. These keys are used to deploy multiple nodes (3 controllers, 2 computes) as mention below.
 - **Node 1:**
 - * This node has MongoDB and Controller roles
 - * The controller node runs the Identity service, Image Service, management portions of Compute and Networking, Networking plug-in and the dashboard
 - * Uses VLAN as an interface
 - **Node 2:**
 - * This node has Ceph-osd and Controller roles
 - * The controller node runs the Identity service, Image Service, management portions of Compute and Networking, Networking plug-in and the dashboard
 - * Ceph is a massively scalable, open source, distributed storage system
 - * Uses VLAN as an interface
 - **Node 3:**
 - * This node has Controller role in order to achieve high availability.
 - * Uses VLAN as an interface
 - **Node 4:**
 - * This node has compute and Ceph-osd roles
 - * Ceph is a massively scalable, open source, distributed storage system
 - * By default, Compute uses KVM as the hypervisor
 - * Uses DPDK as an interface
 - **Node 5:**
 - * This node has compute and Ceph-osd roles
 - * Ceph is a massively scalable, open source, distributed storage system
 - * By default, Compute uses KVM as the hypervisor
 - * Uses DPDK as an interface

The below is the dea-override-config of the ha_nfv-kvm_nfv-ovs-dpdk_heat_ceilometer_scenario.yaml file.

```

dea-override-config:
  fuel:
    FEATURE_GROUPS:
      - experimental
  nodes:
    - id: 1
      interfaces: interfaces_1
      role: controller
    - id: 2
      interfaces: interfaces_1
      role: mongo,controller
    - id: 3
      interfaces: interfaces_1
      role: ceph-osd,controller
    - id: 4
      interfaces: interfaces_dpkg
      role: ceph-osd,compute
      attributes: attributes_1
    - id: 5
      interfaces: interfaces_dpkg
      role: ceph-osd,compute
      attributes: attributes_1

  attributes_1:
    hugepages:
      dpdk:
        value: 1024
      nova:
        value:
          '2048': 1024

  settings:
    editable:
      storage:
        ephemeral_ceph:
          description: Configures Nova to store ephemeral volumes in RBD. This works
↪best if Ceph
is enabled for volumes and images, too. Enables live migration of all types
↪of Ceph
backed VMs (without this option, live migration will only work with VMs
↪launched from
Cinder volumes).
          label: Ceph RBD for ephemeral volumes (Nova)
          type: checkbox
          value: true
          weight: 75
        images_ceph:
          description: Configures Glance to use the Ceph RBD backend to store images.
↪If enabled,
          this option will prevent Swift from installing.
          label: Ceph RBD for images (Glance)
          restrictions:
            - settings:storage.images_vcenter.value == true: Only one Glance backend
↪could be selected.
          type: checkbox
          value: true
          weight: 30

```

- `dha-override-config`: Provides information about the VM definition and Network config for virtual deployment. These configurations override the pod dha definition and points to the controller, compute and fuel definition files.

The below is the `dha-override-config` of the `ha_nfv-kvm_nfv-ovs-dpdk_heat_ceilometer_scenario.yaml` file.

```
dha-override-config:
  nodes:
  - id: 1
    libvirtName: controller1
    libvirtTemplate: templates/virtual_environment/vms/controller.xml
  - id: 2
    libvirtName: controller2
    libvirtTemplate: templates/virtual_environment/vms/controller.xml
  - id: 3
    libvirtName: controller3
    libvirtTemplate: templates/virtual_environment/vms/controller.xml
  - id: 4
    libvirtName: compute1
    libvirtTemplate: templates/virtual_environment/vms/compute.xml
  - id: 5
    libvirtName: compute2
    libvirtTemplate: templates/virtual_environment/vms/compute.xml
  - id: 6
    libvirtName: fuel-master
    libvirtTemplate: templates/virtual_environment/vms/fuel.xml
    isFuel: yes
    username: root
    password: r00tme
```

- `os-nosdn-kvm_ovs_dpdk-ha` scenario is successful when all the 5 Nodes are accessible, up and running.

Note:

- In `os-nosdn-kvm_ovs_dpdk-ha` scenario, OVS is installed on the compute nodes with DPDK configured
- Hugepages for DPDK are configured in the `attributes_1` section of the

`no-ha_nfv-kvm_nfv-ovs-dpdk_heat_ceilometer_scenario.yaml`

- Hugepages are only configured for compute nodes
- This results in faster communication and data transfer among the compute nodes

8.1.3 Scenario Usage Overview

- The high availability feature can be achieved by executing `deploy.py` with `ha_nfv-kvm_nfv-ovs-dpdk_heat_ceilometer_scenario.yaml` as an argument.
- Install Fuel Master and deploy OPNFV Cloud from scratch on Hardware Environment:

Command to deploy the `os-nosdn-kvm_ovs_dpdk-ha` scenario:

```
$ cd ~/fuel/ci/
$ sudo ./deploy.sh -f -b file:///tmp/opnfv-fuel/deploy/config -l devel-pipeline -p_
↪ default \
-s ha_nfv-kvm_nfv-ovs-dpdk_heat_ceilometer_scenario.yaml -i file:///tmp/opnfv.iso
```

where, `-b` is used to specify the configuration directory

-i is used to specify the image downloaded from artifacts.

Note:

```
Check $ sudo ./deploy.sh -h for further information.
```

- os-nosdn-kvm_ovs_dpdk-ha scenario can be executed from the jenkins project “fuel-os-nosdn-kvm_ovs_dpdk-ha-baremetal-daily-master”
- This scenario provides the High Availability feature by deploying 3 controller,2 compute nodes and checking if all the 5 nodes are accessible(IP,up & running).
- Test Scenario is passed if deployment is successful and all 5 nodes have accessibility (IP , up & running).

Using Apex Installer

8.1.4 Scenario Components and Composition

This scenario is composed of common OpenStack services enabled by default, including Nova, Neutron, Glance, Cinder, Keystone, Horizon. Optionally and by default, Tacker and Congress services are also enabled. Ceph is used as the backend storage to Cinder on all deployed nodes.

All services are in HA, meaning that there are multiple cloned instances of each service, and they are balanced by HA Proxy using a Virtual IP Address per service.

The os-nosdn-kvm_ovs_dpdk-ha.yaml file contains following configurations and is passed as an argument to deploy.sh script.

- `global-params`: Used to define the global parameter and there is only one such parameter exists,i.e, `ha_enabled`

```
global-params:
  ha_enabled: true
```

- `deploy_options`: Used to define the type of SDN controller, configure the tacker, congress, service functioning chaining support(sfc) for ODL and ONOS, configure ODL with SDNVPN support, which dataplane to use for overcloud tenant networks, whether to run the kvm real time kernel (rt_kvm) in the compute node(s) to reduce the network latencies caused by network function virtualization and whether to install and configure fdio functionality in the overcloud

```
deploy_options:
  sdn_controller: false
  tacker: true
  congress: true
  sfc: false
  vpn: false
  rt_kvm: true
  dataplane: ovs_dpdk
```

- `performance`: Used to set performance options on specific roles. The valid roles are ‘Compute’, ‘Controller’ and ‘Storage’, and the valid sections are ‘kernel’ and ‘nova’

```
performance:
  Controller:
    kernel:
      hugepages: 1024
      hugepagesz: 2M
  Compute:
```

(continues on next page)

(continued from previous page)

```
kernel:
  hugepagesz: 2M
  hugepages: 2048
  intel_iommu: 'on'
  iommu: pt
ovs:
  socket_memory: 1024
  pmd_cores: 2
  dpdk_cores: 1
```

8.1.5 Scenario Usage Overview

- The high availability feature can be achieved by executing `deploy.sh` with `os-nosdn-kvm_ovs_dpdk-ha.yaml` as an argument.
- Build the undercloud and overcloud images as mentioned below:

```
cd ~/apex/build/
make images-clean
make images
```

- Command to deploy `os-nosdn-kvm_ovs_dpdk-ha` scenario:

```
cd ~/apex/ci/
./clean.sh
./dev_dep_check.sh
./deploy.sh -v --ping-site <ping_ip-address> --dnslookup-site <dns_ip-address> -n \
~/apex/config/network/intc_network_settings.yaml -d ~/apex/config/deploy/os-nosdn-kvm_
↪ ovs_dpdk-ha.yaml
```

where, `-v` is used for virtual deployment `-n` is used for providing the network configuration file `-d` is used for providing the scenario configuration file

8.1.6 References

For more information on the OPNFV Euphrates release, please visit <http://www.opnfv.org/Euphrates>

os-nosdn-kvm_ovs_dpdk_bar-noha Overview and Description

9.1 os-nosdn-kvm_ovs_dpdk_bar-ha Description

9.1.1 Introduction

The purpose of os-nosdn-kvm_ovs_dpdk_bar-noha scenario testing is to test the No High Availability deployment and configuration of OPNFV software suite with OpenStack and without SDN software. This OPNFV software suite includes OPNFV KVM4NFV latest software packages for Linux Kernel and QEMU patches for achieving low latency.No High Availability feature is achieved by deploying OpenStack multi-node setup with 1 controller and 3 computes nodes.

OPNFV Barometer packages is used for traffic,performance and platform monitoring. KVM4NFV packages will be installed on compute nodes as part of deployment. This scenario testcase deployment is happening on multi-node by using OPNFV Fuel deployer.

9.1.2 Scenario Components and Composition

This scenario deploys the No High Availability OPNFV Cloud based on the configurations provided in no-ha_nfv-kvm_nfv-ovs-dpdk-bar_heat_ceilometer_scenario.yaml. This yaml file contains following configurations and is passed as an argument to deploy.py script

- `scenario.yaml`: This configuration file defines translation between a short deployment scenario name(os-nosdn-kvm_ovs_dpdk_bar-noha) and an actual deployment scenario configuration file(no-ha_nfv-kvm_nfv-ovs-dpdk-bar_heat_ceilometer_scenario.yaml)
- `deployment-scenario-metadata`: Contains the configuration metadata like title,version,created,comment.

```
deployment-scenario-metadata:
  title: NFV KVM and OVS-DPDK HA deployment
  version: 0.0.1
  created: Dec 20 2016
  comment: NFV KVM and OVS-DPDK
```

- `stack-extensions`: Stack extensions are opnfv added value features in form of a fuel-plugin. Plugins listed in stack extensions are enabled and configured. `os-nosdn-kvm_ovs_dpdk_bar-noha` scenario currently uses KVM-1.0.0 plugin and barometer-1.0.0 plugin.

```
stack-extensions:
- module: fuel-plugin-kvm
  module-config-name: fuel-nfvkvm
  module-config-version: 1.0.0
  module-config-override:
    # Module config overrides
- module: fuel-plugin-collectd-ceilometer
  module-config-name: fuel-barometer
  module-config-version: 1.0.0
  module-config-override:
    # Module config overrides
```

- `dea-override-config`: Used to configure the HA mode, network segmentation types and role to node assignments. These configurations overrides corresponding keys in the `dea_base.yaml` and `dea_pod_override.yaml`. These keys are used to deploy multiple nodes (1 controller, 3 computes) as mention below.

- **Node 1:**

- * This node has MongoDB and Controller roles
- * The controller node runs the Identity service, Image Service, management portions of Compute and Networking, Networking plug-in and the dashboard
- * Uses VLAN as an interface

- **Node 2:**

- * This node has compute and Ceph-osd roles
- * Ceph is a massively scalable, open source, distributed storage system
- * By default, Compute uses KVM as the hypervisor
- * Uses DPDK as an interface

- **Node 3:**

- * This node has compute and Ceph-osd roles
- * Ceph is a massively scalable, open source, distributed storage system
- * By default, Compute uses KVM as the hypervisor
- * Uses DPDK as an interface

- **Node 4:**

- * This node has compute and Ceph-osd roles
- * Ceph is a massively scalable, open source, distributed storage system
- * By default, Compute uses KVM as the hypervisor
- * Uses DPDK as an interface

The below is the `dea-override-config` of the `no-ha_nfv-kvm_nfv-ovs-dpdk-bar_heat_ceilometer_scenario.yaml` file.

```
dea-override-config:
fuel:
  FEATURE_GROUPS:
```

(continues on next page)

(continued from previous page)

```

- experimental
environment:
  net_segment_type: vlan
nodes:
- id: 1
  interfaces: interfaces_vlan
  role: mongo,controller
- id: 2
  interfaces: interfaces_dpdk
  role: ceph-osd,compute
  attributes: attributes_1
- id: 3
  interfaces: interfaces_dpdk
  role: ceph-osd,compute
  attributes: attributes_1
- id: 4
  interfaces: interfaces_dpdk
  role: ceph-osd,compute
  attributes: attributes_1

attributes_1:
  hugepages:
    dpdk:
      value: 1024
    nova:
      value:
        '2048': 1024

network:
  networking_parameters:
    segmentation_type: vlan
  networks:
  - cidr: null
    gateway: null
    ip_ranges: []
    meta:
      configurable: false
      map_priority: 2
      name: private
      neutron_vlan_range: true
      notation: null
      render_addr_mask: null
      render_type: null
      seg_type: vlan
      use_gateway: false
      vlan_start: null
    name: private
    vlan_start: null

settings:
  editable:
    storage:
      ephemeral_ceph:
        description: Configures Nova to store ephemeral volumes in RBD. This works
↳best if Ceph
        is enabled for volumes and images, too. Enables live migration of all types
↳of Ceph

```

(continues on next page)

(continued from previous page)

```

        backed VMs (without this option, live migration will only work with VMs_
↳launched from
        Cinder volumes).
        label: Ceph RBD for ephemeral volumes (Nova)
        type: checkbox
        value: true
        weight: 75
    images_ceph:
        description: Configures Glance to use the Ceph RBD backend to store images._
↳If enabled,
        this option will prevent Swift from installing.
        label: Ceph RBD for images (Glance)
        restrictions:
        - settings:storage.images_vcenter.value == true: Only one Glance backend_
↳could be selected.
        type: checkbox
        value: true
        weight: 30

```

- `dha-override-config`: Provides information about the VM definition and Network config for virtual deployment. These configurations override the pod dha definition and points to the controller, compute and fuel definition files. The `noha_nfv-kvm_nfv-ovs-dpdk-bar_heat_ceilometer_scenario.yaml` has no dha-config changes i.e., default configuration is used.
- `os-nosdn-kvm_ovs_dpdk_bar-noha` scenario is successful when all the 4 Nodes are accessible, up and running.

Note:

- In `os-nosdn-kvm_ovs_dpdk_bar-noha` scenario, OVS is installed on the compute nodes with DPDK configured
- Barometer plugin is also implemented along with KVM plugin.
- Hugepages for DPDK are configured in the `attributes_1` section of the `no-ha_nfv-kvm_nfv-ovs-dpdk_bar_heat_ceilometer_scenario.yaml`
- Hugepages are only configured for compute nodes
- This results in faster communication and data transfer among the compute nodes

9.1.3 Scenario Usage Overview

- The high availability feature is disabled and deployment is done by `deploy.py` with `noha_nfv-kvm_nfv-ovs-dpdk_bar_heat_ceilometer_scenario.yaml` as an argument.
- Install Fuel Master and deploy OPNFV Cloud from scratch on Hardware Environment:

Command to deploy the `os-nosdn-kvm_ovs_dpdk_bar-noha` scenario:

```

$ cd ~/fuel/ci/
$ sudo ./deploy.sh -f -b file:///tmp/opnfv-fuel/deploy/config -l devel-pipeline -p_
↳default \
-s no-ha_nfv-kvm_nfv-ovs-dpdk_bar_heat_ceilometer_scenario.yaml -i file:///tmp/opnfv.
↳iso

```

where, `-b` is used to specify the configuration directory

`-i` is used to specify the image downloaded from artifacts.

Note:

Check \$ sudo ./deploy.sh -h for further information.

- os-nosdn-kvm_ovs_dpdk_bar-noha scenario can be executed from the jenkins project “fuel-os-nosdn-kvm_ovs_dpdk_bar-noha-baremetal-daily-master”
- This scenario provides the No High Availability feature by deploying 1 controller,3 compute nodes and checking if all the 4 nodes are accessible(IP,up & running).
- Test Scenario is passed if deployment is successful and all 4 nodes have accessibility (IP , up & running).

9.1.4 Known Limitations, Issues and Workarounds

- Test scenario os-nosdn-kvm_ovs_dpdk_bar-noha result is not stable.

9.1.5 References

For more information on the OPNFV Euphrates release, please visit <http://www.opnfv.org/Euphrates>

os-nosdn-kvm_ovs_dpdk_bar-ha Overview and Description

10.1 os-nosdn-kvm_ovs_dpdk_bar-ha Description

10.1.1 Introduction

The purpose of os-nosdn-kvm_ovs_dpdk_bar-ha scenario testing is to test the High Availability deployment and configuration of OPNFV software suite with OpenStack and without SDN software. This OPNFV software suite includes OPNFV KVM4NFV latest software packages for Linux Kernel and QEMU patches for achieving low latency. High Availability feature is achieved by deploying OpenStack multi-node setup with 3 controllers and 2 computes nodes.

OPNFV Barometer packages is used for traffic, performance and platform monitoring. KVM4NFV packages will be installed on compute nodes as part of deployment. This scenario testcase deployment is happening on multi-node by using OPNFV Fuel deployer.

10.1.2 Scenario Components and Composition

This scenario deploys the High Availability OPNFV Cloud based on the configurations provided in ha_nfv-kvm_nfv-ovs-dpdk-bar_heat_ceilometer_scenario.yaml. This yaml file contains following configurations and is passed as an argument to deploy.py script

- `scenario.yaml`: This configuration file defines translation between a short deployment scenario name(os-nosdn-kvm_ovs_dpdk_bar-ha) and an actual deployment scenario configuration file(ha_nfv-kvm_nfv-ovs-dpdk-bar_heat_ceilometer_scenario.yaml)
- `deployment-scenario-metadata`: Contains the configuration metadata like title, version, created, comment.

```
deployment-scenario-metadata:
  title: NFV KVM and OVS-DPDK HA deployment
  version: 0.0.1
  created: Dec 20 2016
  comment: NFV KVM and OVS-DPDK
```

- `stack-extensions`: Stack extensions are opnfv added value features in form of a fuel-plugin. Plugins listed in stack extensions are enabled and configured. `os-nosdn-kvm_ovs_dpdk_bar-ha` scenario currently uses KVM-1.0.0 plugin and barometer plugin.

```
stack-extensions:
- module: fuel-plugin-kvm
  module-config-name: fuel-nfvkvm
  module-config-version: 1.0.0
  module-config-override:
    # Module config overrides
- module: fuel-plugin-collectd-ceilometer
  module-config-name: fuel-barometer
  module-config-version: 1.0.0
  module-config-override:
    # Module config overrides
```

- `dea-override-config`: Used to configure the HA mode, network segmentation types and role to node assignments. These configurations overrides corresponding keys in the `dea_base.yaml` and `dea_pod_override.yaml`. These keys are used to deploy multiple nodes (3 controllers, 2 computes) as mention below.

- **Node 1:**

- * This node has MongoDB and Controller roles
- * The controller node runs the Identity service, Image Service, management portions of Compute and Networking, Networking plug-in and the dashboard
- * Uses VLAN as an interface

- **Node 2:**

- * This node has Ceph-osd and Controller roles
- * The controller node runs the Identity service, Image Service, management portions of Compute and Networking, Networking plug-in and the dashboard
- * Ceph is a massively scalable, open source, distributed storage system
- * Uses VLAN as an interface

- **Node 3:**

- * This node has Controller role in order to achieve high availability.
- * Uses VLAN as an interface

- **Node 4:**

- * This node has compute and Ceph-osd roles
- * Ceph is a massively scalable, open source, distributed storage system
- * By default, Compute uses KVM as the hypervisor
- * Uses DPDK as an interface

- **Node 5:**

- * This node has compute and Ceph-osd roles
- * Ceph is a massively scalable, open source, distributed storage system
- * By default, Compute uses KVM as the hypervisor
- * Uses DPDK as an interface

The below is the dea-override-config of the ha_nfv-kvm_nfv-ovs-dpdk-bar_heat_ceilometer_scenario.yaml file.

```
dea-override-config:
  fuel:
    FEATURE_GROUPS:
      - experimental
  nodes:
    - id: 1
      interfaces: interfaces_1
      role: controller
    - id: 2
      interfaces: interfaces_1
      role: mongo,controller
    - id: 3
      interfaces: interfaces_1
      role: ceph-osd,controller
    - id: 4
      interfaces: interfaces_dpdk
      role: ceph-osd,compute
      attributes: attributes_1
    - id: 5
      interfaces: interfaces_dpdk
      role: ceph-osd,compute
      attributes: attributes_1

  attributes_1:
    hugepages:
      dpdk:
        value: 1024
      nova:
        value:
          '2048': 1024

  settings:
    editable:
      storage:
        ephemeral_ceph:
          description: Configures Nova to store ephemeral volumes in RBD. This works
↪best if Ceph
          is enabled for volumes and images, too. Enables live migration of all types
↪of Ceph
          backed VMs (without this option, live migration will only work with VMs
↪launched from
          Cinder volumes).
          label: Ceph RBD for ephemeral volumes (Nova)
          type: checkbox
          value: true
          weight: 75
        images_ceph:
          description: Configures Glance to use the Ceph RBD backend to store images.
↪If enabled,
          this option will prevent Swift from installing.
          label: Ceph RBD for images (Glance)
          restrictions:
            - settings:storage.images_vcenter.value == true: Only one Glance backend
↪could be selected.
          type: checkbox
```

(continues on next page)

(continued from previous page)

```
value: true
weight: 30
```

- `dha-override-config`: Provides information about the VM definition and Network config for virtual deployment. These configurations overrides the pod dha definition and points to the controller, compute and fuel definition files.

