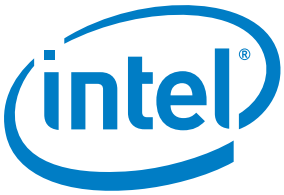


Using Intel® Virtualization Technology (Intel® VT) with Intel® QuickAssist Technology

Application Note

December 2018

Document Number: 330689-008

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or visit www.intel.com/design/literature.htm. No computer system can be absolutely secure.

Intel, Intel Atom, Xeon, and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2018, Intel Corporation. All rights reserved.

Contents

1 Introduction 6
1.1 About This Document 6
1.2 Terminology 6
1.3 About the Software 7
1.3.1 Features 7
1.3.2 Limitations 7
1.4 Documentation 7
1.4.1 Where to Find Current Software and Documentation 7
1.4.2 Product Documentation 7
1.4.3 Documentation Conventions 8
1.5 Software Requirements 8
1.6 Supported Intel® QuickAssist Technology (QAT) Endpoints and Their Device IDs 8
2 Using Intel® QAT Software with KVM 9
2.1 Updating the BIOS Settings 9
2.2 Installing and Configuring the Host Operating System 9
2.3 Installing the Guest OS Image 10
2.4 Installing and Configuring Intel® QuickAssist Technology Software 11
2.4.1 Installing Intel® QuickAssist Technology Software on Host 12
2.4.2 Verifying SR-IOV on the Host 12
2.4.3 Pass-through the PCI Device 13
2.4.4 Installing Intel® QuickAssist Technology Software on the Guest 16
2.5 Running Acceleration Services Simultaneously in Host and Guest 16
2.6 Enabling Virtual Functions in QAT 17
Appendix A FAQ 18
A.1 Q: How can I pass through the QAT PF to a guest? 18

Figures

Figure 1. Virtual Machine Manager 13
Figure 2. View Virtual Machine Details 14
Figure 3. Add New Virtual Hardware 15

Tables

Table 1. Terminology 6
Table 2. Related Documents 7
Table 3. Supported Intel® QAT Endpoints and Their Device IDs 8

Revision History

Document Number	Revision Number	Description	Revision Date
330689	008	• Updated information on enabling virtual functions in QAT	December 2018
330689	007	• Added device ID for Intel® Xeon® processor D family. • Made updates to focus on the QAT1.7 hardware and software	September 2018
330689	006	• Removed section on using QEMU* KVM command line interface • Added device ID for Intel® C62x Chipset and Intel Atom® C3000 Processor Product Family	August 2017
330689	005	Updated: • Section 1.3.2, Limitations • Section 1.5, Software Requirements • Section 2.2, Installing and Configuring the Host Operating System • Section 2.3, Installing the Guest OS Image • Section 2.4.1, Using the libvirt* Virtual Machine Manager GUI • Section 2.4.3, Pass-through the PCI Device • Section 2.4.2.1, Installing Updated QEMU* KVM • Section 2.4.2.2, Pass-through the PCI Device • Section 2.4.2.4, Starting the Guest • Section 2.5, Running Acceleration Services Simultaneously in Host and Guest • Appendix A, FAQ	July 2016
330689	004	Updated: • Section 1.5, Software Requirements • Section 2.4.1, Installing Intel® QuickAssist Technology Software on Host • Section 2.5, Running Acceleration Services Simultaneously in Host and Guest	February 2015
330689	003	Updated Section 2.5, Running Acceleration Services Simultaneously in Host and Guest. Added Appendix A, FAQ	November 2014
330689	002	Updated Section 2.5, Running Acceleration Services Simultaneously in Host and Guest.	September 2014
330689	001	First "public" version of the document. Based on "Intel confidential" document number 476488-1.4 with the revision history of that document retained for reference purposes. Updated Section 1.4.1, Where to Find Current Software and Documentation.	July 2014

