Real-Time Performance Tuning Best Practice Guidelines for KVM-Based Virtual Machines

White Paper

January 2022

Document Number: 710099-1.0

Contents

1.0 **Introduction** 5
  1.1 Terminology 5
  1.2 Reference Documents 6

2.0 **Real-Time Performance Tuning for Virtual Machines** 7
  2.1 Background 7
  2.2 Testbed 7
  2.3 Real-Time Performance Tuning 7
    2.3.1 BIOS Configurations 7
    2.3.2 Kernel Configurations 10
    2.3.3 Host Configurations 11
    2.3.4 QEMU-KVM Configurations for Real-Time VM 13
    2.3.5 Configurations for guest OS in the real-time VM 13
  2.4 Performance Evaluation 14
    2.4.1 Test 14

3.0 **Conclusion** 15

Figures

Figure 1. Cache Allocation Strategy for CPU and GPU 12

Tables

Table 1. Terminology 5
Table 2. Reference Documents 6
Table 3. Testbed Configuration 7
Table 4. BIOS Configurations 8
Table 5. Kernel Command Line Parameters 10

Revision History

Date	Revision	Description
January 2022	1.0	Initial release.

§

1.0 Introduction

In this document, we share our experiences of real-time performance tuning with applications running in a KVM-based virtual machine on an IA platform. This document includes our recommendations for setting up BIOS, kernel, QEMU-KVM, and the applications themselves. Customers looking to fine tune the real-time performance for their own applications might benefit from this experience sharing.

1.1 Terminology

Table 1. Terminology

Term	Description
BIOS	Basic input/output system.
CAT	Cache allocation technology.
Cycle time	The amount of time allotted to complete the cyclic workload.
IA	Intel Architecture.
IRQ	Interrupt request.
Jitter	The difference between the start times (relative to the request times) of two or more instances of a periodic task.
Latency	The duration of time between two events.
NMI	Nonmaskable interrupt.
NVME	Nonvolatile memory express.
RCU	Read copy update.
SMP	Symmetric multiprocessing.
TLB	Translation lookaside buffer.
TSC	Time Stamp Counter.
VM	Virtual machine.
Workload	An application that performs some useful computational work, including (perhaps) receiving input, performing computation, and generating an output.

1.2 Reference Documents

Log in to the Resource and Design Center (rdc.intel.com) to search for and download the document numbers listed in the following table. Contact your Intel field representative for access.

Note: Third-party links are provided as a reference only. Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Table 2. Reference Documents

Document	Document No./Location
[1] Edge Control for Industrial	https://software.intel.com/content/www/us/en/develop/topics/iot/edge-solutions/controls-for-industrial.html
[2] BIOS	https://en.wikipedia.org/wiki/BIOS
[3] chrt	https://www.cyberciti.biz/faq/howto-set-real-time-scheduling-priority-process/
[4] taskset	https://man7.org/linux/man-pages/man1/taskset.1.html
[5] QEMU options	https://manpages.debian.org/jessie/qemu-system-x86/qemu-system-x86_64.1.en.html
[6] Kernel command-line parameters	https://www.kernel.org/doc/html/latest/admin-guide/kernel-parameters.html?highlight=kernel%20parameters
[7] Intel® Resource Director Technology (Intel® RDT) Framework	https://www.intel.com/content/www/us/en/architecture-and-technology/resource-director-technology.html

§

2.0 Real-Time Performance Tuning for Virtual Machines

2.1 Background

Real-time systems are used in many industries. Basically, a real-time system is a computing system that must react within time constraints to events in the environment. Software running on a real-time system must be able to guarantee a response within a specified timeframe. While it has become increasingly popular to run applications in containers, it is still common to find applications running in VM, especially in the manufacturing sector. Consequently, there is a need to study how to tune the real-time performance of VM.

2.2 Testbed

To ensure repeatable results, we chose an industrial PC (IPC) based on Intel Whiskey Lake-U platform as the testbed. Tabe 3 outlines the specifications of the testbed.

Table 3. Testbed Configuration

CPU	i7-8665U-quad core, 1.9 GHz
Memory	2x SO-DIMM DDR4-2400/1233 MHZ, 64GB
GPU	Integrated GPU
BIOS	Version WL37R107
OS	Ubuntu 18.04

2.3 Real-Time Performance Tuning

A computing system consists of quite a few software components, including BIOS, kernel, hypervisor, system services, applications, etc. Any one of these components can impact the real-time performance of the computing system. In this chapter, we will introduce how to tune these software components to achieve a good real-time performance for a VM.