The below is the `dha-override-config` of the `ha_nfv-kvm_nfv-ovs-dpdk-bar_heat_ceilometer_scenario.yaml` file.

```
dha-override-config:
  nodes:
  - id: 1
    libvirtName: controller1
    libvirtTemplate: templates/virtual_environment/vms/controller.xml
  - id: 2
    libvirtName: controller2
    libvirtTemplate: templates/virtual_environment/vms/controller.xml
  - id: 3
    libvirtName: controller3
    libvirtTemplate: templates/virtual_environment/vms/controller.xml
  - id: 4
    libvirtName: compute1
    libvirtTemplate: templates/virtual_environment/vms/compute.xml
  - id: 5
    libvirtName: compute2
    libvirtTemplate: templates/virtual_environment/vms/compute.xml
  - id: 6
    libvirtName: fuel-master
    libvirtTemplate: templates/virtual_environment/vms/fuel.xml
    isFuel: yes
    username: root
    password: r00tme
```

- `os-nosdn-kvm_ovs_dpdk_bar-ha` scenario is successful when all the 5 Nodes are accessible, up and running.

Note:

- In `os-nosdn-kvm_ovs_dpdk_bar-ha` scenario, OVS is installed on the compute nodes with DPDK configured
- Baraometer plugin is also implemented along with KVM plugin
- Hugepages for DPDK are configured in the `attributes_1` section of the

`no-ha_nfv-kvm_nfv-ovs-dpdk_heat_ceilometer_scenario.yaml`

- Hugepages are only configured for compute nodes
- This results in faster communication and data transfer among the compute nodes

10.1.3 Scenario Usage Overview

- The high availability feature can be achieved by executing `deploy.py` with `ha_nfv-kvm_nfv-ovs-dpdk-bar_heat_ceilometer_scenario.yaml` as an argument.
- Install Fuel Master and deploy OPNFV Cloud from scratch on Hardware Environment:

Command to deploy the `os-nosdn-kvm_ovs_dpdk_bar-ha` scenario:

```
$ cd ~/fuel/ci/
$ sudo ./deploy.sh -f -b file:///tmp/opnfv-fuel/deploy/config -l devel-pipeline -p_
↪default \
-s ha_nfv-kvm_nfv-ovs-dpdk-bar_heat_ceilometer_scenario.yaml -i file:///tmp/opnfv.iso
```

where, -b is used to specify the configuration directory

-i is used to specify the image downloaded from artifacts.

Note:

Check `$ sudo ./deploy.sh -h` for further information.

- os-nosdn-kvm_ovs_dpdk_bar-ha scenario can be executed from the jenkins project “fuel-os-nosdn-kvm_ovs_dpdk_bar-ha-baremetal-daily-master”
- This scenario provides the High Availability feature by deploying 3 controller,2 compute nodes and checking if all the 5 nodes are accessible(IP,up & running).
- Test Scenario is passed if deployment is successful and all 5 nodes have accessibility (IP , up & running).

10.1.4 Known Limitations, Issues and Workarounds

- Test scenario os-nosdn-kvm_ovs_dpdk_bar-ha result is not stable.

10.1.5 References

For more information on the OPNFV Euphrates release, please visit <http://www.opnfv.org/Euphrates>

11.1 Userguide Abstract

In KVM4NFV project, we focus on the KVM hypervisor to enhance it for NFV, by looking at the following areas initially-

- **Minimal Interrupt latency variation for data plane VNFs:**
 - Minimal Timing Variation for Timing correctness of real-time VNFs
 - Minimal packet latency variation for data-plane VNFs
- Inter-VM communication
- Fast live migration

11.2 Userguide Introduction

11.2.1 Overview

The project “NFV Hypervisors-KVM” makes collaborative efforts to enable NFV features for existing hypervisors, which are not necessarily designed or targeted to meet the requirements for the NFVI. The KVM4NFV scenario consists of Continuous Integration builds, deployments and testing combinations of virtual infrastructure components.

11.2.2 KVM4NFV Features

Using this project, the following areas are targeted-

- **Minimal Interrupt latency variation for data plane VNFs:**
 - Minimal Timing Variation for Timing correctness of real-time VNFs
 - Minimal packet latency variation for data-plane VNFs

- Inter-VM communication
- Fast live migration

Some of the above items would require software development and/or specific hardware features, and some need just configurations information for the system (hardware, BIOS, OS, etc.).

We include a requirements gathering stage as a formal part of the project. For each subproject, we will start with an organized requirement stage so that we can determine specific use cases (e.g. what kind of VMs should be live migrated) and requirements (e.g. interrupt latency, jitters, Mpps, migration-time, down-time, etc.) to set out the performance goals.

Potential future projects would include:

- Dynamic scaling (via scale-out) using VM instantiation
- Fast live migration for SR-IOV

The user guide outlines how to work with key components and features in the platform, each feature description section will indicate the scenarios that provide the components and configurations required to use it.

The configuration guide details which scenarios are best for you and how to install and configure them.

11.2.3 General usage guidelines

The user guide for KVM4NFV features and capabilities provide step by step instructions for using features that have been configured according to the installation and configuration instructions.

11.2.4 Scenarios User Guide

The procedure to deploy/test [KVM4NFV scenarios](#) in a nested virtualization or on bare-metal environment is mentioned in the below link. The kvm4nfv user guide can be found at docs/scenarios

```
http://artifacts.opnfv.org/kvmfornfv/docs/index.html#kvmfornfv-scenarios-overview-and-↪description
```

The deployment has been verified for [os-nosdn-kvm-ha](#), [os-nosdn-kvm-noha](#), [os-nosdn-kvm_ovs_dpdk-ha](#), [os-nosdn-kvm_ovs_dpdk-noha](#) and [os-nosdn-kvm_ovs_dpdk_bar-ha](#), [os-nosdn-kvm_ovs_dpdk_bar-noha](#) test scenarios.

For brief view of the above scenarios use:

```
http://artifacts.opnfv.org/kvmfornfv/docs/index.html#scenario-abstract
```

11.3 Using common platform components

This section outlines basic usage principals and methods for some of the commonly deployed components of supported OPNFV scenario's in Euphrates. The subsections provide an outline of how these components are commonly used and how to address them in an OPNFV deployment. The components derive from autonomous upstream communities and where possible this guide will provide direction to the relevant documentation made available by those communities to better help you navigate the OPNFV deployment.

11.4 Using Euphrates Features

The following sections of the user guide provide feature specific usage guidelines and references for KVM4NFV project.

- <project>/docs/userguide/low_latency.userguide.rst
- <project>/docs/userguide/live_migration.userguide.rst
- <project>/docs/userguide/tuning.userguide.rst

11.5 FTrace Debugging Tool

11.5.1 About Ftrace

Ftrace is an internal tracer designed to find what is going on inside the kernel. It can be used for debugging or analyzing latencies and performance related issues that take place outside of user-space. Although ftrace is typically considered the function tracer, it is really a frame work of several assorted tracing utilities.

One of the most common uses of ftrace is the event tracing.

Note: - For KVM4NFV, Ftrace is preferred as it is in-built kernel tool - More stable compared to other debugging tools

11.5.2 Version Features

Release	Features
Colorado	<ul style="list-style-type: none"> • Ftrace Debugging tool is not implemented in Colorado release of KVM4NFV
Danube	<ul style="list-style-type: none"> • Ftrace aids in debugging the KVM4NFV 4.4-linux-kernel level issues • Option to disable if not required
Euphrates	<ul style="list-style-type: none"> • Breaktrace option is implemented. • Implemented post-execute script option to disable the ftrace when it is enabled.

11.5.3 Implementation of Ftrace

Ftrace uses the debugfs file system to hold the control files as well as the files to display output.

When debugfs is configured into the kernel (which selecting any ftrace option will do) the directory /sys/kernel/debug will be created. To mount this directory, you can add to your /etc/fstab file:

debugfs	/sys/kernel/debug	debugfs defaults	0	0
---------	-------------------	------------------	---	---

Or you can mount it at run time with:

```
mount -t debugfs nodev /sys/kernel/debug
```

Some configurations for Ftrace are used for other purposes, like finding latency or analyzing the system. For the purpose of debugging, the kernel configuration parameters that should be enabled are:

```
CONFIG_FUNCTION_TRACER=y
CONFIG_FUNCTION_GRAPH_TRACER=y
CONFIG_STACK_TRACER=y
CONFIG_DYNAMIC_FTRACE=y
```

The above parameters must be enabled in `/boot/config-4.4.0-el7.x86_64` i.e., kernel config file for ftrace to work. If not enabled, change the parameter to `y` and run.,

```
On CentOS
grub2-mkconfig -o /boot/grub2/grub.cfg
sudo reboot
```

Re-check the parameters after reboot before running ftrace.

11.5.4 Files in Ftrace:

The below is a list of few major files in Ftrace.

`current_tracer:`

This is used to set or display the current tracer that is configured.

`available_tracers:`

This holds the different types of tracers that have been compiled into the kernel. The tracers listed here can be configured by echoing their name into `current_tracer`.

`tracing_on:`

This sets or displays whether writing to the tracing buffer is enabled. Echo 0 into this file to disable the tracer or 1 to enable it.

`trace:`

This file holds the output of the trace in a human readable format.

`tracing_cpumask:`

This is a mask that lets the user only trace on specified CPUs. The format is a hex string representing the CPUs.

`events:`

It holds event tracepoints (also known as static tracepoints) that have been compiled into the kernel. It shows what event tracepoints exist and how they are grouped by system.

11.5.5 Available Tracers

Here is the list of current tracers that may be configured based on usage.

- `function`
- `function_graph`
- `irqsoff`

- preemptoff
- preemptirqsoff
- wakeup
- wakeup_rt

Brief about a few:

function:

Function call tracer to trace all kernel functions.

function_graph:

Similar to the function tracer except that the function tracer probes the functions on their entry whereas the function graph tracer traces on both entry and exit of the functions.

nop:

This is the “trace nothing” tracer. To remove tracers from tracing simply echo “nop” into `current_tracer`.

Examples:

```
To list available tracers:
[tracing]# cat available_tracers
function_graph function wakeup wakeup_rt preemptoff irqsoff preemptirqsoff nop

Usage:
[tracing]# echo function > current_tracer
[tracing]# cat current_tracer
function

To view output:
[tracing]# cat trace | head -10

To Stop tracing:
[tracing]# echo 0 > tracing_on

To Start/restart tracing:
[tracing]# echo 1 > tracing_on;
```

11.5.6 Ftrace in KVM4NFV

Ftrace is part of KVM4NFV D-Release. KVM4NFV built 4.4-linux-Kernel will be tested by executing `cyclictest` and analyzing the results/latency values (max, min, avg) generated. Ftrace (or) function tracer is a stable kernel inbuilt debugging tool which tests real time kernel and outputs a log as part of the code. These output logs are useful in following ways.

- Kernel Debugging.
- Helps in Kernel code optimization and
- Can be used to better understand the kernel level code flow

Ftrace logs for KVM4NFV can be found [here](#):

11.5.7 Ftrace Usage in KVM4NFV Kernel Debugging:

Kvm4nfv has two scripts in /ci/envs to provide ftrace tool:

- enable_trace.sh
- disable_trace.sh

```
Found at.,  
$ cd kvmfornfv/ci/envs
```

11.5.8 Enabling Ftrace in KVM4NFV

The enable_trace.sh script is triggered by changing ftrace_enable value in test_kvmfornfv.sh script to 1 (which is zero by default). Change as below to enable Ftrace.

```
ftrace_enable=1
```

Note:

- Ftrace is enabled before

11.5.9 Details of enable_trace script

- CPU coremask is calculated using getcpumask()
- **All the required events are enabled by**, echoing “1” to \$TRACEDIR/events/event_name/enable file

Example,

```
$TRACEDIR = /sys/kernel/debug/tracing/  
sudo bash -c "echo 1 > $TRACEDIR/events/irq/enable"  
sudo bash -c "echo 1 > $TRACEDIR/events/task/enable"  
sudo bash -c "echo 1 > $TRACEDIR/events/syscalls/enable"
```

The set_event file contains all the enabled events list

- Function tracer is selected. May be changed to other available tracers based on requirement

```
sudo bash -c "echo function > $TRACEDIR/current_tracer"
```

• When tracing is turned ON by setting tracing_on=1, the trace file keeps getting append with the traced data until tracing_on=0 and then ftrace_buffer gets cleared.

```
To Stop/Pause,  
echo 0 >tracing_on;  
  
To Start/Restart,  
echo 1 >tracing_on;
```

- Once tracing is disabled, disable_trace.sh script is triggered.

11.5.10 BREAKTRACE

- Send break trace command when latency > USEC. This is a debugging option to control the latency

tracer in the realtime preemption patch. It is useful to track down unexpected large latencies on a system. This option does only work with following kernel config options.

```
For kernel < 2.6.24: * CONFIG_PREEMPT_RT=y * CONFIG_WAKEUP_TIMING=y *
CONFIG_LATENCY_TRACE=y * CONFIG_CRITICAL_PREEMPT_TIMING=y * CON-
FIG_CRITICAL_IRQSOFF_TIMING=y
```

```
For kernel >= 2.6.24: * CONFIG_PREEMPT_RT=y * CONFIG_FTRACE * CONFIG_IRQSOFF_TRACER=y *
CONFIG_PREEMPT_TRACER=y * CONFIG_SCHED_TRACER=y * CONFIG_WAKEUP_LATENCY_HIST
```

- Kernel configuration options enabled. The USEC parameter to the -b option defines a maximum

latency value, which is compared against the actual latencies of the test. Once the measured latency is higher than the given maximum, the kernel tracer and cyclicttest is stopped. The trace can be read from /proc/latency_trace. Please be aware that the tracer adds significant overhead to the kernel, so the latencies will be much higher than on a kernel with latency tracing disabled.

- Breaktrace option will enable the trace by default, suppress the tracing by using -notrace option.

11.5.11 Post-execute scripts

post-execute script to yardstick node context teardown is added to disable the ftrace soon after the completion of cyclicttest execution throughyardstick. This option is implemented to collect only required ftrace logs for effective debugging if needed.

11.5.12 Details of disable_trace Script

In disable trace script the following are done:

- The trace file is copied and moved to /tmp folder based on timestamp
- The current tracer file is set to nop
- The set_event file is cleared i.e., all the enabled events are disabled
- Kernel Ftrace is disabled/unmounted

11.5.13 Publishing Ftrace logs:

The generated trace log is pushed to [artifacts](#) by kvmfornfv-upload-artifact.sh script available in releng which will be triggered as a part of kvm4nfv daily job. The [trigger](#) in the script is.,

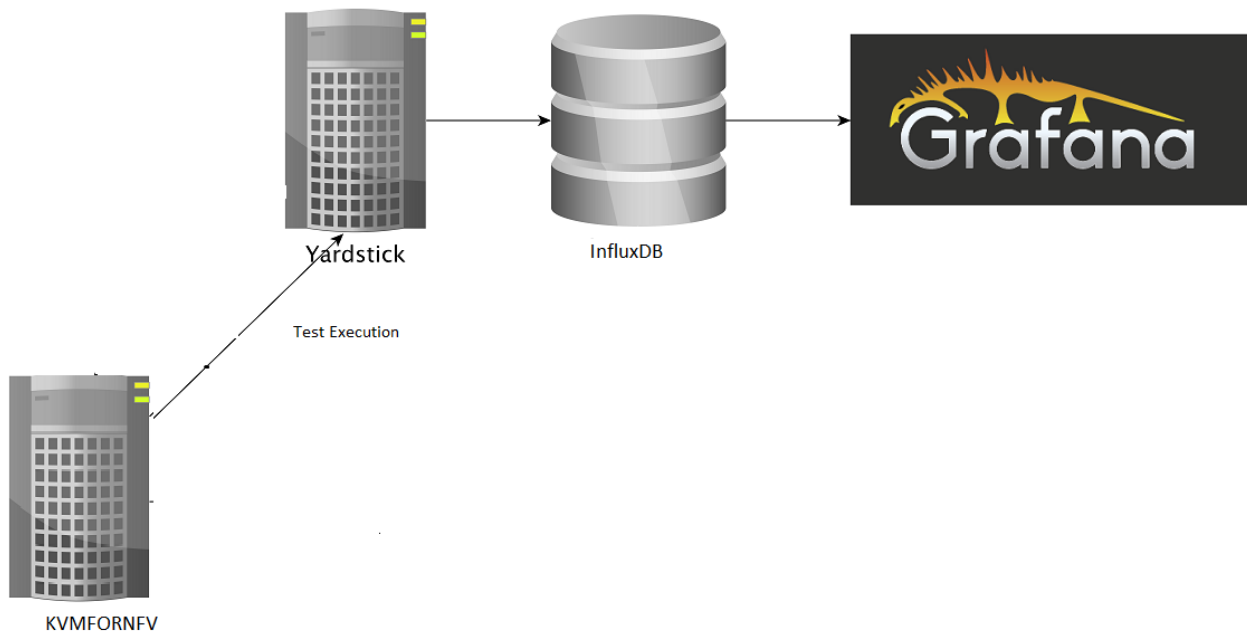
```
echo "Uploading artifacts for future debugging needs...."
gsutil cp -r $WORKSPACE/build_output/log-*.tar.gz $GS_LOG_LOCATION > $WORKSPACE/
↪gsutil.log 2>&1
```

11.6 KVM4NFV Dashboard Guide

11.6.1 Dashboard for KVM4NFV Daily Test Results

11.6.2 Abstract

This chapter explains the procedure to configure the InfluxDB and Grafana on Node1 or Node2 depending on the testtype to publish KVM4NFV test results. The cyclicttest cases are executed and results are published on Yardstick Dashboard(Grafana). InfluxDB is the database which will store the cyclicttest results and Grafana is a visualisation suite to view the maximum,minimum and average values of the time series data of cyclicttest results.The framework is shown in below image.



11.6.3 Version Features

Release	Features
Colorado	<ul style="list-style-type: none"> • Data published in Json file format • No database support to store the test's latency values of cyclicttest • For each run, the previous run's output file is replaced with a new file with currents latency values.
Danube	<ul style="list-style-type: none"> • Test results are stored in Influxdb • Graphical representation of the latency values using Grafana suite. (Dashboard) • Supports graphical view for multiple testcases and test-types (Stress/Idle)

11.6.4 Installation Steps:

To configure Yardstick, InfluxDB and Grafana for KVM4NFV project following sequence of steps are followed:

Note:

All the below steps are done as per the script, which is a part of CICD integration of kvmfornfv.

```
For Yardstick:
git clone https://gerrit.opnfv.org/gerrit/yardstick

For InfluxDB:
docker pull tutum/influxdb
docker run -d --name influxdb -p 8083:8083 -p 8086:8086 --expose 8090 --expose 8099_
↳tutum/influxdb
docker exec -it influxdb bash
$influx
>CREATE USER root WITH PASSWORD 'root' WITH ALL PRIVILEGES
>CREATE DATABASE yardstick;
>use yardstick;
>show MEASUREMENTS;

For Grafana:
docker pull grafana/grafana
docker run -d --name grafana -p 3000:3000 grafana/grafana
```

The Yardstick document for Grafana and InfluxDB configuration can be found [here](#).