Document Number	Revision Number	Description	Revision Date
		Removed Fedora 14 information from Section 1.5, Software Requirements and Section 2.4.1, "Using the libvirt* Virtual Machine Manager GUI" on page 10. Added new step at the end of Section 2.2, "Installing and Configuring the Host Operating System" on page 8. Updated Section 2.5, Running Acceleration Services Simultaneously in Host and Guest.	
476488	1.4	Updates include: Modified step 8 in Section2.3, Installing the Guest OS Image. Added Section 2.5, Running Acceleration Services Simultaneously in Host and Guest.	March 2014
476488	1.3	Updates to make applicable to multiple platforms that use Intel® QuickAssist Technology.	June 2013
476488	1.2	Added new FAQ items, deleted outdated FAQ items.	February 2013
476488	1.1	Added Limitation.	October 2012
476488	1.0	Initial release of this document.	September 2012

§

1 Introduction

This document discusses the following topics related to using Intel® Virtualization Technology (Intel® VT) with the Intel® QuickAssist Technology Software:

- Features and limitations
- Build and installation

1.1 About This Document

Users of this document are expected to be familiar with virtualization technologies.

In this document, for convenience:

- *Software package* is used as a generic term for the Intel® QuickAssist Technology Software package.
- *Acceleration drivers* is used as a generic term for the software that allows the QuickAssist Software Library APIs to access the Intel® QuickAssist Accelerator(s) integrated in the Intel® QuickAssist Technology.

1.2 Terminology

Table 1. Terminology

Term	Description
CLI	Command Line Interface
GigE	Gigabit Ethernet
GUI	Graphical User Interface
Intel® QAT	Intel® QuickAssist Technology Software
Intel® VT	Intel® Virtualization Technology
IOMMU	Input-Output Memory Management Unit
KVM	Kernel-based Virtual Machine
PCH	Platform Controller Hub
SR-IOV	Single-root Input/Output Virtualization
PF	Physical Function
VF	Virtual Function

1.3 About the Software

This section lists the features and limitations.

1.3.1 Features

- PCI pass-through with Kernel-based Virtual Machine (KVM)
- SR-IOV with KVM

1.3.2 Limitations

- SR-IOV may not work on GNU*/Linux* kernel versions older than 2.6.38.
- KVM limitation: the maximum number of Virtual Functions that can be mapped to a single VM is specific to qemu-kvm version.

1.4 Documentation

1.4.1 Where to Find Current Software and Documentation

Associated software and collateral can be found on the open source website:
https://01.org/intel-quickassist-technology

Table 1 includes a list of related documentation.

1.4.2 Product Documentation

Documentation includes:

- Using Intel® Virtualization Technology (Intel® VT) with Intel® QuickAssist Technology Application Note (this document)
- Additional related documents listed in Table 2, which may be accessed as described in Section 1.3.1.

Table 2. Related Documents

Document Name	Number
Intel® QuickAssist Technology Software Release Notes	330683
Intel® QuickAssist Technology API Programmer's Guide	330684
Intel® QuickAssist Technology Cryptographic API Reference Manual	330685
Intel® QuickAssist Technology Data Compression API Reference Manual	330686
Intel® QuickAssist Technology for Linux* Release Notes	330683
Intel® QuickAssist Technology for Linux* Getting Started Guide	336212
Intel® Communications Chipset 89xx Series Datasheet	327879

Intel® QuickAssist Technology Software for Linux* - Getting Started Guide - HW version 1.7	336212
Intel® QuickAssist Technology Software for Linux* - Release Notes - HW version 1.7	336211
Intel® QuickAssist Technology Software for Linux* - Programmer's Guide - HW version 1.7	336210
Intel® QuickAssist Technology Driver for Linux* - HW version 1.7	

Note: Sample configuration files are included with the software package.

1.4.3 Documentation Conventions

The following conventions are used in this manual:

- `Courier font` - code examples, command line entries, API names, parameters, filenames, directory paths, and executables
- **Bold text** - graphical user interface entries and buttons

1.5 Software Requirements

Software requirements will vary by the particular use case.

- Required: the Intel® QuickAssist Technology Software for Linux*

  Intel recommends using the same version of the QuickAssist driver on both host and guest OS. Consult your Intel representative if you have a requirement to use different versions of the driver.

