OPNFV

KVM4NFV project
Release WW-2015-52 (10565da)

OPNFV

January 21, 2016

Welcome to KVM4NFV project!

Contents:

ONE

KVM4NFV 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, and we need to make collaborative efforts toward enabling the NFV features.

The KVM4NFV project focuses on the KVM hypervisor to enhance it for NFV, by looking at the following areas

- **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

hile these items require software development and/or specific hardware features there are also some adjustments that need to be made to system configuration information, like hardware, BIOS, OS, etc.

SETUP GUIDES

2.1 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.

2.1.1 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.

Below is the CPU information on the baseline test platform.

```
processor       : 35
vendor_id       : GenuineIntel
cpu family      : 6
model           : 63
model name      : Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz
stepping        : 2
microcode       : 0x2d
cpu MHz         : 2294.795
cache size      : 46080 KB
physical id     : 1
siblings        : 18
core id         : 27
cpu cores       : 18
apicid          : 118
initial apicid  : 118
fpu             : yes
fpu_exception   : yes
cpuid level     : 15
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
                  mca cmov pat pse36 clflush dts acpi mmx fxsr sse
                  sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm
                  constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc
                  aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2
```

```
                    ssse3 fma cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt
                    tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm arat epb
                    pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase
                    tsc_adjust bmi1 avx2 smep bmi2 erms invpcid cqm xsaveopt cqm_llc
                    cqm_occup_llcbugs
bogomips        : 4595.54
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

CPU Topology

NUMA topology is also important for latency reducation.

Below is the CPU topology on the baseline test platform.

```
[nfv@otcnfv02 ~]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                36
On-line CPU(s) list:   0-35
Thread(s) per core:    1
Core(s) per socket:    18
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 63
Model name:            Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz
Stepping:              2
CPU MHz:               2294.795
BogoMIPS:              4595.54
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              46080K
NUMA node0 CPU(s):     0-17
NUMA node1 CPU(s):     18-35
```

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.

```
CPU Power and Performance <Performance>
CPU C-State <Disabled>
C1E Autopromote <Disabled>
Processor C3 <Disabled>
Processor C6 <Disabled>
Select Memory RAS <Maximum Performance>
NUMA Optimized <Enabled>
Cluster-on-Die <Disabled>
Patrol Scrub <Disabled>
```

```
Demand Scrub <Disabled>
Correctable Error <10>
Intel(R) Hyper-Threading <Disabled>
Active Processor Cores <All>
Execute Disable Bit <Enabled>
Intel(R) Virtualization Technology <Enabled>
Intel(R) TXT <Disabled>
Enhanced Error Containment Mode <Disabled>
USB Controller <Enabled>
USB 3.0 Controller <Auto>
Legacy USB Support <Disabled>
Port 60/64 Emulation <Disabled>
```

2.1.2 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 etailed 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_huge
```

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 Low Latency Tunning Suggestion 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 Low Latency Tunning Suggestion for more explanation.

2.2 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.

2.2.1 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 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.

2.2.2 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 deamon 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.

2.3 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.

2.3.1 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.

2.3.2 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. Testingt 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.

2.3.3 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:

```
/root/qemu.git/x86_64-softmmu/qemu-system-x86_64-enable-kvm -cpu host -smp 4 -device virtio-net-pci,r
```

Network connection
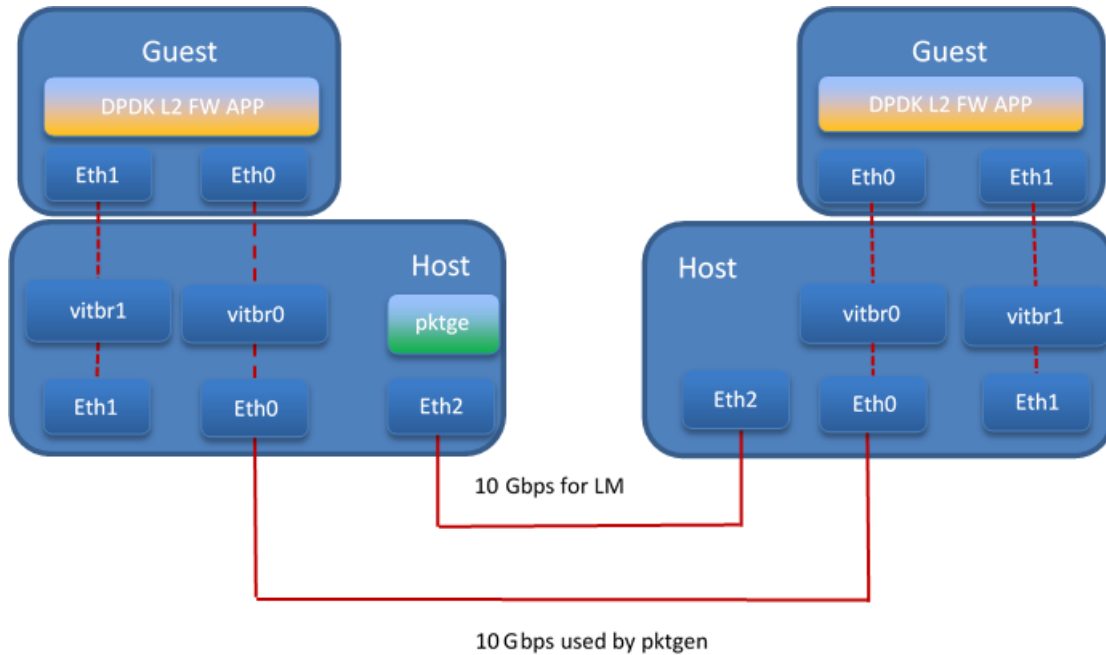
2.3.4 Test Result

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.
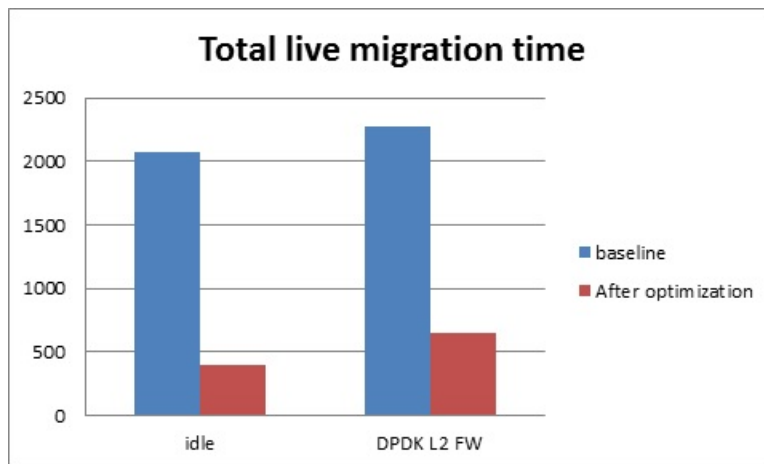
2. VM downtime

   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.