11.6.5 Configuring the Dispatcher Type:

Need to configure the dispatcher type in /etc/yardstick/yardstick.conf depending on the dispatcher methods which are used to store the cyclicttest results. A sample yardstick.conf can be found at /yardstick/etc/yardstick.conf.sample, which can be copied to /etc/yardstick.

```
mkdir -p /etc/yardstick/
cp /yardstick/etc/yardstick.conf.sample /etc/yardstick/yardstick.conf
```

Dispatcher Modules:

Three type of dispatcher methods are available to store the cyclicttest results.

- File
- InfluxDB
- HTTP

1. File: Default Dispatcher module is file. If the dispatcher module is configured as a file, then the test results are stored in a temporary file yardstick.out(default path: /tmp/yardstick.out). Dispatcher module of “Verify Job” is “Default”. So,the results are stored in Yardstick.out file for verify job. Storing all the verify jobs in InfluxDB database causes redundancy of latency values. Hence, a File output format is preferred.

```
[DEFAULT]
debug = False
dispatcher = file

[dispatcher_file]
file_path = /tmp/yardstick.out
```

(continues on next page)

(continued from previous page)

```
max_bytes = 0
backup_count = 0
```

2. Influxdb: If the dispatcher module is configured as influxdb, then the test results are stored in Influxdb. Users can check test results stored in the Influxdb(Database) on Grafana which is used to visualize the time series data.

To configure the influxdb, the following content in /etc/yardstick/yardstick.conf need to updated

```
[DEFAULT]
debug = False
dispatcher = influxdb

[dispatcher_influxdb]
timeout = 5
target = http://127.0.0.1:8086 ##Mention the IP where influxdb is running
db_name = yardstick
username = root
password = root
```

Dispatcher module of “Daily Job” is Influxdb. So, the results are stored in influxdb and then published to Dashboard.

3. HTTP: If the dispatcher module is configured as http, users can check test result on OPNFV testing dashboard which uses MongoDB as backend.

```
[DEFAULT]
debug = False
dispatcher = http

[dispatcher_http]
timeout = 5
target = http://127.0.0.1:8000/results
```

11.6.5.1 Detailing the dispatcher module in verify and daily Jobs:

KVM4NFV updates the dispatcher module in the yardstick configuration (/etc/yardstick/yardstick.conf) file depending on the Job type(Verify/Daily). Once the test is completed, results are published to the respective dispatcher modules.

Dispatcher module is configured for each Job type as mentioned below.

1. Verify Job : Default “DISPATCHER_TYPE” i.e. file(/tmp/yardstick.out) is used. User can also see the test results on Jenkins console log.

```
*"max": "00030", "avg": "00006", "min": "00006"*
```

2. Daily Job : Opnfv Influxdb url is configured as dispatcher module.

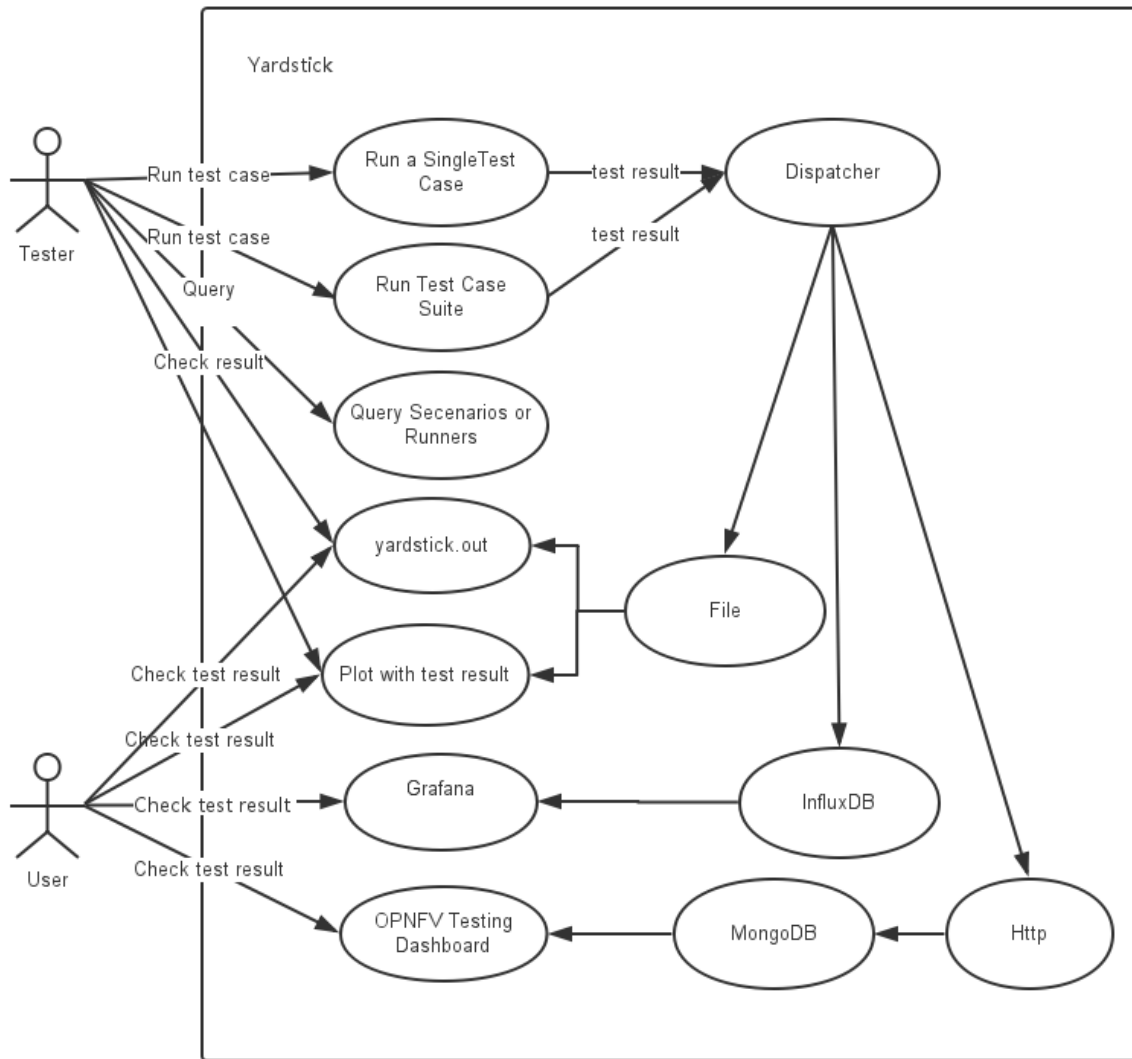
```
DISPATCHER_TYPE=influxdb
DISPATCHER_INFLUXDB_TARGET="http://104.197.68.199:8086"
```

Influxdb only supports line protocol, and the json protocol is deprecated.

For example, the raw_result of cyclicttest in json format is:

```
"benchmark": {
  "timestamp": 1478234859.065317,
  "errors": "",
```

(continues on next page)



(continued from previous page)

```

    "data": {
      "max": "00012",
      "avg": "00008",
      "min": "00007"
    },
    "sequence": 1
  },
  "runner_id": 23
}

```

With the help of “[influxdb_line_protocol](#)”, the json is transformed as a line string:

```

'kvmfornfv_cyclictest_idle_idle,deploy_scenario=unknown,host=kvm.LF,
installer=unknown,pod_name=unknown,runner_id=23,scenarios=Cyclictest,
task_id=e7be7516-9eae-406e-84b6-e931866fa793,version=unknown
avg="00008",max="00012",min="00007" 1478234859065316864'

```

Influxdb api which is already implemented in [Influxdb](#) will post the data in line format into the database.

Displaying Results on Grafana dashboard:

- Once the test results are stored in Influxdb, dashboard configuration file(Json) which used to display the cyclictest results on Grafana need to be created by following the [Grafana-procedure](#) and then pushed into [yardstick-repo](#)
- Grafana can be accessed at [Login](#) using credentials opnfv/opnfv and used for visualizing the collected test data as shown in [Visual](#)

11.6.6 Understanding Kvm4nfv Grafana Dashboard

The Kvm4nfv dashboard found at <http://testresults.opnfv.org/> currently supports graphical view of cyclictest. For viewing Kvm4nfv dashboard use,

```
http://testresults.opnfv.org/grafana/dashboard/db/kvmfornfv-cyclictest
```

The login details are:

```

Username: opnfv
Password: opnfv

```

The JSON of the kvmfonfv-cyclictest dashboard can be found at.,

```

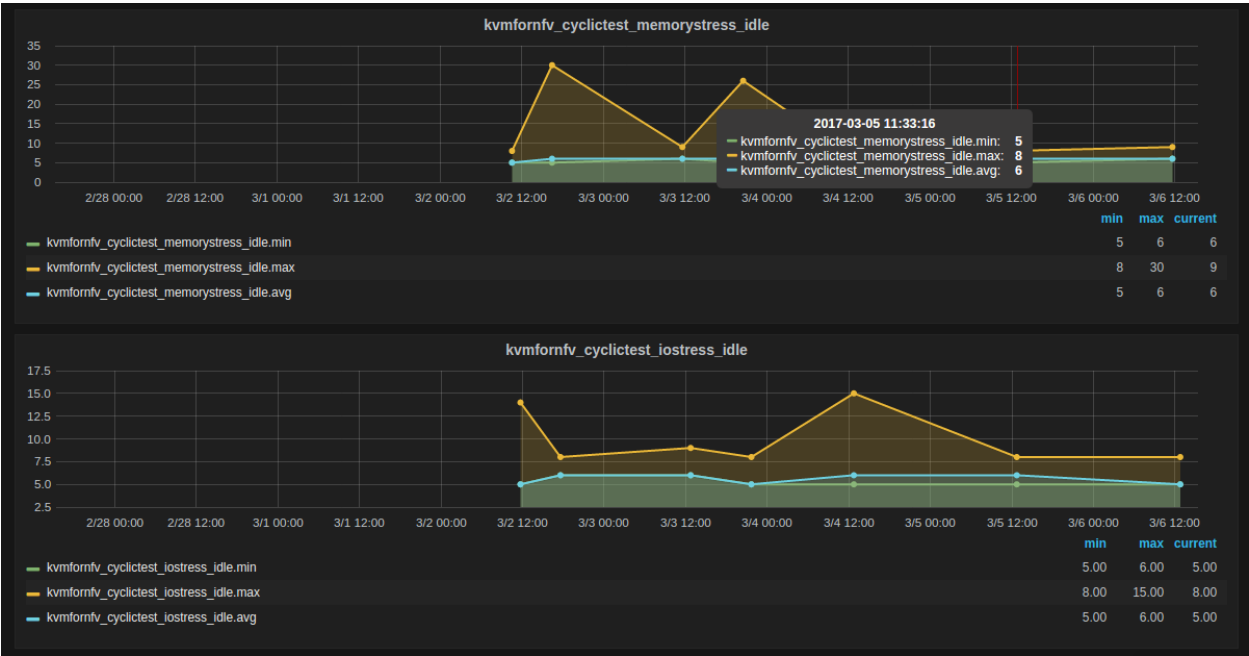
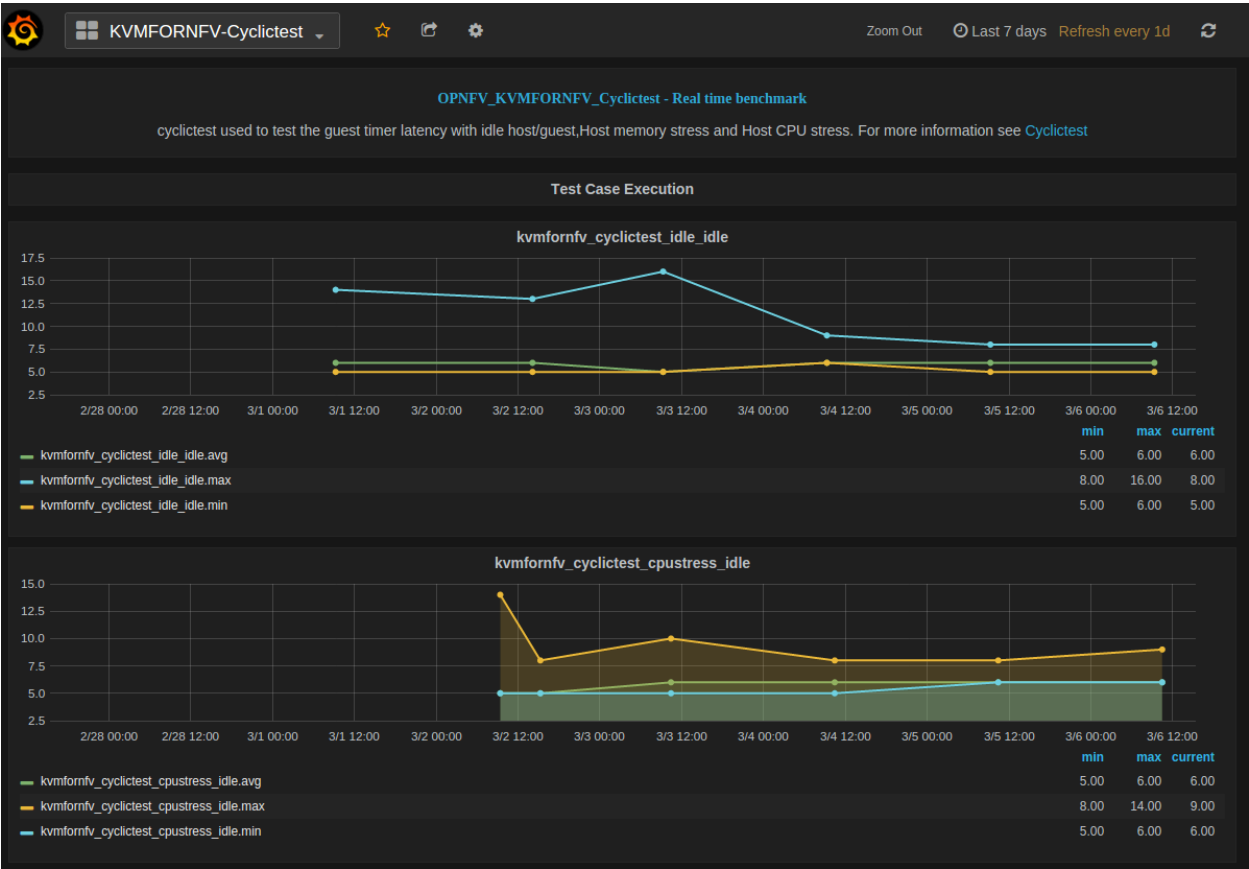
$ git clone https://gerrit.opnfv.org/gerrit/yardstick.git
$ cd yardstick/dashboard
$ cat KVMFORNFV-Cyclictest

```

The Dashboard has four tables, each representing a specific test-type of cyclictest case,

- Kvmfornfv_Cyclictest_Idle-Idle
- Kvmfornfv_Cyclictest_CPUstress-Idle
- Kvmfornfv_Cyclictest_Memorstress-Idle
- Kvmfornfv_Cyclictest_IOstress-Idle

Note:

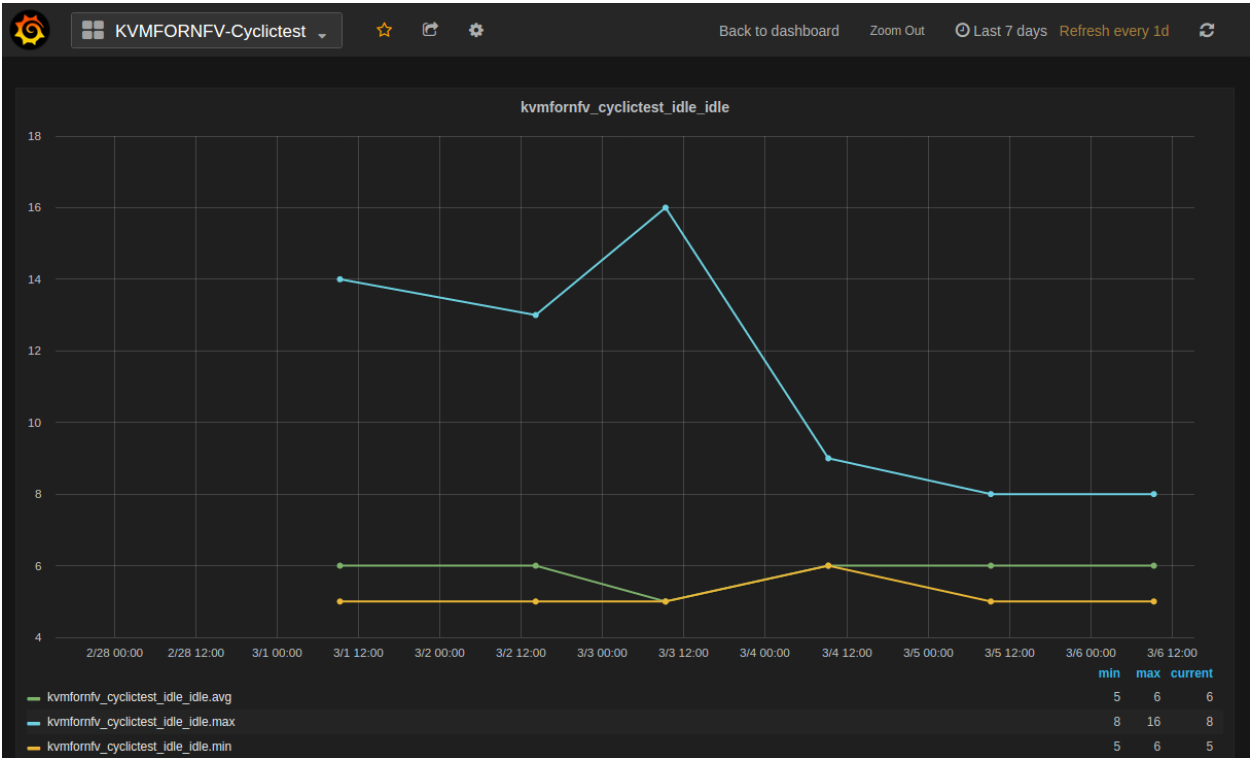


- For all graphs, X-axis is marked with time stamps, Y-axis with value in microsecond units.

A brief about what each graph of the dashboard represents:

11.6.6.1 1. Idle-Idle Graph

Idle-Idle graph displays the Average, Maximum and Minimum latency values obtained by running Idle_Idle test-type of the cyclicttest. Idle_Idle implies that no stress is applied on the Host or the Guest.