2.3.1 BIOS Configurations

BIOS is firmware used to perform hardware initialization during the booting process. Some BIOS settings can have a significant impact on the real-time performance of the system [2]. The configurations in Table 4 are tuned for real-time performance.

Table 4. BIOS Configurations

Setting Name	Setting Location on Example System	Description	Suggested Setting
Hyper Threading	Intel Advanced Menu > CPU Configuration	Intel's proprietary simultaneous multithreading implementation to improve parallelization performance on x86 CPUs	Disabled
Intel VMX	Intel Advanced Menu > CPU Configuration	Intel Virtual Machine Extensions	Enabled
Intel SpeedStep®	Intel Advanced Menu > Power & Performance > CPU - Power Management Control	A dynamic frequency scaling technology	Disabled
Intel® Speed Shift Technology	Intel Advanced Menu > Power & Performance > CPU - Power Management Control	Increases performance and efficiency by managing hardware P states in a better way	Disabled
C States	Intel Advanced Menu > Power & Performance > CPU - Power Management Control	The level of activity in which a core resides	Disabled
RC6	Intel Advanced Menu > Power & Performance > GT - Power Management	A deep sleep state, Graphics Render Standby (RC6)	Disabled
GT freq	Intel Advanced Menu > Power & Performance > GT - Power Management	The integrated graphics frequency of Intel CPUs	Lowest (usually the lowest is 100MHz)
SA GV	Intel Advanced Menu > Memory Configuration	System Agent Enhanced Intel SpeedStep®	Fixed High
VT-d	Intel Advanced Menu > System Agent Configuration	Intel® Virtualization Technology (Intel® VT) for Directed I/O (Intel® VT-d)	Enabled

Setting Name	Setting Location on Example System	Description	Suggested Setting
Gfx Low Power Mode	Intel Advanced Menu > System Agent Configuration > Graphics Configuration	Graphics low-power state mode	Disabled
DMI spine clock gating	Intel Advanced Menu > System Agent Configuration > DMI/OPI Configuration	A desktop management interface power-saving function	Disabled
PCH Cross Throttling	Intel Advanced Menu > PCH-IO Configuration	A Platform Control Hub thermal management feature	Disabled
Legacy IO Low Latency	Intel Advanced Menu > PCH-IO Configuration > PCI Express Configuration	Enable or disable low latency of legacy IO	Enabled
PCI Express Clock Gating	Intel Advanced Menu > PCH-IO Configuration > PCI Express Configuration	PCI Express clock distribution network power-saving function	Disabled
Delay Enable DMI ASPM	Intel Advanced Menu > PCH-IO Configuration > PCI Express Configuration	Delay of Desktop Management Interface Active State Power Management	Disabled
DMI Link ASPM	Intel Advanced Menu > PCH-IO Configuration > PCI Express Configuration	Delay of Desktop Management Interface link Active State Power Management control (Before you Enable Legacy IO Low Latency, disable the DMI Link ASPM control)	Disabled
Aggressive LPM Support	Intel Advanced Menu > PCH-IO Configuration > SATA And RST Configuration	Low Power Mode support	Disabled

Setting Name	Setting Location on Example System	Description	Suggested Setting
USB Periodic SMI	Intel Advanced Menu > LEGACY USB Configuration	System-management interrupt	Disabled
ACPI S3 Support	Intel Advanced Menu > ACPI Settings	Advanced Configuration and Power Interface sleep state3	Disabled
Native ASPM	Intel Advanced Menu > ACPI Settings	Active State Power Management	Disabled

2.3.2 Kernel Configurations

The native kernel contained in Ubuntu 18.04 release is not a real-time kernel. To get a better real-time performance, we replaced it with the kernel image contained in Intel ECI 1.0 release [1], which is Linux 4.19.59 with pre-empt patches.

The kernel command-line parameters [6] are shown below. Some of them are essential for real-time performance.

Kernel parameters: "quiet splash default_hugepagesz=1G hugepagesz=1G hugepages=8 intel_iommu=on rdt=l3cat nmi_watchdog=0 idle=poll clocksource=tsc tsc=reliable noht audit=0 skew_tick=1 intel_pstate=disable intel.max_cstate=0 intel_idle.max_cstate=0 processor.max_cstate=0 processor_idle.max_cstate=0 nosoftlockup nohz=on no_timer_check nospectre_v2 spectre_v2_user=off kvm.kvmclock_periodic_sync=N kvm_intel.ple_gap=0 irqaffinity=0 nohz_full=2-3 rcu_nocbs=2-3 isolcpus=1-3"

