OPNFV User Guide

Release brahmaputra.1.0 (2b76a9c)

OPNFV

August 22, 2016

CONTENTS

CHAPTER

ONE

COLORADO 1.0

1.1 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

1.2 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 CICD scenario consists of Continuous Integration builds, deployments and testing combinations of virtual infrastructure components.

1.2.1 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.

1.2.2 General usage guidelines

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

1.3 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 Colorado. 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.

1.4 Using Colorado Features

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

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

1.5 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.

1.5.1 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:

CPU Features

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

CPU Topology

NUMA topology is also important for latency reduction.

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.

1.5.2 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.

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.

Run-time Environment Setup

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

1.6 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.

1.6.1 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.

1.6.2 Optimizations

1. 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.
2. 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%.
3. 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.

1.6.3 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) QEMU parameters: :: ${qemu} -smp ${guest_cpus} -monitor unix:${qmp_sock},server,nowait -daemonize -cpu host,migratable=off,+invtsc,+tsc-deadline,pmu=off -realtime mlock=on -mem-prealloc -enable-kvm -m 1G -mem-path /mnt/hugetlbfs-1g -drive file=/root/minimal-centos1.qcow2,cache=none,aio=threads -netdev user,id=guest0,hostfwd=tcp:5555-:22 -device virtio-net-pci,netdev=guest0 -nographic -serial /dev/null -parallel /dev/null

Network connection

1.6.4 Test Result

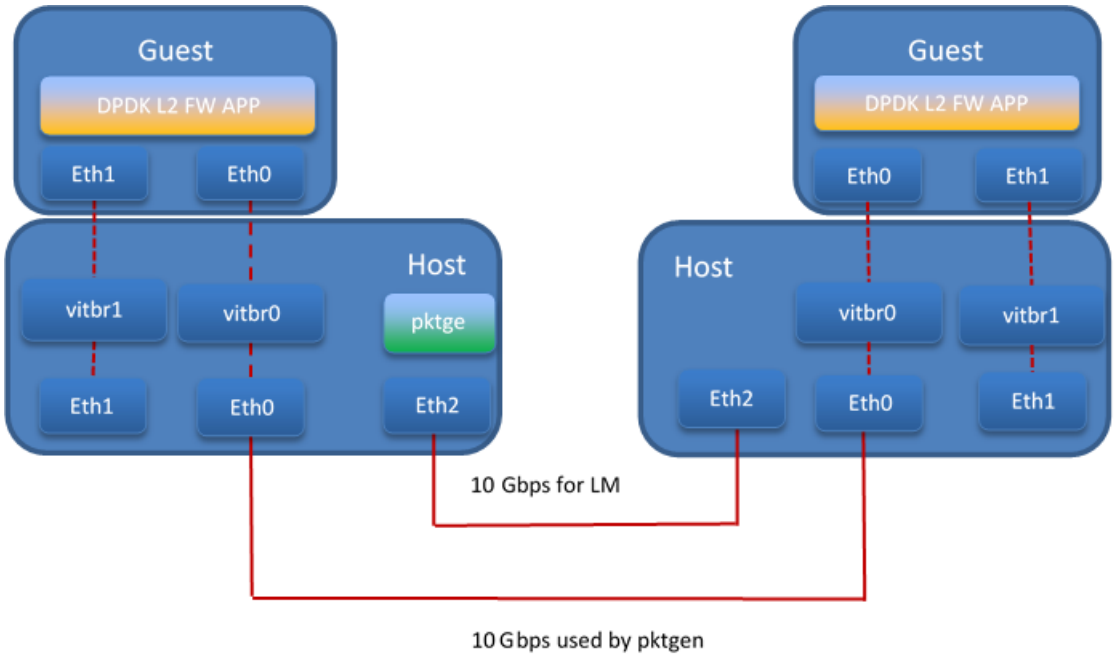
The down time is set to 10ms when doing the test. We use pktgen to send the packages to guest, the package size is 64 bytes, and the line rate is 2013 Mbps.

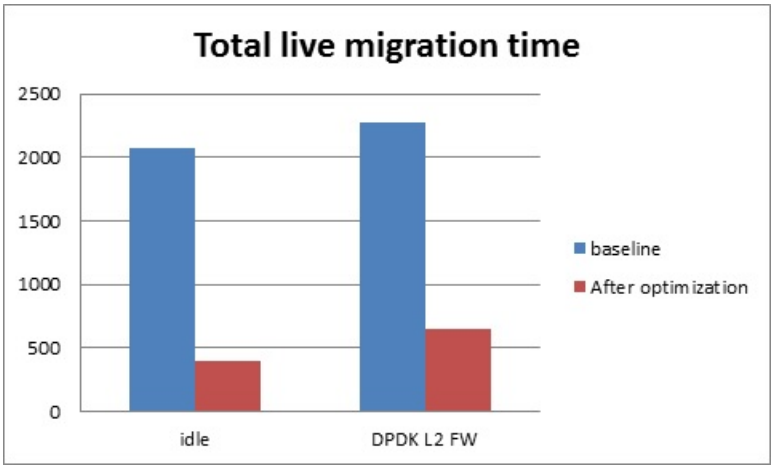
1. Total live migration time

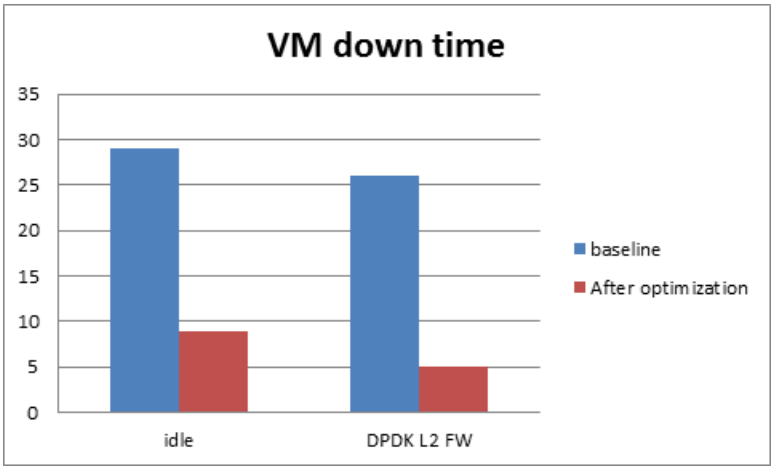
   The total live migration time before and after optimization is shown in the chart below. For an idle guest, we can reduce the total live migration time from 2070ms to 401ms. For a guest running the DPDK L2 forwarding workload, the total live migration time is reduced from 2271ms to 654ms.

2. 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.


Guest
DPDK L2 FW APP
Eth1
Eth0
Host
vitbr1
vitbr0
pktge
Eth1
Eth0
Eth2
Guest
DPDK L2 FW APP
Eth0
Eth1
Host
vitbr0
vitbr1
Eth2
Eth0
Eth1
10 Gbps for LM
10 Gbps used by pktgen


Total live migration time
2500
2000
1500
1000
500
0
idle
DPDK L2 FW
baseline
After optimization


VM down time
35
30
25
20
15
10
5
0
idle
DPDK L2 FW
baseline
After optimization

1.7 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.

1.7.1 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 expensive 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.

1.7.2 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 shoud be reserved for realtime applications and usually hugepage should be used to reduce page fauts/TLB misses.
- **IRQ affinity:** All the non-realtime IRQs should be affinitized 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.