11.6.6.2 2. CPU_Stress-Idle Graph

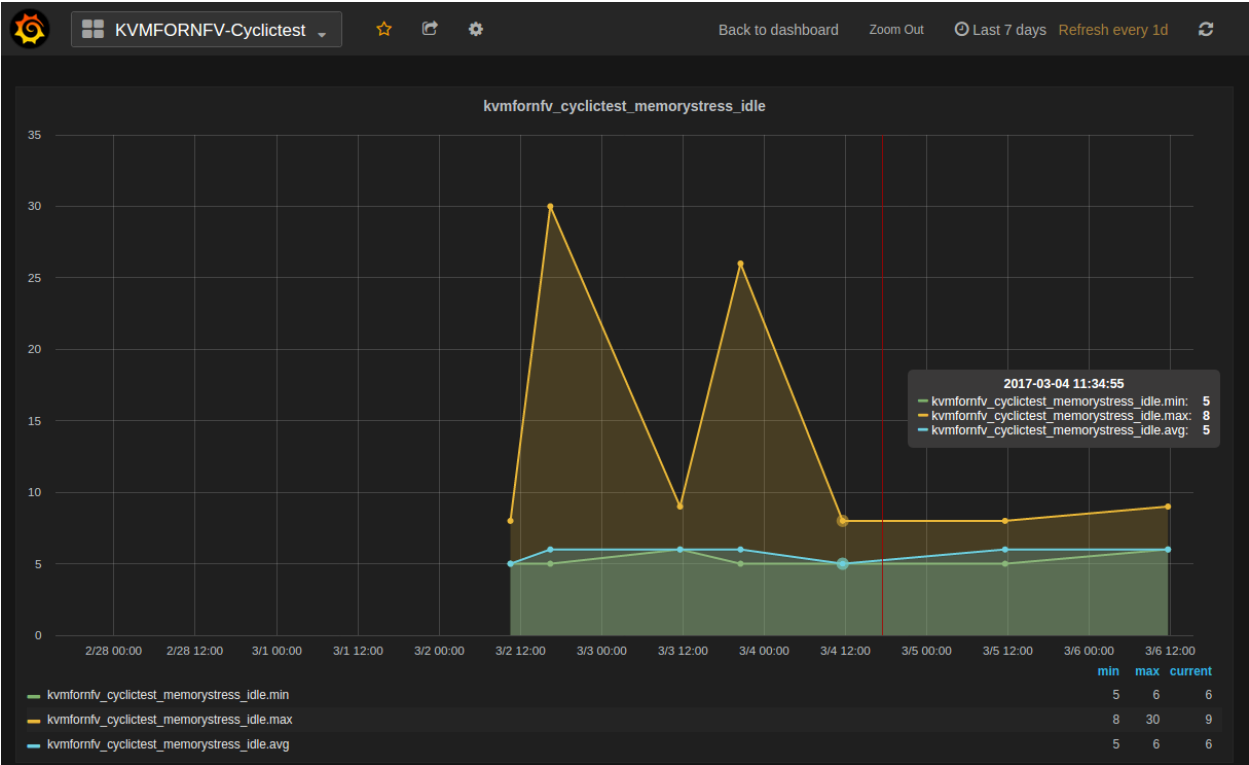
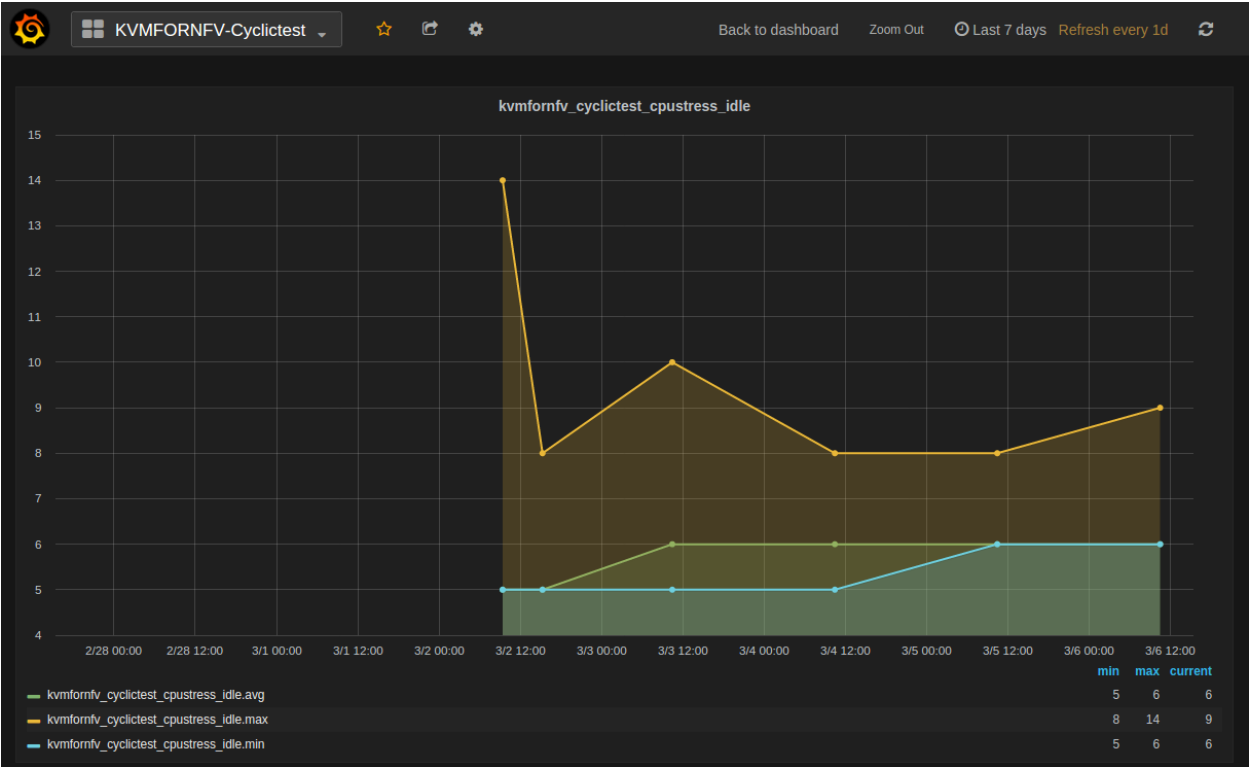
Cpu_Stress-Idle graph displays the Average, Maximum and Minimum latency values obtained by running Cpu-stress_Idle test-type of the cyclicttest. Cpu-stress_Idle implies that CPU stress is applied on the Host and no stress on the Guest.

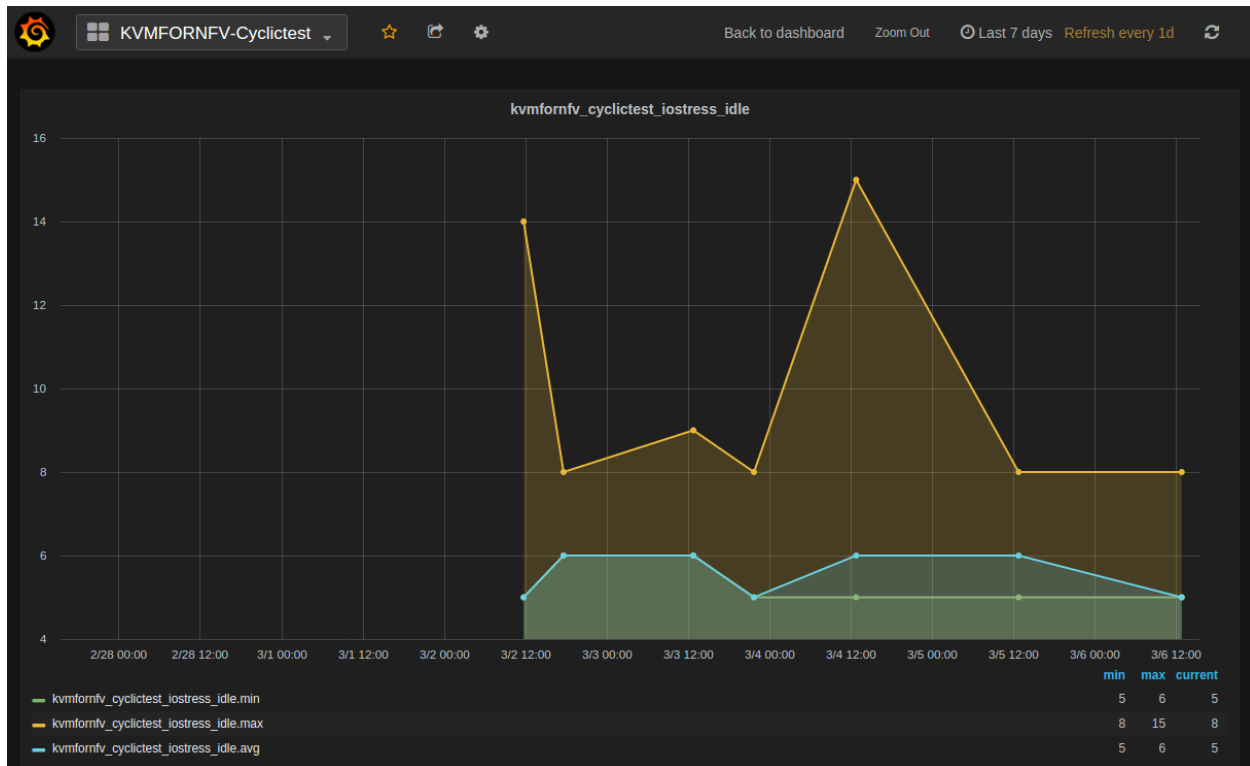
11.6.6.3 3. Memory_Stress-Idle Graph

Memory_Stress-Idle graph displays the Average, Maximum and Minimum latency values obtained by running Memory-stress_Idle test-type of the Cyclicttest. Memory-stress_Idle implies that Memory stress is applied on the Host and no stress on the Guest.

11.6.6.4 4. IO_Stress-Idle Graph

IO_Stress-Idle graph displays the Average, Maximum and Minimum latency values obtained by running IO-stress_Idle test-type of the Cyclicttest. IO-stress_Idle implies that IO stress is applied on the Host and no stress on the Guest.





11.6.6.5 Packet Forwarding Results

11.6.7 Understanding Kvm4nfv Grafana Dashboard

The Kvm4nfv dashboard found at <http://testresults.opnfv.org/grafana/> currently supports graphical view of packet forwarding as well. For viewing Kvm4nfv packet forwarding dashboard use,

```
http://testresults.opnfv.org/grafana/dashboard/db/kvmfornfv-packet-forwarding
```

The login details are:

```
Username: opnfv
Password: opnfv
```

The JSON of the KVMFORNFV-Packet-Forwarding dashboard can be found at.,

```
$ git clone https://gerrit.opnfv.org/gerrit/yardstick.git
$ cd yardstick/dashboard
$ cat KVMFORNFV-Packet-Forwarding
```

The Dashboard has five tables for each specific test of packet forwarding, one for each frame size.

- KVM4NFV-PHY2PHY-TPUT-OVS_WITH_DPDK_AND_VHOST_USER
- KVM4NFV-PVP-TPUT-OVS_WITH_DPDK_AND_VHOST_USER
- KVM4NFV-PVP-TPUT-SRIOV
- KVM4NFV-PVVP-TPUT-OVS_WITH_DPDK_AND_VHOST_USER
- KVM4NFV-PVVP-TPUT-OVS_WITH_DPDK_AND_VHOST_USER

Note:

- For all graphs, X-axis is marked with time stamps, Y-axis with value in microsecond units.

11.6.8 Future Scope

The future work will include adding new tables to packet forwarding Grafana dashboard to publish the results of new packet forwarding test cases to be added if any.

11.7 Low Latency Environment

Achieving low latency with the KVM4NFV project requires setting up a special test environment. This environment includes the BIOS settings, kernel configuration, kernel parameters and the run-time environment.

11.7.1 Hardware Environment Description

BIOS setup plays an important role in achieving real-time latency. A collection of relevant settings, used on the platform where the baseline performance data was collected, is detailed below:

11.7.1.1 CPU Features

Some special CPU features like TSC-deadline timer, invariant TSC and Process posted interrupts, etc, are helpful for latency reduction.

11.7.1.2 CPU Topology

NUMA topology is also important for latency reduction.

11.7.1.3 BIOS Setup

Careful BIOS setup is important in achieving real time latency. Different platforms have different BIOS setups, below are the important BIOS settings on the platform used to collect the baseline performance data.

11.7.2 Software Environment Setup

Both the host and the guest environment need to be configured properly to reduce latency variations. Below are some suggested kernel configurations. The ci/envs/ directory gives detailed implementation on how to setup the environment.

11.7.2.1 Kernel Parameter

Please check the default kernel configuration in the source code at: kernel/arch/x86/configs/opnfv.config.

Below is host kernel boot line example:

```
isolcpus=11-15,31-35 nohz_full=11-15,31-35 rcu_nocbs=11-15,31-35
iommu=pt intel_iommu=on default_hugepagesz=1G hugepagesz=1G mce=off idle=poll
intel_pstate=disable processor.max_cstate=1 pcie_asmp=off tsc=reliable
```

Below is guest kernel boot line example

```
isolcpus=1 nohz_full=1 rcu_nocbs=1 mce=off idle=poll default_hugepagesz=1G
hugepagesz=1G
```

Please refer to *tuning.userguide* for more explanation.

11.7.2.2 Run-time Environment Setup

Not only are special kernel parameters needed but a special run-time environment is also required. Please refer to *tuning.userguide* for more explanation.

11.7.3 Test cases to measure Latency

The performance of the kvm4nfv is assessed by the latency values. Cyclicttest and Packet forwarding Test cases result in real time latency values of average, minimum and maximum.

- Cyclicttest
- Packet Forwarding test

11.7.4 1. Cyclicttest case

Cyclicttest results are the most frequently cited real-time Linux metric. The core concept of Cyclicttest is very simple. In KVM4NFV cyclicttest is implemented on the Guest-VM with 4.4-Kernel RPM installed. It generated Max,Min and Avg values which help in assesing the kernel used. Cyclicttest in currently divided into the following test types,

- Idle-Idle
- CPU_stress-Idle
- Memory_stress-Idle
- IO_stress-Idle

Future scope of work may include the below test-types,

- CPU_stress-CPU_stress
- Memory_stress-Memory_stress
- IO_stress-IO_stress

11.7.4.1 Understanding the naming convention

```
[Host-Type ] - [Guest-Type]
```

- **Host-Type** : Mentions the type of stress applied on the kernel of the Host
- **Guest-Type** : Mentions the type of stress applied on the kernel of the Guest

Example.,

```
Idle - CPU_stress
```

The above name signifies that,

- No Stress is applied on the Host kernel

- CPU Stress is applied on the Guest kernel

Note:

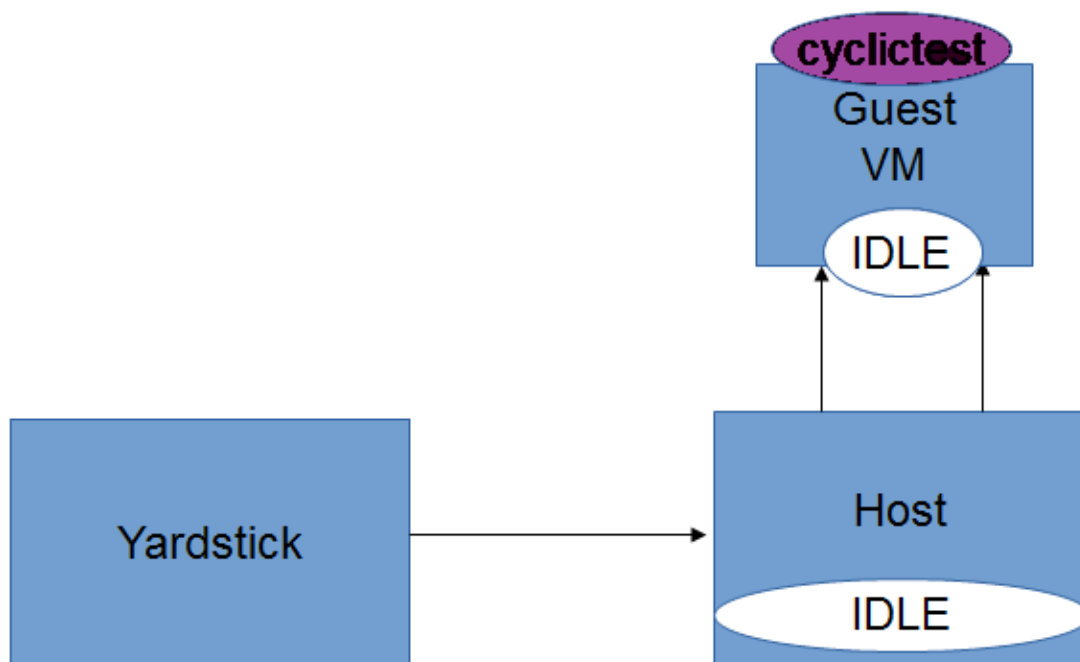
- Stress is applied using the stress which is installed as part of the deployment. Stress can be applied on CPU, Memory and Input-Output (Read/Write) operations using the stress tool.

11.7.4.2 Version Features

Test Name	Colorado	Danube	Euphrates
• Idle - Idle	Y	Y	Y
• Cpustress - Idle		Y	Y
• Memorstress - Idle		Y	Y
• IOstress - Idle		Y	Y

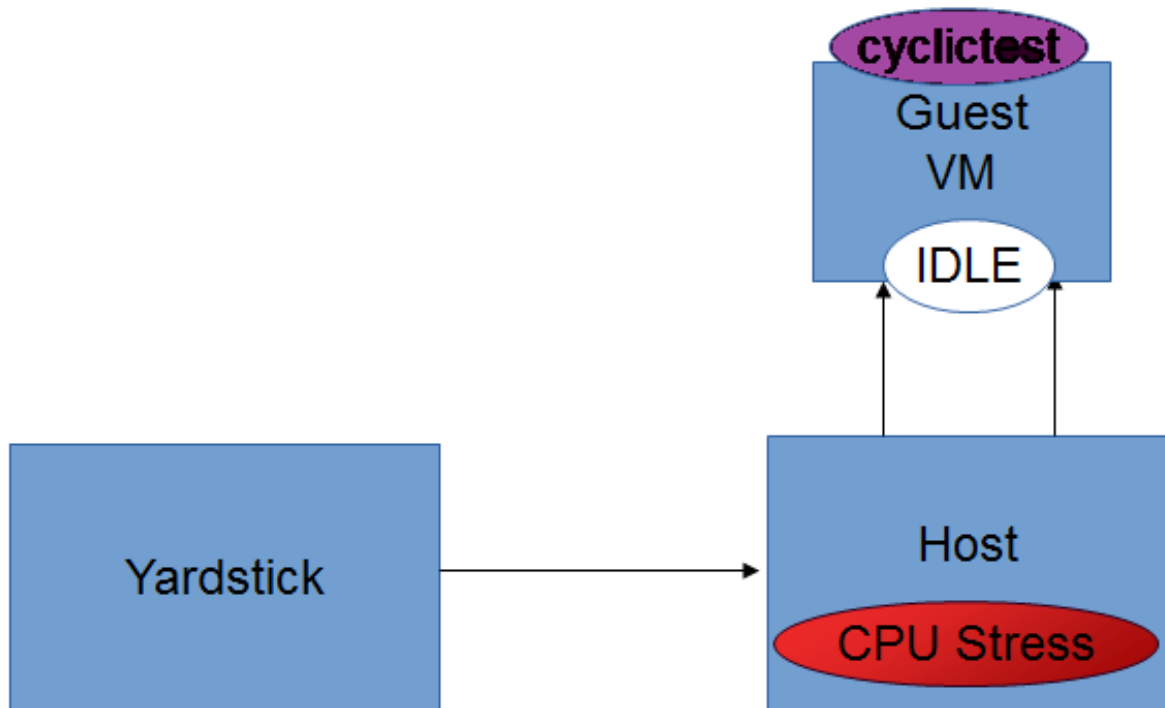
11.7.4.3 Idle-Idle test-type

Cyclictest is run on the Guest VM when Host, Guest are not under any kind of stress. This is the basic cyclictest of the KVM4NFV project. Outputs Avg, Min and Max latency values.



11.7.4.4 CPU_Stress-Idle test-type

Here, the host is under CPU stress, where multiple times `sqrt()` function is called on kernel which results increased CPU load. The `cyclicttest` will run on the guest, where the guest is under no stress. Outputs Avg, Min and Max latency values.



11.7.4.5 Memory_Stress-Idle test-type

In this type, the host is under memory stress where continuous memory operations are implemented to increase the Memory stress (Buffer stress). The `cyclicttest` will run on the guest, where the guest is under no stress. It outputs Avg, Min and Max latency values.