Table 5. Kernel Command Line Parameters

Configuration	Comments
default_hugepagesz=1G hugepagesz=1G hugepages=5	To reduce TLB miss and avoid mem swap for VMs
intel_iommu=on	For dev passthrough
rdt=l3cat	Try to enable l3cat that kernel does not support it on some CPU versions by default.
idle=poll	Avoid hlt/mwait instructions; for host, avoid CPU entering power saving state; for VMs, avoid vmexist
nmi_watchdog=0	It is used for hard lockup detection; turn off it to avoid NMI
clocksource=tsc tsc=reliable	Avoid TSC correction at runtime
skew_tick=1	Offset the periodic timer tick per CPU to mitigate lock contention

Configuration	Comments
intel_pstate=disable intel.max_cstate=0 intel_idle.max_cstate=0 processor.max_cstate=0 processor_idle.max_cstate=0	Disable power saving
nosoftlockup	Thread with SCHED_FIFO policy might occupy CPU for long time
no_timer_check	Avoid checking broken timer IRQ
nospectre_v2	Avoid this security patch impact
kvm_intel.ple_gap=0	VT-x provides "Pause Loop Exit" to detect busy acquiring lock CPU. The parameter is the upper bound on the amount of time between two successive executions of PAUSE in a loop. It needs to disable it to avoid the vmexit for "atomic operations" inside guest OS.
kvm.kvmclock_periodic_sync=N	Don't sync clock; avoid time jitter
irqaffinity=0	Set the default IRQ affinity mask. For real-time, it always binds IRQs to non-isolated CPUs.
nohz_full=2-3	The specified list of CPUs whose tick will be stopped whenever possible. For real-time, it can reduce the tick IRQ impact.
rcu_nocbs=2-3	Not to run RCU callback on real-time CPUs
isolcpus=1-3	Isolate CPUs from the general SMP balancing and scheduling algorithms. It is used to reserve CPUs for real-time workloads.

2.3.3 Host Configurations

Besides the BIOS and the kernel, we made some additional configurations on the host for better real-time performance.

- Set "CPUAffinity=0" in "/etc/systemd/system.conf".

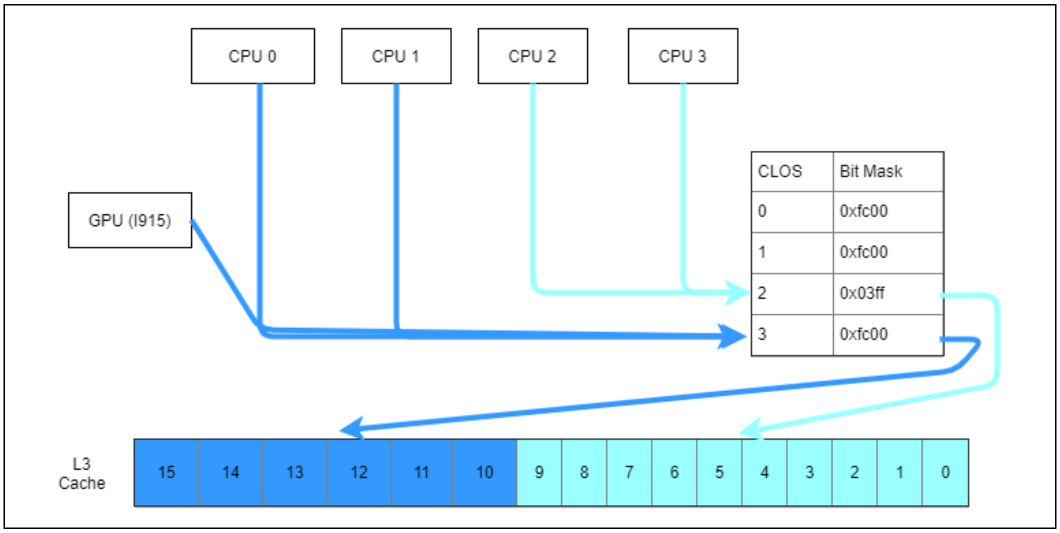
  It is to set the CPU affinity of all "systemd" services to CPU 0.

- Cache isolation for CPU and GPU

  Cache allocation technology (CAT) helps address shared resource concerns by providing software control of where data is allocated into the last-level cache (LLC), enabling isolation and prioritization of key applications [9].

  Cache allocation can be performed through Linux resctrl interface [7][8]. In the setup, CPU 0 and 1 are used for the system services and non-realtime workloads. We chose the cache allocation strategy as shown in Figure 1.

Figure 1. Cache Allocation Strategy for CPU and GPU



- Set the CPU affinity of IRQ thread to CPU 0

  The CPU affinity of IRQ threads is not controlled by the irqaffinity parameter of the kernel command line. We used a script to manually set them.