11.7.4.6 IO_Stress-Idle test-type

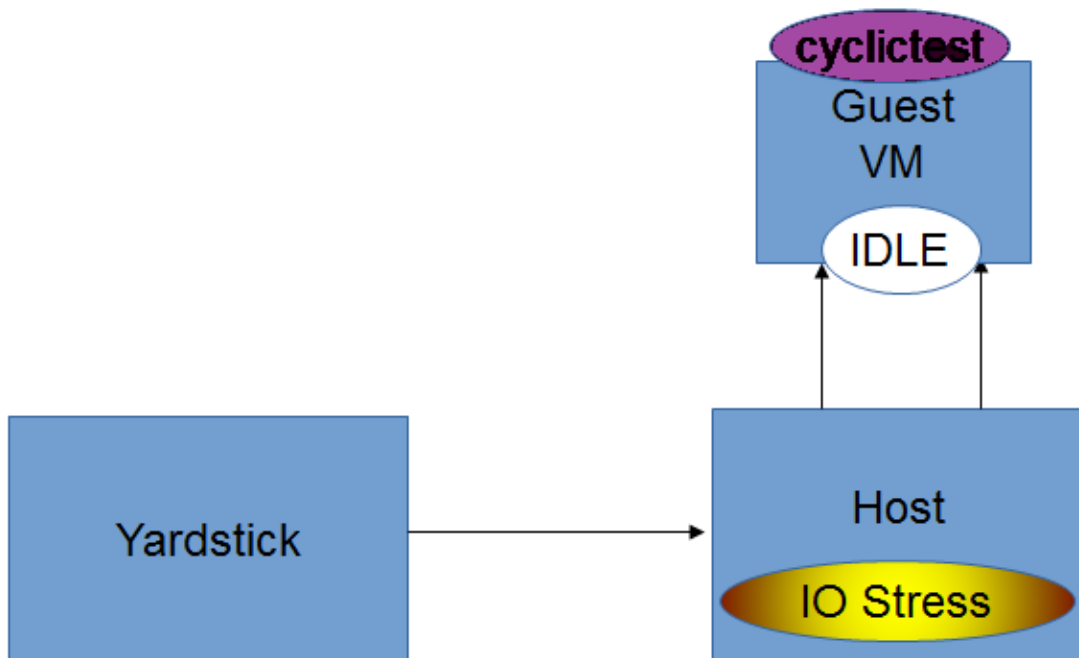
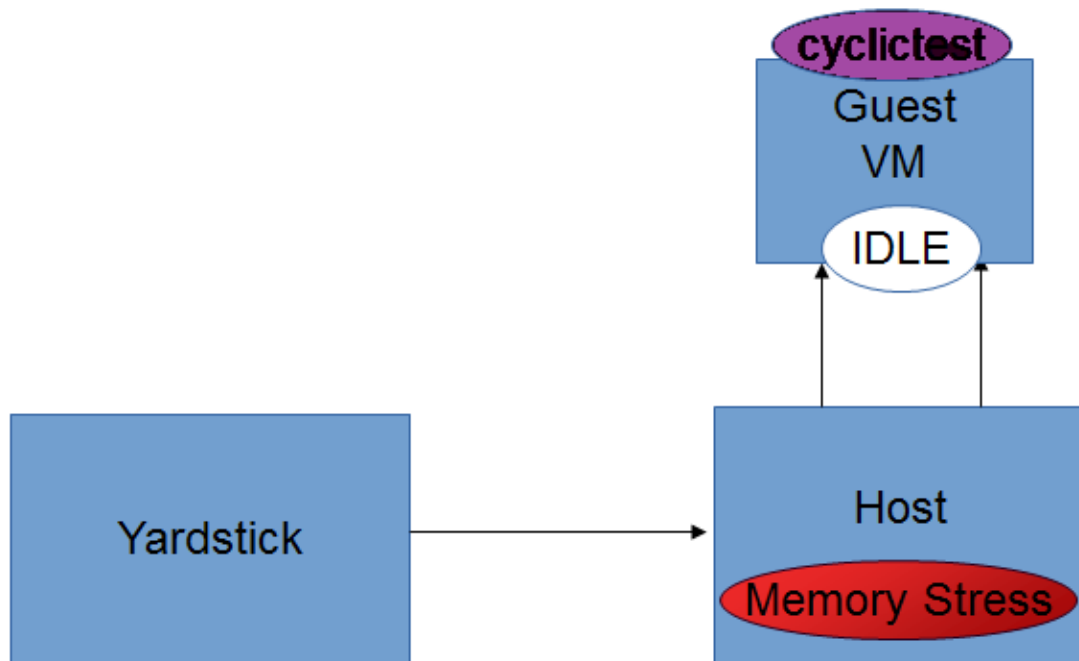
The host is under constant Input/Output stress .i.e., multiple read-write operations are invoked to increase stress. `Cyclicttest` will run on the guest VM that is launched on the same host, where the guest is under no stress. It outputs Avg, Min and Max latency values.

11.7.4.7 CPU_Stress-CPU_Stress test-type

Not implemented for Euphrates release.

11.7.4.8 Memory_Stress-Memory_Stress test-type

Not implemented for Euphrates release.



11.7.4.9 IO_Stress-IO_Stress test type

Not implemented for Euphrates release.

11.7.5 2. Packet Forwarding Test cases

Packet forwarding is an other test case of Kvm4nfv. It measures the time taken by a packet to return to source after reaching its destination. This test case uses automated test-framework provided by OPNFV VSWITCHPERF project and a traffic generator (IXIA is used for kvm4nfv). Only latency results generating test cases are triggered as a part of kvm4nfv daily job.

Latency test measures the time required for a frame to travel from the originating device through the network to the destination device. Please note that RFC2544 Latency measurement will be superseded with a measurement of average latency over all successfully transferred packets or frames.

Packet forwarding test cases currently supports the following test types:

- Packet forwarding to Host
- Packet forwarding to Guest
- Packet forwarding to Guest using SRIOV

The testing approach adopted is black box testing, meaning the test inputs can be generated and the outputs captured and completely evaluated from the outside of the System Under Test(SUT).

11.7.5.1 Packet forwarding to Host

This is also known as Physical port → vSwitch → physical port deployment. This test measures the time taken by the packet/frame generated by traffic generator(phy) to travel through the network to the destination device(phy). This test results min,avg and max latency values. This value signifies the performance of the installed kernel.

Packet flow,

11.7.5.2 Packet forwarding to Guest

This is also known as Physical port → vSwitch → VNF → vSwitch → physical port deployment.

This test measures the time taken by the packet/frame generated by traffic generator(phy) to travel through the network involving a guest to the destination device(phy). This test results min,avg and max latency values. This value signifies the performance of the installed kernel.

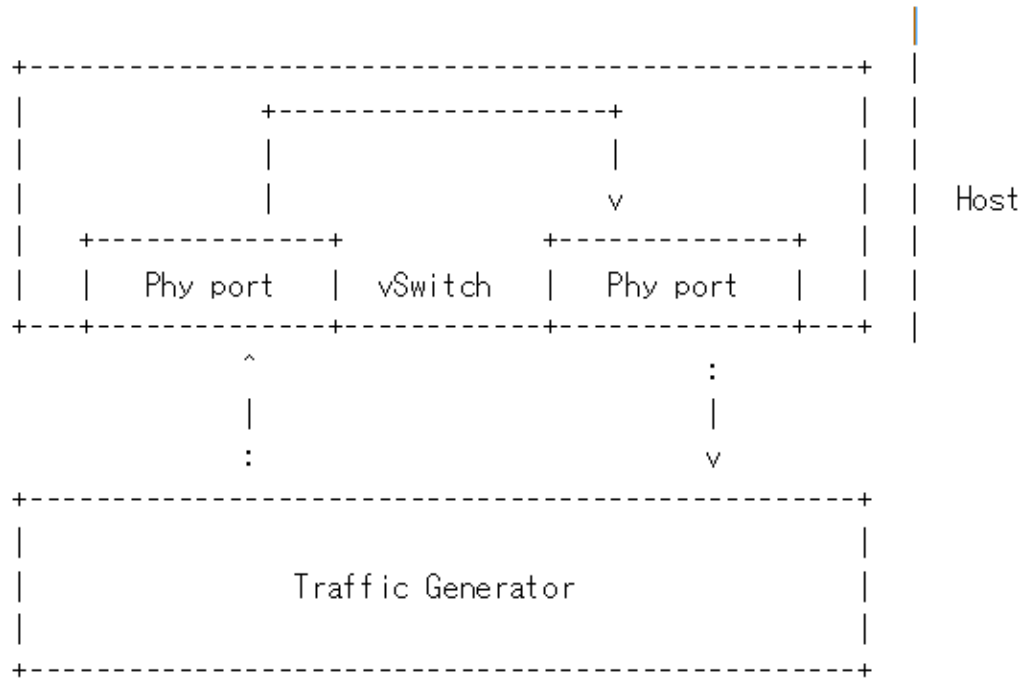
Packet flow,

11.7.5.3 Packet forwarding to Guest using SRIOV

This test is used to verify the VNF and measure the base performance (maximum forwarding rate in fps and latency) that can be achieved by the VNF without a vSwitch. The performance metrics collected by this test will serve as a key comparison point for NIC passthrough technologies and vSwitches. VNF in this context refers to the hypervisor and the VM.

Note: The Vsperf running on the host is still required.

Packet flow,



11.8 Fast Live Migration

The NFV project requires fast live migration. The specific requirement is total live migration time < 2Sec, while keeping the VM down time < 10ms when running DPDK L2 forwarding workload.

We measured the baseline data of migrating an idle 8GiB guest running a DPDK L2 forwarding work load and observed that the total live migration time was 2271ms while the VM downtime was 26ms. Both of these two indicators failed to satisfy the requirements.

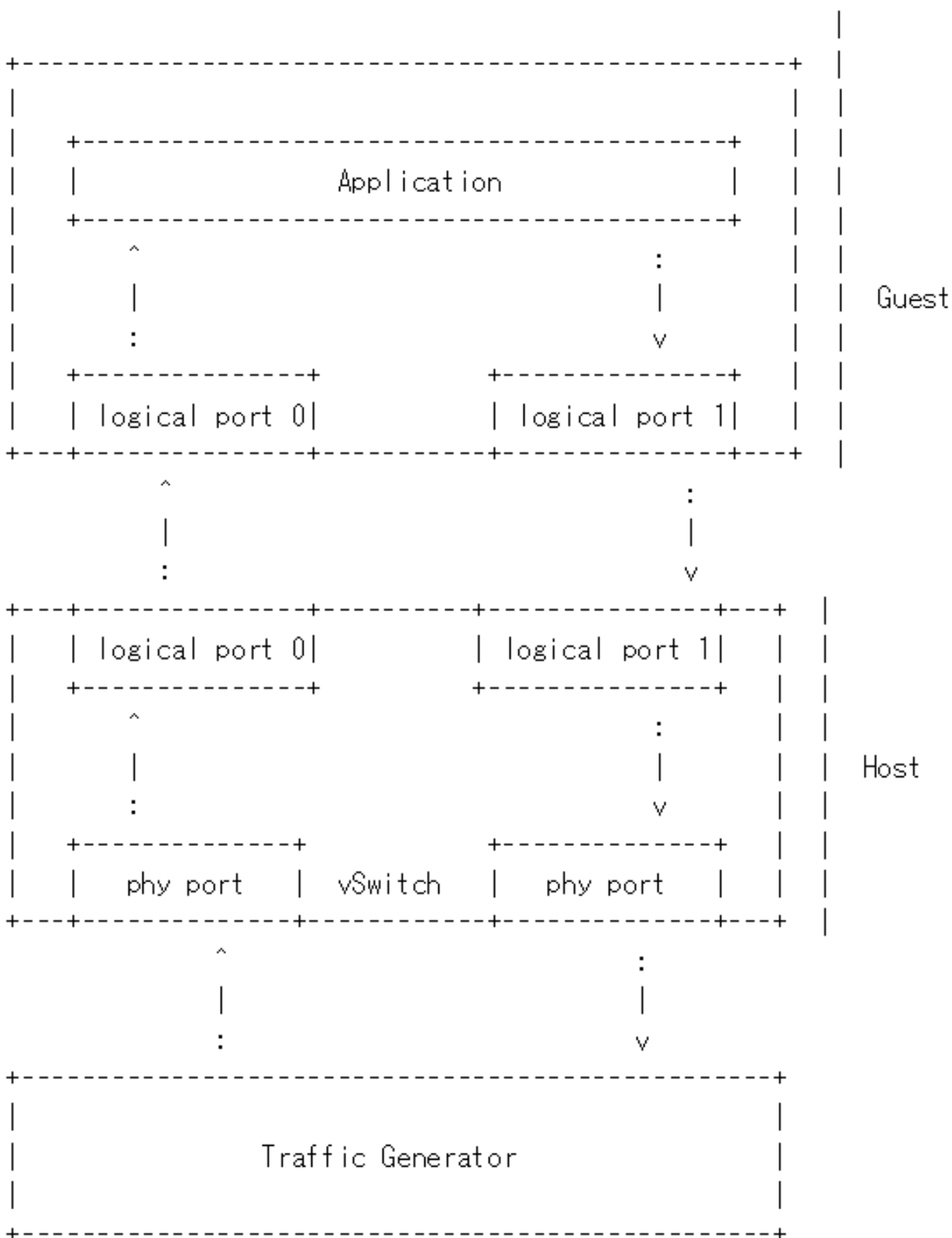
11.8.1 Current Challenges

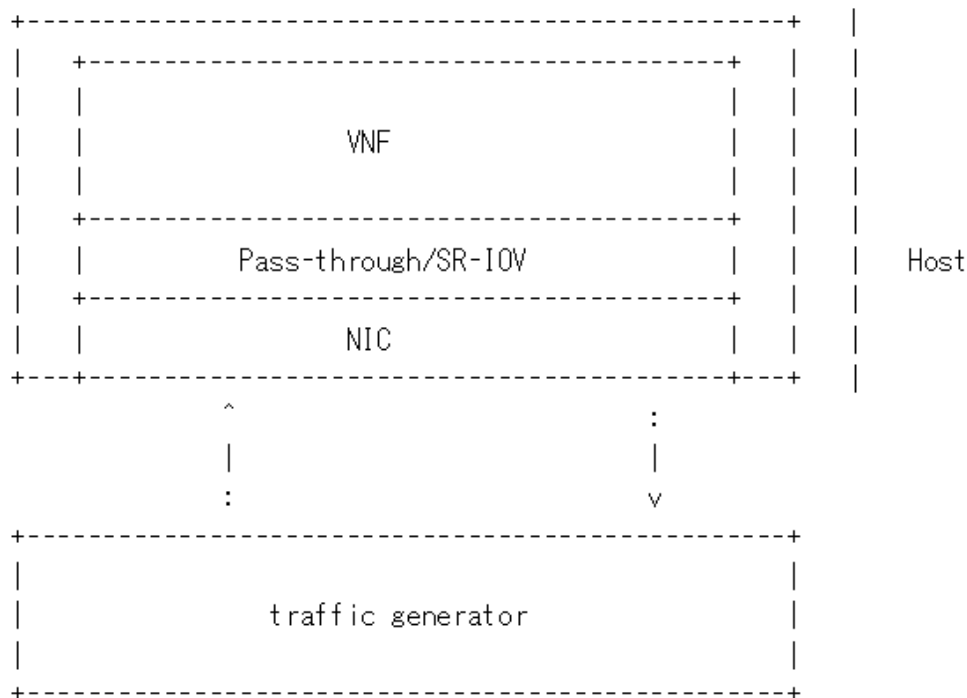
The following 4 features have been developed over the years to make the live migration process faster.

- **XBZRLE:** Helps to reduce the network traffic by just sending the compressed data.
- **RDMA:** Uses a specific NIC to increase the efficiency of data transmission.
- **Multi thread compression:** Compresses the data before transmission.
- **Auto convergence:** Reduces the data rate of dirty pages.

Tests show none of the above features can satisfy the requirement of NFV. XBZRLE and Multi thread compression do the compression entirely in software and they are not fast enough in a 10Gbps network environment. RDMA is not flexible because it has to transport all the guest memory to the destination without zero page optimization. Auto convergence is not appropriate for NFV because it will impact guest's performance.

So we need to find other ways for optimization.





11.8.2 Optimizations

- Delay non-emergency operations By profiling, it was discovered that some of the cleanup operations during the stop and copy stage are the main reason for the long VM down time. The cleanup operation includes stopping the dirty page logging, which is a time consuming operation. By deferring these operations until the data transmission is completed the VM down time is reduced to about 5-7ms.
- Optimize zero page checking Currently QEMU uses the SSE2 instruction to optimize the zero pages checking. The SSE2 instruction can process 16 bytes per instruction. By using the AVX2 instruction, we can process 32 bytes per instruction. Testing shows that using AVX2 can speed up the zero pages checking process by about 25%.
- Remove unnecessary context synchronization. The CPU context was being synchronized twice during live migration. Removing this unnecessary synchronization shortened the VM downtime by about 100us.

11.8.3 Test Environment

The source and destination host have the same hardware and OS: :: Host: HSW-EP CPU: Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz RAM: 64G OS: RHEL 7.1 Kernel: 4.2 QEMU v2.4.0

Ethernet controller: Intel Corporation Ethernet Controller 10-Gigabit X540-AT2 (rev 01)

Vhost-user with OVS/DPDK as backend: :: The goal is to connect guests' virtio-net devices having vhost-user backend to OVS dpdkvhostuser ports and be able to run any kind of network traffic between them.

Installation of OVS and DPDK: :: Using vsperf, installing the OVS and DPDK. Prepare the directories

```
mkdir -p /var/run/openvswitch
mount -t hugetlbfs -o pagesize=2048k none /dev/hugepages
```

Load Kernel modules

```
modprobe openvswitch
```

For OVS setup, clean the environment

```
rm -f /usr/local/var/run/openvswitch/vhost-user*
rm -f /usr/local/etc/openvswitch/conf.db
```

Start database server

```
ovsdb-tool create /usr/local/etc/openvswitch/conf.db $VSPERF/src/ovs/ovs/vswitchd/
↪vswitch.ovsschema
ovsdb-server --remote=punix:$DB_SOCKET --remote=db:Open_vSwitch,Open_vSwitch,manager_
↪options --pidfile --detach
```

Start OVS

```
ovs-vsctl --no-wait init
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-lcore-mask=0xf
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-socket-mem=1024
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true
```

Configure the bridge

```
ovs-vsctl add-br ovsbr0 -- set bridge ovsbr0 datapath_type=netdev
ovs-vsctl add-port ovsbr0 vhost-user1 -- set Interface vhost-user1 type=dpdkvhostuser
ovs-vsctl add-port ovsbr0 vhost-user2 -- set Interface vhost-user2 type=dpdkvhostuser
```

QEMU parameters: :: qemu-system-x86_64 -enable-kvm -cpu host -smp 2 -chardev socket,id=char1,path=/usr/local/var/run/openvswitch/vhost-user1 -netdev type=vhost-user,id=net1,chardev=char1,vhostforce -device virtio-net-pci,netdev=net1,mac=52:54:00:12:34:56 -chardev socket,id=char2,path=/usr/local/var/run/openvswitch/vhost-user2-netdev type=vhost-user,id=net2,chardev=char2,vhostforce -device virtio-net-pci,netdev=net2,mac=54:54:00:12:34:56 -m 1024 -mem-path /dev/hugepages -mem-prealloc -realtime mlock=on -monitor unix:/tmp/qmp-sock-src,server,nowait -balloon virtio -drive file=/root/guest1.qcow2 -vnc :1 &

Run the standby qemu with -incoming tcp:-incoming tcp:\${incoming_ip}:\${migrate_port}:\${migrate_port}

For local live migration

```
incoming ip=0
```

For peer -peer live migration

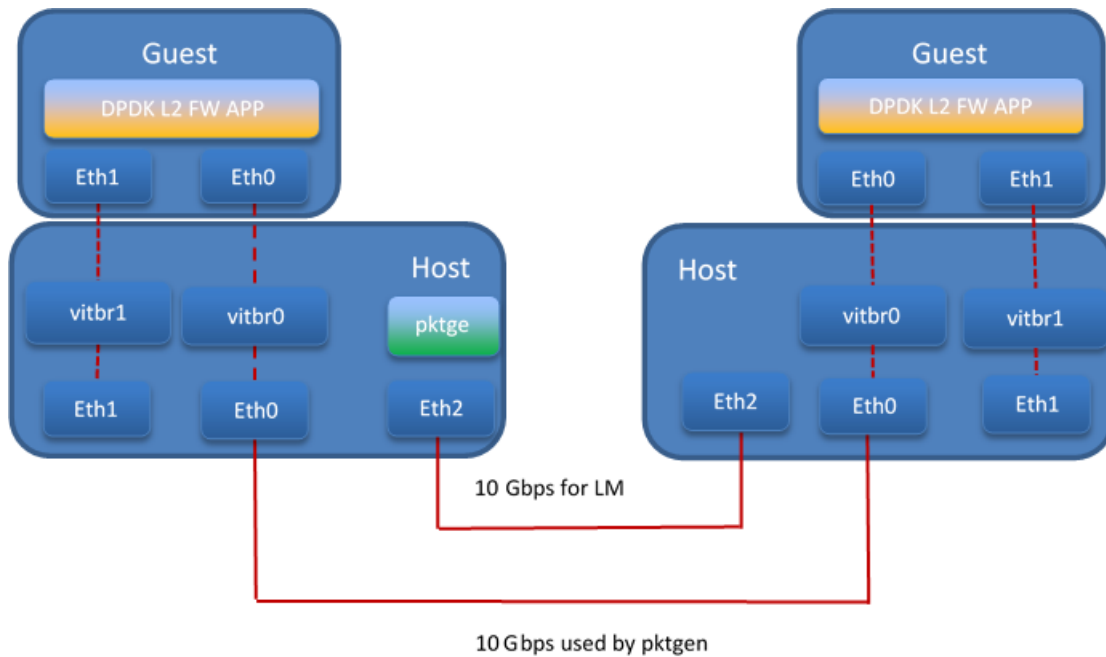
```
incoming ip=dest_host
```

Network connection

Commands for performing live migration:

```
.. code:: bash
```

```
echo "migrate_set_speed 0" | nc -U /tmp/qmp-sock-src echo "migrate_set_downtime 0.10" | nc -U /tmp/qmp-sock-src echo "migrate -d tcp:0:4444" | nc -U /tmp/qmp-sock-src #Wait till livemigration completed echo "info migrate" | nc -U /tmp/qmp-sock-src
```

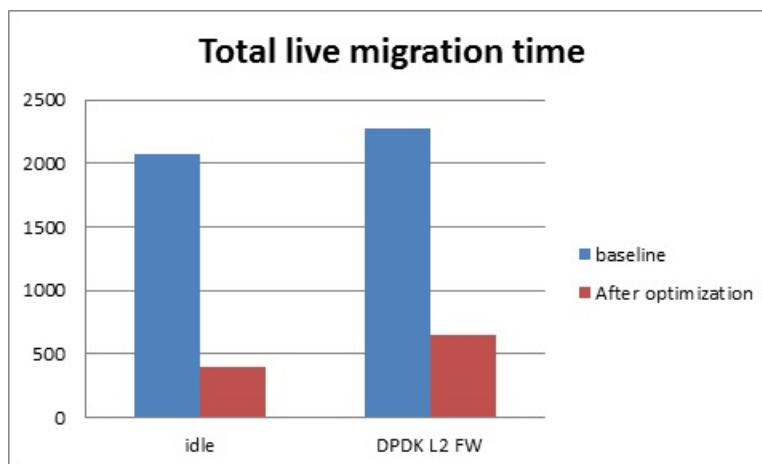


11.8.4 Test Result

The down time is set to 10ms when doing the test. We use pktgen to send the packages to guest, the package size is 64 bytes, and the line rate is 1013 Mbps.

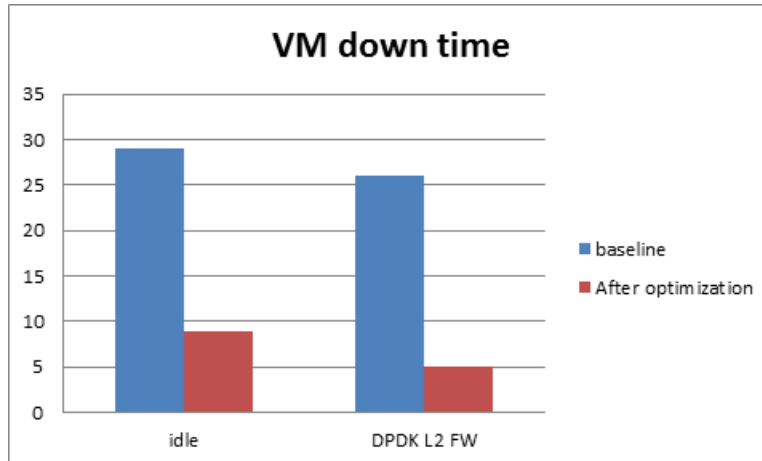
a. Total live migration time

The total live migration time before and after optimization is shown in the chart below. For an idle guest, we can reduce the total live migration time from 2070ms to 401ms. For a guest running the DPDK L2 forwarding workload, the total live migration time is reduced from 2271ms to 654ms.



b. VM downtime

The VM down time before and after optimization is shown in the chart below. For an idle guest, we can reduce the VM down time from 29ms to 9ms. For a guest running the DPDK L2 forwarding workload, the VM down time is reduced from 26ms to 5ms.



11.9 Euphrates OpenStack User Guide

OpenStack is a cloud operating system developed and released by the [OpenStack project](#). OpenStack is used in OPNFV for controlling pools of compute, storage, and networking resources in a Pharos compliant infrastructure.

OpenStack is used in Euphrates to manage tenants (known in OpenStack as projects), users, services, images, flavours, and quotas across the Pharos infrastructure. The OpenStack interface provides the primary interface for an operational Euphrates deployment and it is from the “horizon console” that an OPNFV user will perform the majority of administrative and operational activities on the deployment.

11.9.1 OpenStack references

The [OpenStack user guide](#) provides details and descriptions of how to configure and interact with the OpenStack deployment. This guide can be used by lab engineers and operators to tune the OpenStack deployment to your liking.

Once you have configured OpenStack to your purposes, or the Euphrates deployment meets your needs as deployed, an operator, or administrator, will find the best guidance for working with OpenStack in the [OpenStack administration guide](#).

11.9.2 Connecting to the OpenStack instance

Once familiar with the basic of working with OpenStack you will want to connect to the OpenStack instance via the Horizon Console. The Horizon console provide a Web based GUI that will allow you operate the deployment. To do this you should open a browser on the JumpHost to the following address and enter the username and password:

`http://{Controller-VIP}:80/index.html` username: admin password: admin

Other methods of interacting with and configuring OpenStack, like the REST API and CLI are also available in the Euphrates deployment, see the [OpenStack administration guide](#) for more information on using those interfaces.

11.10 Packet Forwarding

11.10.1 About Packet Forwarding

Packet Forwarding is a test suite of KVM4NFV. These latency tests measures the time taken by a **Packet** generated by the traffic generator to travel from the originating device through the network to the destination device. Packet

Forwarding is implemented using test framework implemented by OPNFV VSWITCHPERF project and an IXIA Traffic Generator.

11.10.2 Version Features

Release	Features
Colorado	<ul style="list-style-type: none"> • Packet Forwarding is not part of Colorado release of KVM4NFV
Danube	<ul style="list-style-type: none"> • Packet Forwarding is a testcase in KVM4NFV • Implements three scenarios (Host/Guest/SRIOV) as part of testing in KVM4NFV • Uses automated test framework of OPNFV VSWITCHPERF software (PVP/PVVP) • Works with IXIA Traffic Generator
Euphrates	<ul style="list-style-type: none"> • Test cases involving multiple guests (PVVP/PVPV) included. • Implemented Yardstick Grafana dashboard to publish results of packet forwarding test cases

11.10.3 VSPERF

VSPerf is an OPNFV testing project. VSPerf will develop a generic and architecture agnostic vSwitch testing framework and associated tests, that will serve as a basis for validating the suitability of different vSwitch implementations in a Telco NFV deployment environment. The output of this project will be utilized by the OPNFV Performance and Test group and its associated projects, as part of OPNFV Platform and VNF level testing and validation.

For complete VSPERF documentation go to [link](#).

11.10.3.1 Installation

Guidelines of installing [VSPERF](#).

11.10.3.2 Supported Operating Systems

- CentOS 7
- Fedora 20
- Fedora 21
- Fedora 22
- RedHat 7.2
- Ubuntu 14.04

11.10.3.3 Supported vSwitches

The vSwitch must support Open Flow 1.3 or greater.

- OVS (built from source).
- OVS with DPDK (built from source).

11.10.3.4 Supported Hypervisors

- Qemu version 2.6.

11.10.3.5 Other Requirements

The test suite requires Python 3.3 and relies on a number of other packages. These need to be installed for the test suite to function.

Installation of required packages, preparation of Python 3 virtual environment and compilation of OVS, DPDK and QEMU is performed by script **systems/build_base_machine.sh**. It should be executed under user account, which will be used for vsperf execution.

Please Note: Password-less sudo access must be configured for given user before script is executed.

Execution of installation script:

```
$ cd vswitchperf
$ cd systems
$ ./build_base_machine.sh
```

Script **build_base_machine.sh** will install all the vsperf dependencies in terms of system packages, Python 3.x and required Python modules. In case of CentOS 7 it will install Python 3.3 from an additional repository provided by Software Collections ([a link](#)). In case of RedHat 7 it will install Python 3.4 as an alternate installation in /usr/local/bin. Installation script will also use [virtualenv](#) to create a vsperf virtual environment, which is isolated from the default Python environment. This environment will reside in a directory called **vsperfenv** in \$HOME.

You will need to activate the virtual environment every time you start a new shell session. Its activation is specific to your OS:

For running testcases VSPERF is installed on Intel pod1-node2 in which centos operating system is installed. Only VSPERF installion on Centos is discussed here. For installation steps on other operating systems please refer to [here](#).

11.10.3.6 For CentOS 7

Python 3 Packages

To avoid file permission errors and Python version issues, use virtualenv to create an isolated environment with Python3. The required Python 3 packages can be found in the *requirements.txt* file in the root of the test suite. They can be installed in your virtual environment like so:

```
scl enable python33 bash
# Create virtual environment
virtualenv vsperfenv
cd vsperfenv
source bin/activate
pip install -r requirements.txt
```

You need to activate the virtual environment every time you start a new shell session. To activate, simple run:

```
scl enable python33 bash
cd vsperfenv
source bin/activate
```

11.10.3.7 Working Behind a Proxy

If you're behind a proxy, you'll likely want to configure this before running any of the above. For example:

```
export http_proxy="http://<username>:<password>@<proxy>:<port>/" ;
export https_proxy="https://<username>:<password>@<proxy>:<port>/" ;
export ftp_proxy="ftp://<username>:<password>@<proxy>:<port>/" ;
export socks_proxy="socks://<username>:<password>@<proxy>:<port>/" ;
```

For other OS specific activation click *this link*:

<http://artifacts.opnfv.org/vswitchperf/colorado/configguide/installation.html#other-requirements>

11.10.4 Traffic-Generators

VSPERF supports many Traffic-generators. For configuring VSPERF to work with the available traffic generator go through *this*.

VSPERF supports the following traffic generators:

- Dummy (DEFAULT): Allows you to use your own external traffic generator.
- IXIA (IxNet and IxOS)
- Spirent TestCenter
- Xena Networks
- MoonGen

To see the list of traffic gens from the cli:

```
$ ./vsperf --list-trafficgens
```

This guide provides the details of how to install and configure the various traffic generators.

As KVM4NFV uses only IXIA traffic generator, it is discussed here. For complete documentation regarding traffic generators please follow this *link*.

11.10.5 IXIA Setup

11.10.5.1 Hardware Requirements

VSPERF requires the following hardware to run tests: IXIA traffic generator (IxNetwork), a machine that runs the IXIA client software and a CentOS Linux release 7.1.1503 (Core) host.

11.10.5.2 Installation

Follow the installation instructions to install.

11.10.5.3 On the CentOS 7 system

You need to install `IxNetworkTclClient$(VER_NUM)Linux.bin.tgz`.

11.10.5.4 On the IXIA client software system

Find the IxNetwork TCL server app

- (start -> All Programs -> IXIA -> IxNetwork -> IxNetwork_\$(VER_NUM) -> IxNetwork TCL Server)
- Right click on IxNetwork TCL Server, select properties
- Under shortcut tab in the Target dialogue box make sure there is the argument “-tcpport xxxx”

where xxxx is your port number (take note of this port number you will need it for the `10_custom.conf` file).

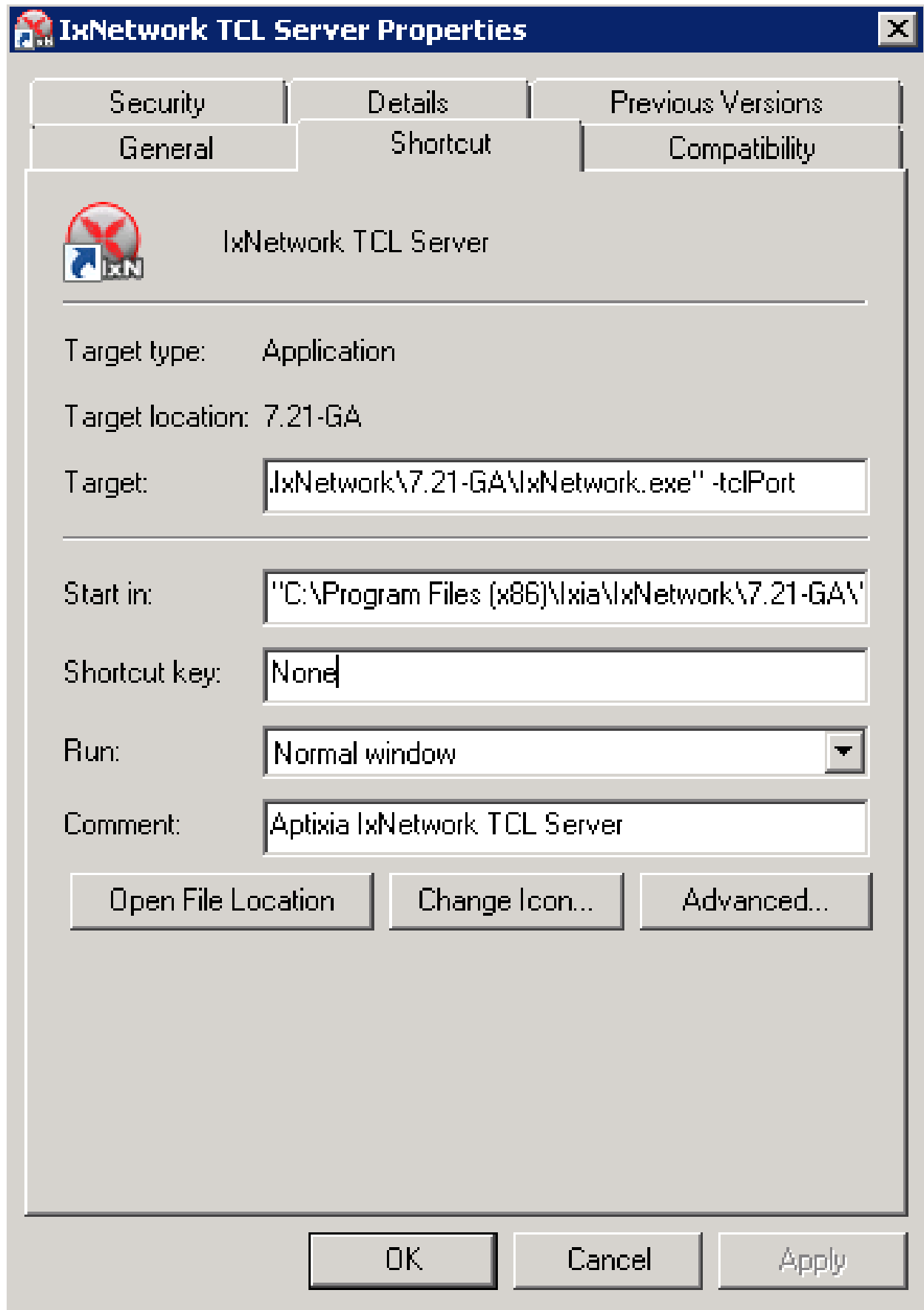
- Hit Ok and start the TCL server application

11.10.6 VSPERF configuration

There are several configuration options specific to the IxNetworks traffic generator from IXIA. It is essential to set them correctly, before the VSPERF is executed for the first time.

Detailed description of options follows:

- `TRAFFICGEN_IXNET_MACHINE` - IP address of server, where IxNetwork TCL Server is running
- `TRAFFICGEN_IXNET_PORT` - PORT, where IxNetwork TCL Server is accepting connections from TCL clients
- `TRAFFICGEN_IXNET_USER` - username, which will be used during communication with IxNetwork TCL Server and IXIA chassis
- `TRAFFICGEN_IXIA_HOST` - IP address of IXIA traffic generator chassis
- `TRAFFICGEN_IXIA_CARD` - identification of card with dedicated ports at IXIA chassis
- `TRAFFICGEN_IXIA_PORT1` - identification of the first dedicated port at `TRAFFICGEN_IXIA_CARD` at IXIA chassis; VSPERF uses two separated ports for traffic generation. In case of unidirectional traffic, it is essential to correctly connect 1st IXIA port to the 1st NIC at DUT, i.e. to the first PCI handle from `WHITELIST_NICS` list. Otherwise traffic may not be able to pass through the vSwitch.
- `TRAFFICGEN_IXIA_PORT2` - identification of the second dedicated port at `TRAFFICGEN_IXIA_CARD` at IXIA chassis; VSPERF uses two separated ports for traffic generation. In case of unidirectional traffic, it is essential to correctly connect 2nd IXIA port to the 2nd NIC at DUT, i.e. to the second PCI handle from `WHITELIST_NICS` list. Otherwise traffic may not be able to pass through the vSwitch.
- `TRAFFICGEN_IXNET_LIB_PATH` - path to the DUT specific installation of IxNetwork TCL API
- `TRAFFICGEN_IXNET_TCL_SCRIPT` - name of the TCL script, which VSPERF will use for communication with IXIA TCL server
- `TRAFFICGEN_IXNET_TESTER_RESULT_DIR` - folder accessible from IxNetwork TCL server, where test results are stored, e.g. `c:/ixia_results`; see [test-results-share](#)
- `TRAFFICGEN_IXNET_DUT_RESULT_DIR` - directory accessible from the DUT, where test results from IxNetwork TCL server are stored, e.g. `/mnt/ixia_results`; see [test-results-share](#)



11.10.6.1 Test results share

VSPERF is not able to retrieve test results via TCL API directly. Instead, all test results are stored at IxNetwork TCL server. Results are stored at folder defined by `TRAFFICGEN_IXNET_TESTER_RESULT_DIR` configuration parameter. Content of this folder must be shared (e.g. via samba protocol) between TCL Server and DUT, where VSPERF is executed. VSPERF expects, that test results will be available at directory configured by `TRAFFICGEN_IXNET_DUT_RESULT_DIR` configuration parameter.

Example of sharing configuration:

- Create a new folder at IxNetwork TCL server machine, e.g. `c:\ixia_results`
- Modify sharing options of `ixia_results` folder to share it with everybody
- Create a new directory at DUT, where shared directory with results will be mounted, e.g. `/mnt/ixia_results`
- Update your custom VSPERF configuration file as follows:

```
TRAFFICGEN_IXNET_TESTER_RESULT_DIR = 'c:/ixia_results'
TRAFFICGEN_IXNET_DUT_RESULT_DIR = '/mnt/ixia_results'
```

Note: It is essential to use slashes '/' also in path configured by `TRAFFICGEN_IXNET_TESTER_RESULT_DIR` parameter.

- Install `cifs-utils` package.

e.g. at rpm based Linux distribution:

```
yum install cifs-utils
```

- Mount shared directory, so VSPERF can access test results.

e.g. by adding new record into `/etc/fstab`

```
mount -t cifs //_TCL_SERVER_IP_OR_FQDN/ixia_results /mnt/ixia_results
-o file_mode=0777,dir_mode=0777,nounix
```

It is recommended to verify, that any new file inserted into `c:/ixia_results` folder is visible at DUT inside `/mnt/ixia_results` directory.

11.10.6.2 Cloning and building src dependencies

In order to run VSPERF, you will need to download DPDK and OVS. You can do this manually and build them in a preferred location, or you could use `vswitchperf/src`. The `vswitchperf/src` directory contains makefiles that will allow you to clone and build the libraries that VSPERF depends on, such as DPDK and OVS. To clone and build simply:

```
cd src
make
```

To delete a src subdirectory and its contents to allow you to re-clone simply use:

```
make cleanse
```

11.10.6.3 Configure the `./conf/10_custom.conf` file

The supplied `10_custom.conf` file must be modified, as it contains configuration items for which there are no reasonable default values.

The configuration items that can be added is not limited to the initial contents. Any configuration item mentioned in any `.conf` file in `./conf` directory can be added and that item will be overridden by the custom configuration value.

11.10.6.4 Using a custom settings file

Alternatively a custom settings file can be passed to `vsperf` via the `-conf-file` argument.

```
./vsperf --conf-file <path_to_settings_py> ...
```

Note that configuration passed in via the environment (`-load-env`) or via another command line argument will override both the default and your custom configuration files. This “priority hierarchy” can be described like so (1 = max priority):

1. Command line arguments
2. Environment variables
3. Configuration file(s)

11.10.6.5 vloop_vnf

VSPERF uses a VM image called `vloop_vnf` for looping traffic in the deployment scenarios involving VMs. The image can be downloaded from <http://artifacts.opnfv.org/>.

Please see the installation instructions for information on `vloop-vnf` images.

11.10.6.6 l2fwd Kernel Module

A Kernel Module that provides OSI Layer 2 Ipv4 termination or forwarding with support for Destination Network Address Translation (DNAT) for both the MAC and IP addresses. `l2fwd` can be found in `<vswitchperf_dir>/src/l2fwd`

11.10.6.7 Executing tests

Before running any tests make sure you have root permissions by adding the following line to `/etc/sudoers`:

```
username ALL=(ALL) NOPASSWD: ALL
```

`username` in the example above should be replaced with a real username.