  Below is an example for the IRQ thread of NVME. With below commands, the IRQ thread can be moved to CPU 0 to avoid impacting real-time workloads running on CPU 2&3.

  ps -e | grep 'irq/.*nvme'

  taskset -a -p -c 0 <pid>

- Set the device driver work queue to CPU 0

  The work queues of the device drivers are not controlled by the isolcpus parameter of the kernel command line. We use below commands to move them to CPU 0.

  "echo 1 > /sys/devices/virtual/workqueue/cpumask"

  "echo 1 > /sys/bus/workqueue/devices/writeback/cpumask"

- Disable machine check

  By default, the Linux kernel periodically scans hardware for reported errors. Disabling this check improves real-time performance of workloads by preventing the Linux kernel from interrupting the running tasks.

  "echo 0 > /sys/devices/system/machinecheck/machinecheck0/check_interval".

- Stop a few services
  - irqbalance.service: irqbalance is a daemon that distributes interrupts over among the processors and cores in a computer system.
  - thermald.service: thermald is a daemon that monitors and controls temperature of computer systems. Once the system temperature reaches a

  - certain threshold, the Linux daemon activates various cooling methods to try to cool the system.
  - wpa_supplicant.service: wpa_supplicant is a daemon that manages wireless interfaces.

2.3.4 QEMU-KVM Configurations for Real-Time VM

Here are some configurations for QEMU-KVM:

- Tune lapic timer advance.

  In our setup, we chose the value of 7500 by the command

  "echo 7500 > /sys/module/kvm/parameters/lapic_timer_advance_ns".

For different chips, the value could be different.

- Some QEMU options [5] for real-time VM
  - -realtime mlock=on

    Lock mem to avoid mem swap and lazy allocation
  - -balloon none

    Avoid to free and return memory to host OS
  - -mem-prealloc -mem-path /dev/hugepages/;

    Use hugepage to reduce tlb missing
- In the setup, we set the CPU affinity of the two QEMU CPU threads of the real-time VM to CPU 2 and 3 [4] respectively which was already isolated during boot stage. Furthermore, set the scheduling policy as "FIFO" [3] and the highest priority for the two threads.
- Passthrough PCI devices into the VM if needed. In our test, we didn't passthrough any PCI device.

2.3.5 Configurations for guest OS in the real-time VM

We use the same OS and kernel image as the host for the real-time VM. The configurations for the guest OS are very similar with them done on the host.

- Ubuntu 18.04 is installed with minimal installation.
- Use two commands "sudo systemctl enable multi-user.target --force" and "sudo systemctl set-default multi-user.target" to boot into a text-based environment.
- Kernel parameters are similar, except nohz_full, rcu_nocbs and isolcpus. Since there are only two CPUs for the VM, we set nohz_full=1, rcu_nocbs=1 and isolcpus=1.

- Except “Cache isolation for CPU and GPU” and “Disable machine check”, all other host configurations are done on the guest OS.
- Tune sched_rt_runtime_us

  sched_rt_runtime_us is the time allocated for real-time tasks. Default value is 950000 (0.95s). If set to -1, real-time tasks will occupy all CPU time if they are in active, which will hang the whole system. In the setup, we chose the value of 100000 by the command

  “echo 100000 > /proc/sys/kernel/sched_rt_runtime_us”.

2.4 Performance Evaluation

2.4.1 Test

To ensure the reliability of our test result, we launched two heavy workload programs on the platform before testing the real-time performance.

The first is a KVM-based VM, which is running on CPU 0 and CPU 1. The OS is Window 10, and the GPU hardware is loaded down with a GUI program.

The second is a stress test program running on the host. The command line is “taskset -c 0 stress -q -i 4 -c 4 -d 4 --hdd-bytes 20M -q -m 4 --vm-bytes 10Ms”. The CPU affinity of the process is set to CPU 0. This stress test program imposes heavy workload on CPU, disk, and memory.

Then we launch the program “cyclictest” with the command “cyclictest -p99 -m -H 20 -t 1 -a 1” in the real-time VM we tuned. The CPU affinity of the process is set to vCPU 1.

The whole test lasted more than 48 hours. Finally, our latency measures scored in the range of tens of microseconds.

§

3.0 Conclusion

Our test results that, when combined, the tuning methods described here result in significantly enhanced real-time performance for KVM-based VM for our testbed platform. Even when co-run with other heavy workloads, real-time VM still continues to achieve reliably low latency running in the range of just tens of microseconds.

§