To list the available tests:

```
./vsperf --list-tests
```

To run a group of tests, for example all tests with a name containing ‘RFC2544’:

```
./vsperf --conf-file=user_settings.py --tests="RFC2544"
```

To run all tests:

```
./vsperf --conf-file=user_settings.py
```

Some tests allow for configurable parameters, including test duration (in seconds) as well as packet sizes (in bytes).

```
./vsperf --conf-file user_settings.py
--tests RFC2544Tput
--test-param` "rfc2544_duration=10;packet_sizes=128"
```

For all available options, check out the help dialog:

```
./vsperf --help
```

11.10.7 Testcases

Available Tests in VSPERF are:

- phy2phy_tput
- phy2phy_forwarding
- back2back
- phy2phy_tput_mod_vlan
- phy2phy_cont
- pvp_cont
- pvvp_cont
- pvpv_cont
- phy2phy_scalability
- pvp_tput
- pvp_back2back
- pvvp_tput
- pvvp_back2back
- phy2phy_cpu_load
- phy2phy_mem_load

11.10.8 VSPERF modes of operation

VSPERF can be run in different modes. By default it will configure vSwitch, traffic generator and VNF. However it can be used just for configuration and execution of traffic generator. Another option is execution of all components except traffic generator itself.

Mode of operation is driven by configuration parameter `-m` or `--mode`

```
-m MODE, --mode MODE  vsperf mode of operation;
  Values:
    "normal" - execute vSwitch, VNF and traffic generator
    "trafficgen" - execute only traffic generator
    "trafficgen-off" - execute vSwitch and VNF
    "trafficgen-pause" - execute vSwitch and VNF but wait before traffic_
↳transmission
```


In case, that VSPERF is executed in “trafficgen” mode, then configuration of traffic generator can be modified through TRAFFIC dictionary passed to the `--test-params` option. It is not needed to specify all values of TRAFFIC dictionary. It is sufficient to specify only values, which should be changed. Detailed notes on TRAFFIC dictionary can be found at: [ref:configuration-of-traffic-dictionary](#).

Example of execution of VSPERF in “trafficgen” mode:

```
$ ./vsperf -m trafficgen --trafficgen IxNet --conf-file vsperf.conf \
  --test-params "TRAFFIC={'traffic_type':'rfc2544_continuous','bidir':'False',
  ↪ 'framerate':60}"
```

11.10.9 Packet Forwarding Test Scenarios

KVM4NFV currently implements three scenarios as part of testing:

- Host Scenario
- Guest Scenario.
- SR-IOV Scenario.

11.10.9.1 Packet Forwarding Host Scenario

Here host DUT has VSPERF installed in it and is properly configured to use IXIA Traffic-generator by providing IXIA CARD, PORTS and Lib paths along with IP. please refer to figure.2

11.10.9.2 Packet Forwarding Guest Scenario (PXP Deployment)

Here the guest is a Virtual Machine (VM) launched by using `vloop_vnf` provided by `vsperf` project on host/DUT using Qemu. In this latency test the time taken by the frame/packet to travel from the originating device through network involving a guest to destination device is calculated. The resulting latency values will define the performance of installed kernel.

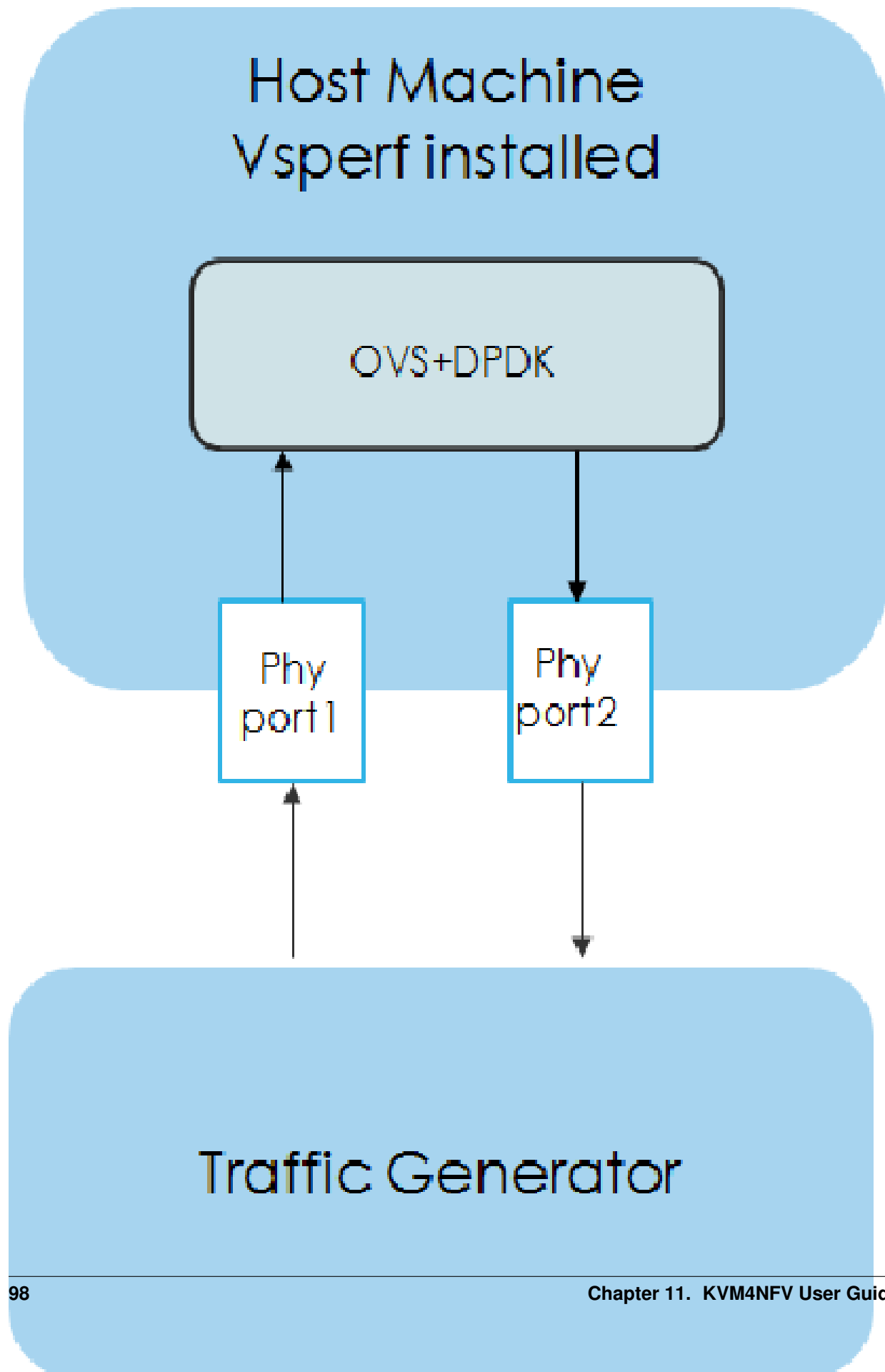
Every testcase uses one of the supported deployment scenarios to setup test environment. The controller responsible for a given scenario configures flows in the vswitch to route traffic among physical interfaces connected to the traffic generator and virtual machines. VSPERF supports several deployments including PXP deployment, which can setup various scenarios with multiple VMs.

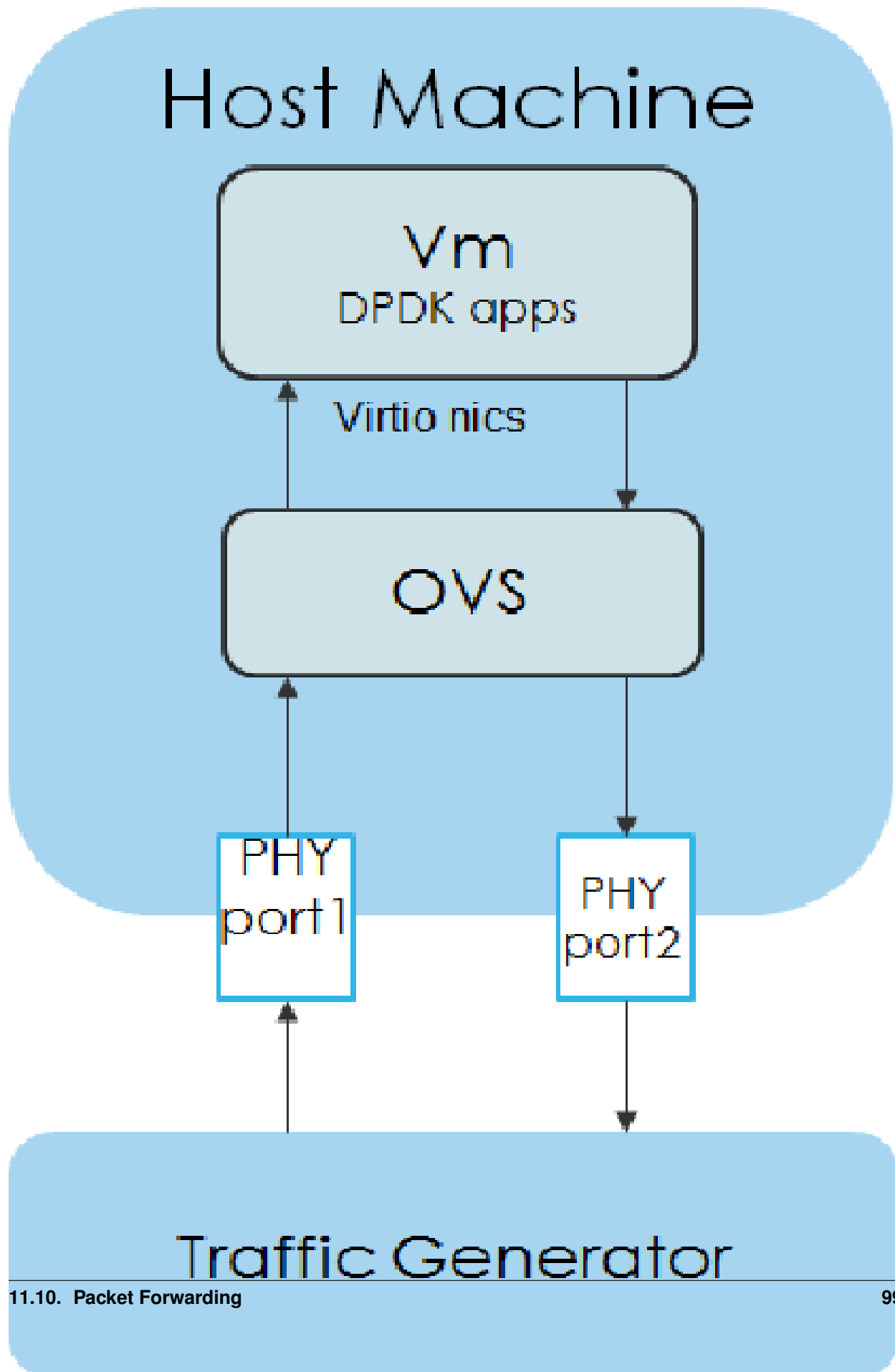
These scenarios are realized by `VswitchControllerPXP` class, which can configure and execute given number of VMs in serial or parallel configurations. Every VM can be configured with just one or an even number of interfaces. In case that VM has more than 2 interfaces, then traffic is properly routed among pairs of interfaces.

Example of traffic routing for VM with 4 NICs in serial configuration:

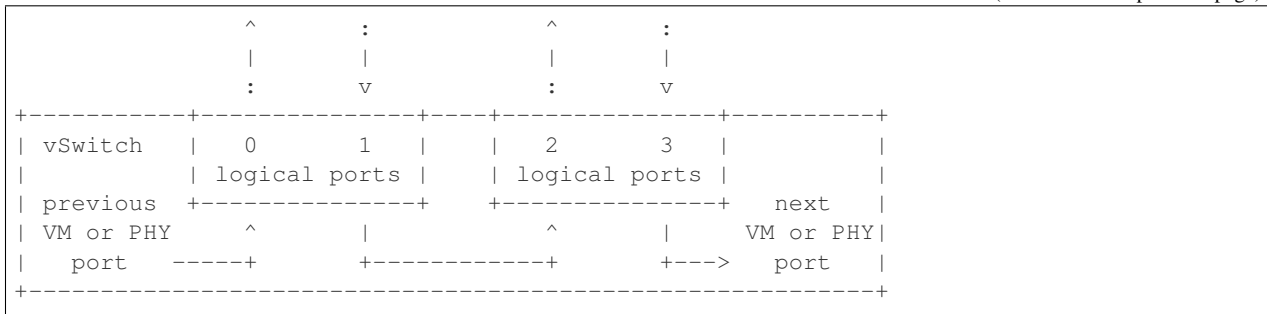
```
+-----+
| VM with 4 NICs                                     |
| +-----+ +-----+                               |
| | Application | | Application | |                 |
| +-----+ +-----+                               |
|   ^       |       ^       |                       |
|   |       v       |       v       |               |
| +-----+ +-----+                               |
| | logical ports | | logical ports | |             |
| | 0       1   | | 2       3   | |                 |
| +-----+ +-----+                               |
+-----+
```

(continues on next page)





(continued from previous page)

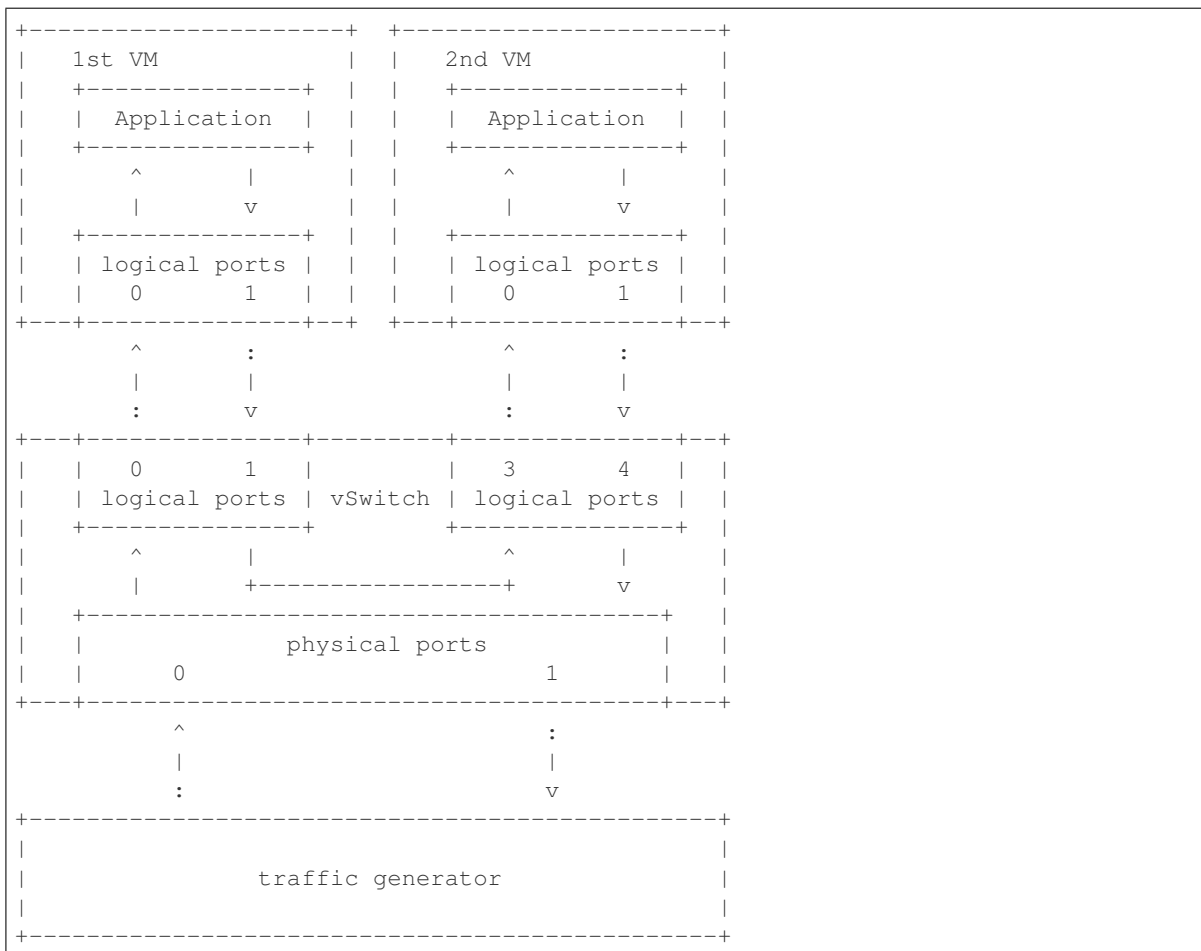


It is also possible to define different number of interfaces for each VM to better simulate real scenarios.

The number of VMs involved in the test and the type of their connection is defined by deployment name as follows:

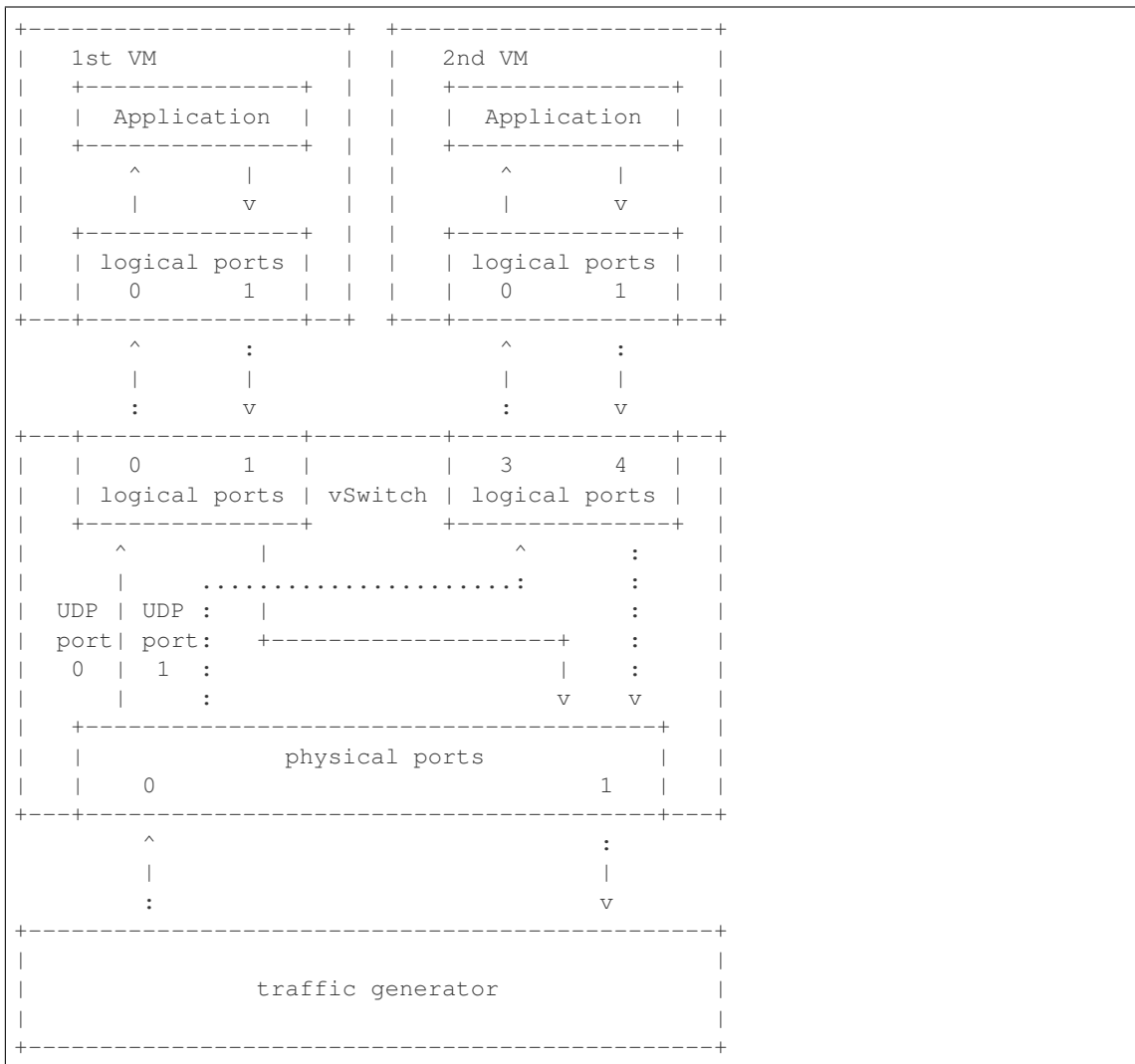
- **pvvp[number]** - configures scenario with VMs connected in series with optional number of VMs. In case that number is not specified, then 2 VMs will be used.

Example of 2 VMs in a serial configuration:



- **pvvp[number]** - configures scenario with VMs connected in parallel with optional number of VMs. In case that number is not specified, then 2 VMs will be used. Multistream feature is used to route traffic to particular VMs (or NIC pairs of every VM). It means, that VSPERF will enable multistream feature and sets the number of streams to the number of VMs and their NIC pairs. Traffic will be dispatched based on Stream Type, i.e. by UDP port, IP address or MAC address.

Example of 2 VMs in a parallel configuration, where traffic is dispatched based on the UDP port.



PXP deployment is backward compatible with PVP deployment, where `pvp` is an alias for `pvpv1` and it executes just one VM.

The number of interfaces used by VMs is defined by configuration option `GUEST_NICS_NR`. In case that more than one pair of interfaces is defined for VM, then:

- for `pvpv` (serial) scenario every NIC pair is connected in serial before connection to next VM is created
- for `pvpv` (parallel) scenario every NIC pair is directly connected to the physical ports and unique traffic stream is assigned to it

Examples:

- Deployment `pvpv10` will start 10 VMs and connects them in series
- Deployment `pvpv4` will start 4 VMs and connects them in parallel
- Deployment `pvpv1` and `GUEST_NICS_NR = [4]` will start 1 VM with 4 interfaces and every NIC pair is directly connected to the physical ports
- Deployment `pvpv` and `GUEST_NICS_NR = [2, 4]` will start 2 VMs; 1st VM will have 2 interfaces and 2nd VM 4 interfaces. These interfaces will be connected in serial, i.e. traffic will flow as follows: `PHY1 -> VM1_1`

-> VM1_2 -> VM2_1 -> VM2_2 -> VM2_3 -> VM2_4 -> PHY2

Note: In case that only 1 or more than 2 NICs are configured for VM, then `testpmd` should be used as forwarding application inside the VM. As it is able to forward traffic between multiple VM NIC pairs.

Note: In case of `linux_bridge`, all NICs are connected to the same bridge inside the VM.

11.10.9.3 Packet Forwarding SRIOV Scenario

In this test the packet generated at the IXIA is forwarded to the Guest VM launched on Host by implementing SR-IOV interface at NIC level of host i.e., DUT. The time taken by the packet to travel through the network to the destination the IXIA traffic-generator is calculated and published as a test result for this scenario.

SRIOV-support is given below, it details how to use SR-IOV.

11.10.9.4 Using vfio_pci with DPDK

To use vfio with DPDK instead of `igb_uio` add into your custom configuration file the following parameter:

```
PATHS['dpdk']['src']['modules'] = ['uio', 'vfio-pci']
```

NOTE: In case, that DPDK is installed from binary package, then please

set `PATHS['dpdk']['bin']['modules']` instead.

NOTE: Please ensure that Intel VT-d is enabled in BIOS.

NOTE: Please ensure your boot/grub parameters include the following:

```
iommu=pt intel_iommu=on
```

To check that IOMMU is enabled on your platform:

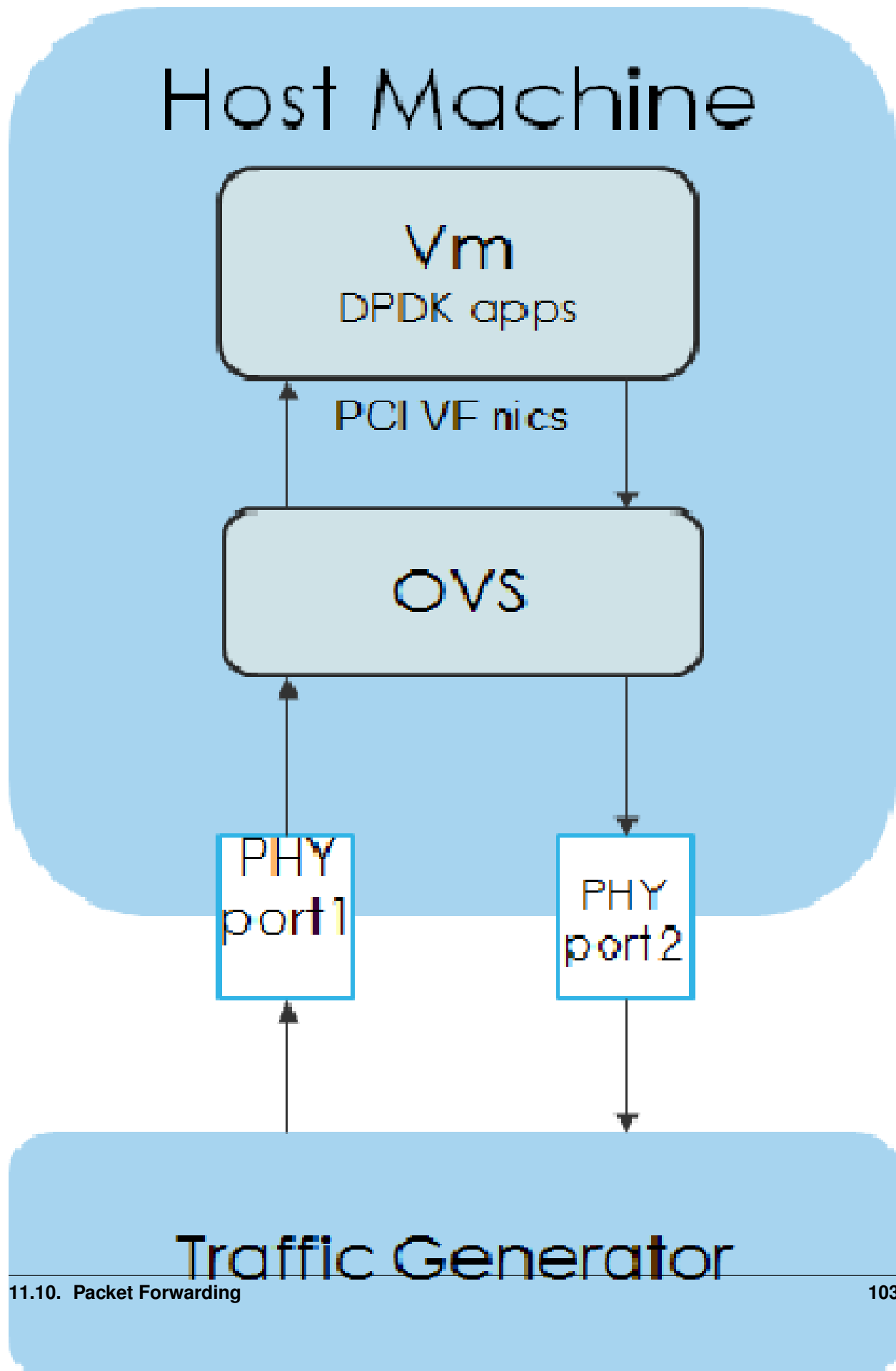
```
$ dmesg | grep IOMMU
[ 0.000000] Intel-IOMMU: enabled
[ 0.139882] dmar: IOMMU 0: reg_base_addr fbffe000 ver 1:0 cap d2078c106f0466 ecap_
↪f020de
[ 0.139888] dmar: IOMMU 1: reg_base_addr ebffc000 ver 1:0 cap d2078c106f0466 ecap_
↪f020de
[ 0.139893] IOAPIC id 2 under DRHD base 0xfbf000 IOMMU 0
[ 0.139894] IOAPIC id 0 under DRHD base 0xebffc000 IOMMU 1
[ 0.139895] IOAPIC id 1 under DRHD base 0xebffc000 IOMMU 1
[ 3.335744] IOMMU: dmar0 using Queued invalidation
[ 3.335746] IOMMU: dmar1 using Queued invalidation
....
```

11.10.9.5 Using SRIOV support

To use virtual functions of NIC with SRIOV support, use extended form of NIC PCI slot definition:

```
WHITELIST_NICS = ['0000:03:00.0|vf0', '0000:03:00.1|vf3']
```

Where `vf` is an indication of virtual function usage and following number defines a VF to be used. In case that VF usage is detected, then `vswitchperf` will enable SRIOV support for given card and it will detect PCI slot numbers of selected VFs.



So in example above, one VF will be configured for NIC '0000:05:00.0' and four VFs will be configured for NIC '0000:05:00.1'. Vswitchperf will detect PCI addresses of selected VFs and it will use them during test execution.

At the end of vswitchperf execution, SRIOV support will be disabled.

SRIOV support is generic and it can be used in different testing scenarios. For example:

- vSwitch tests with DPDK or without DPDK support to verify impact of VF usage on vSwitch performance
- tests without vSwitch, where traffic is forwarded directly between VF interfaces by packet forwarder (e.g. testpmd application)
- tests without vSwitch, where VM accesses VF interfaces directly by PCI-passthrough to measure raw VM throughput performance.

Using QEMU with PCI passthrough support

Raw virtual machine throughput performance can be measured by execution of PVP test with direct access to NICs by PCI passthrough. To execute VM with direct access to PCI devices, enable vfio-pci. In order to use virtual functions, *SRIOV-support* must be enabled.

Execution of test with PCI passthrough with vswitch disabled:

```
$ ./vsperf --conf-file=<path_to_custom_conf>/10_custom.conf \
--vswitch none --vnf QemuPciPassthrough pvp_tput
```

Any of supported guest-loopback-application can be used inside VM with PCI passthrough support.

Note: Qemu with PCI passthrough support can be used only with PVP test deployment.

Guest Core and Thread Binding

VSPERF provides options to achieve better performance by guest core binding and guest vCPU thread binding as well. Core binding is to bind all the qemu threads. Thread binding is to bind the house keeping threads to some CPU and vCPU thread to some other CPU, this helps to reduce the noise from qemu house keeping threads.

```
GUEST_CORE_BINDING = [ ('#EVAL(6+2*#VMINDEX)', '#EVAL(7+2*#VMINDEX)') ]
```

NOTE By default the GUEST_THREAD_BINDING will be none, which means same as the GUEST_CORE_BINDING, i.e. the vcpu threads are sharing the physical CPUs with the house keeping threads. Better performance using vCPU thread binding can be achieved by enabling affinity in the custom configuration file.

For example, if an environment requires 28,29 to be core binded and 30,31 for guest thread binding to achieve better performance.

```
VNF_AFFINITIZATION_ON = True
GUEST_CORE_BINDING = [ ('28', '29') ]
GUEST_THREAD_BINDING = [ ('30', '31') ]
```

Qemu CPU features

QEMU default to a compatible subset of performance enhancing cpu features. To pass all available host processor features to the guest.

```
GUEST_CPU_OPTIONS = ['host,migratable=off']
```


NOTE To enhance the performance, cpu features tsc deadline timer for guest, the guest PMU, the invariant TSC can be provided in the custom configuration file.

Selection of loopback application for tests with VMs

To select the loopback applications which will forward packets inside VMs, the following parameter should be configured:

```
GUEST_LOOPBACK = ['testpmd']
```

or use `--test-params` CLI argument:

```
$ ./vsperf --conf-file=<path_to_custom_conf>/10_custom.conf \
  --test-params "GUEST_LOOPBACK=['testpmd']"
```

Supported loopback applications are:

'testpmd'	- testpmd from dpdk will be built and used
'l2fwd'	- l2fwd module provided by Huawei will be built and used
'linux_bridge'	- linux bridge will be configured
'buildin'	- nothing will be configured by vsperf; VM image must ensure traffic forwarding between its interfaces

Guest loopback application must be configured, otherwise traffic will not be forwarded by VM and testcases with VM related deployments will fail. Guest loopback application is set to 'testpmd' by default.

NOTE: In case that only 1 or more than 2 NICs are configured for VM, then 'testpmd' should be used. As it is able to forward traffic between multiple VM NIC pairs.

NOTE: In case of linux_bridge, all guest NICs are connected to the same bridge inside the guest.

11.10.9.6 Results

The results for the packet forwarding test cases are uploaded to artifacts and also published on Yardstick Grafana dashboard. The links for the same can be found below

```
http://artifacts.opnfv.org/kvmfornfv.html
http://testresults.opnfv.org/KVMFORNFV-Packet-Forwarding
```

11.11 PCM Utility in KVM4NFV

11.11.1 Collecting Memory Bandwidth Information using PCM utility

This chapter includes how the PCM utility is used in kvm4nfv to collect memory bandwidth information

11.11.2 About PCM utility

The Intel® Performance Counter Monitor provides sample C++ routines and utilities to estimate the internal resource utilization of the latest Intel® Xeon® and Core™ processors and gain a significant performance boost. In Intel PCM toolset, there is a pcm-memory.x tool which is used for observing the memory traffic intensity

11.11.3 Version Features

Release	Features
Colorado	<ul style="list-style-type: none">• In Colorado release, we don't have memory bandwidth information collected through the cyclic testcases.
Danube	<ul style="list-style-type: none">• pcm-memory.x will be executed before the execution of every testcase• pcm-memory.x provides the memory bandwidth data throughout out the testcases• used for all test-types (stress/idle)• Generated memory bandwidth logs are published to the KVMFORNFV artifacts

11.11.3.1 Implementation of pcm-memory.x:

The tool measures the memory bandwidth observed for every channel reporting separate throughput for reads from memory and writes to the memory. pcm-memory.x tool tends to report values slightly higher than the application's own measurement.

Command:

```
sudo ./pcm-memory.x [Delay]/[external_program]
```

Parameters

- pcm-memory can be called with either delay or external_program/application as a parameter
- If delay is given as 5, then the output will be produced with refresh of every 5 seconds.
- If external_program is script/application, then the output will be produced after the execution of the application or the script passed as a parameter.

Sample Output:

The output produced with default refresh of 1 second.

Socket 0	Socket 1
Memory Performance Monitoring	Memory Performance Monitoring
Mem Ch 0: Reads (MB/s): 6870.81 Writes(MB/s): 1805.03	Mem Ch 0: Reads (MB/s): 7406.36 Writes(MB/s): 1951.25
Mem Ch 1: Reads (MB/s): 6873.91 Writes(MB/s): 1810.86	Mem Ch 1: Reads (MB/s): 7411.11 Writes(MB/s): 1957.73
Mem Ch 2: Reads (MB/s): 6866.77 Writes(MB/s): 1804.38	Mem Ch 2: Reads (MB/s): 7403.39 Writes(MB/s): 1951.42
Mem Ch 3: Reads (MB/s): 6867.47 Writes(MB/s): 1805.53	Mem Ch 3: Reads (MB/s): 7403.66 Writes(MB/s): 1950.95
NODE0 Mem Read (MB/s) : 27478.96 NODE0 Mem Write (MB/s): 7225.79 NODE0 P. Write (T/s) : 214810 NODE0 Memory (MB/s) : 34704.75	NODE1 Mem Read (MB/s) : 29624.51 NODE1 Mem Write (MB/s): 7811.36 NODE1 P. Write (T/s) : 238294 NODE1 Memory (MB/s) : 37435.87
<ul style="list-style-type: none"> • System Read Throughput(MB/s): 57103.47 • System Write Throughput(MB/s): 15037.15 • System Memory Throughput(MB/s): 72140.62 	

11.11.3.2 pcm-memory.x in KVM4NFV:

pcm-memory is a part of KVM4NFV in D release.pcm-memory.x will be executed with delay of 60 seconds before starting every testcase to monitor the memory traffic intensity which was handled in collect_MBWInfo function .The memory bandwidth information will be collected into the logs through the testcase updating every 60 seconds.

Pre-requisites:

- 1.Check for the processors supported by PCM .Latest pcm utility version (2.11)support Intel® Xeon® E5 v4 processor family.
- 2.Disabling NMI Watch Dog
- 3.Installing MSR registers

Memory Bandwidth logs for KVM4NFV can be found [here](http://artifacts.opnfv.org/kvmfornfv.html):

```
http://artifacts.opnfv.org/kvmfornfv.html
```

Details of the function implemented:

In install_Pcm function, it handles the installation of pcm utility and the required prerequisites for pcm-memory.x tool to execute.

```
$ git clone https://github.com/opcm/pcm
$ cd pcm
$ make
```

In collect_MBWInfo Function,the below command is executed on the node which was collected to the logs with the timestamp and testType.The function will be called at the begining of each testcase and signal will be passed to terminate the pcm-memory process which was executing throughout the cyclic testcase.

```
$ pcm-memory.x 60 &>/root/MBWInfo/MBWInfo_${testType}_${timeStamp}

where,
${testType} = verify (or) daily
```

11.11.4 Future Scope

PCM information will be added to cyclicttest of kvm4nfv in yardstick.

11.12 Low Latency Tunning Suggestion

The correct configuration is critical for improving the NFV performance/latency. Even working on the same codebase, configurations can cause wildly different performance/latency results.

There are many combinations of configurations, from hardware configuration to Operating System configuration and application level configuration. And there is no one simple configuration that works for every case. To tune a specific scenario, it's important to know the behaviors of different configurations and their impact.

11.12.1 Platform Configuration

Some hardware features can be configured through firmware interface (like BIOS) but others may not be configurable (e.g. SMI on most platforms).

- **Power management:** Most power management related features save power at the expense of latency. These features include: Intel® Turbo Boost Technology, Enhanced Intel® SpeedStep, Processor C state and P state. Normally they should be disabled but, depending on the real-time application design and latency requirements, there might be some features that can be enabled if the impact on deterministic execution of the workload is small.
- **Hyper-Threading:** The logic cores that share resource with other logic cores can introduce latency so the recommendation is to disable this feature for realtime use cases.
- **Legacy USB Support/Port 60/64 Emulation:** These features involve some emulation in firmware and can introduce random latency. It is recommended that they are disabled.
- **SMI (System Management Interrupt):** SMI runs outside of the kernel code and can potentially cause latency. It is a pity there is no simple way to disable it. Some vendors may provide related switches in BIOS but most machines do not have this capability.

11.12.2 Operating System Configuration

- **CPU isolation:** To achieve deterministic latency, dedicated CPUs should be allocated for realtime application. This can be achieved by isolating cpus from kernel scheduler. Please refer to <http://lxr.free-electrons.com/source/Documentation/kernel-parameters.txt#L1608> for more information.
- **Memory allocation:** Memory should be reserved for realtime applications and usually hugepage should be used to reduce page faults/TLB misses.
- **IRQ affinity:** All the non-realtime IRQs should be affinity to non realtime CPUs to reduce the impact on realtime CPUs. Some OS distributions contain an irqbalance daemon which balances the IRQs among all the cores dynamically. It should be disabled as well.
- **Device assignment for VM:** If a device is used in a VM, then device passthrough is desirable. In this case, the IOMMU should be enabled.
- **Tickless:** Frequent clock ticks cause latency. CONFIG_NOHZ_FULL should be enabled in the linux kernel. With CONFIG_NOHZ_FULL, the physical CPU will trigger many fewer clock tick interrupts (currently, 1 tick per second). This can reduce latency because each host timer interrupt triggers a VM exit from guest to host which causes performance/latency impacts.

- **TSC:** Mark TSC clock source as reliable. A TSC clock source that seems to be unreliable causes the kernel to continuously enable the clock source watchdog to check if TSC frequency is still correct. On recent Intel platforms with Constant TSC/Invariant TSC/Synchronized TSC, the TSC is reliable so the watchdog is useless but cause latency.
- **Idle:** The poll option forces a polling idle loop that can slightly improve the performance of waking up an idle CPU.
- **RCU_NOCB:** RCU is a kernel synchronization mechanism. Refer to <http://lxr.free-electrons.com/source/Documentation/RCU/whatisRCU.txt> for more information. With RCU_NOCB, the impact from RCU to the VNF will be reduced.
- **Disable the RT throttling:** RT Throttling is a Linux kernel mechanism that occurs when a process or thread uses 100% of the core, leaving no resources for the Linux scheduler to execute the kernel/housekeeping tasks. RT Throttling increases the latency so should be disabled.
- **NUMA configuration:** To achieve the best latency. CPU/Memory and device allocated for realtime application/VM should be in the same NUMA node.

CHAPTER 12

KVM4NFV Releasenotes