These instructions were tested against the following Linux distribution:

- CentOS*

1.6 Supported Intel® QuickAssist Technology (QAT) Endpoints and Their Device IDs

Table 3. Supported Intel® QAT Endpoints and Their Device IDs

Intel® QAT Endpoint	Physical Function (PF) Device ID	VF Device ID
8925-8955	0435	0443
Intel® C62x Chipset	37c8	37c9
Intel Atom® C3000 Processor Product Family	19e2	19e3
Intel® Xeon® processor D family	6f54	6f55

2 Using Intel® QAT Software with KVM

The Intel® Virtualization Technology can use both Single-Root I/O Virtualization (SR-IOV) and PCI pass-through for the acceleration services. SR-IOV enables the creation of Virtual Functions from a single Intel® QuickAssist Technology acceleration device to support acceleration for multiple virtual machines. If you do not need to share a single PCH device with accelerator capabilities between multiple virtual machines, PCI pass-through is sufficient. The following sections describe the steps necessary to enable this functionality, with a focus on the SR-IOV use case.

2.1 Updating the BIOS Settings

Note: The BIOS settings for your system may differ from the following steps.

1. Power on the development board. Watch closely for the prompt to enter BIOS setup. Press **F2** when prompted.
2. Enable the VT-d parameter in BIOS. The option may be available under: **Advanced > System Agent (SA) Configuration > VT-d**
3. Enable the SR-IOV parameter in BIOS. The option may be available under: **Advanced > System Agent (SA) Configuration > SRIOV**

Note: Enabling the SR-IOV BIOS parameter is not required if you are not using SR-IOV.

4. Press **F4** to Save and Exit. The BIOS changes are saved and the system will boot.

2.2 Installing and Configuring the Host Operating System

1. Install the CentOS 7 64-bit version. If necessary, consult the *Getting Started Guide* section "Installing the OS on a Development Board" (refer to Table 2), taking note that this guide assumes one of those CentOS 7 64-bit versions as the host OS when SR-IOV is used.

Note: CentOS 7 requires the `intel_iommu=on` kernel boot parameter to use SR-IOV and VT-d functionality.

2. Install virtualization related packages using the following command (root privileges required):
```
# yum -y install @virtualization
```

Note: Alternatively, use `yum -y groupinstall Virtualization`.
This will install qemu-kvm qemu-img virt-manager libvirt libvirt-python python-virtinst libvirt-client virt-install virt-viewer and all of the dependencies that are needed.

3. If the libvirtd service is not running, start it by using the commands:
```
# chkconfig libvirtd on
# service libvirtd start
```
4. Verify SR-IOV hardware capabilities using the command:
```
# lspci -vnc 8086:<Device ID>
```

Refer to Section 1.5 for supported devices and their device IDs.

It should display one of the capabilities as:

```
Capabilities: [140] Single Root I/O Virtualization (SR-IOV)
```

5. Verify BIOS settings using the command:
```
# lsmod | grep kvm
```
```
kvm_intel     42122 0
kvm  257132 1 kvm_intel
```
6. Ensure that the system supports VT extensions:
```
# egrep '^flags.*(vmx|svm)' /proc/cpuinfo
```

Note: If nothing is printed out after executing the above command, then the system does not support VT extensions.

7. If kernel boot parameters changed, restart the system:
```
# shutdown -r now
```
8. Power on the system and proceed with the instructions in the following sections.
9. Once the system is restarted, check for DMAR and IOMMU messages, similar to the following:
```
# dmesg | grep -e DMAR -e IOMMU
```
```
[    0.000000] ACPI: DMAR 000000007b79c000 00080 (v01 INTEL INTEL ID
00000001 INTL 20091013)
[    0.000000] Intel-IOMMU: enabled
[    0.064454] dmar: IOMMU 0: reg_base_addr fbffc000 ver 1:0 cap
d2078c106f0466 ecap f020df
[   0.065560] IOAPIC id 8 under DRHD base  0xfbffc000 IOMMU 0 [
0.065919] IOAPIC id 9 under DRHD base  0xfbffc000 IOMMU 0 [ 2.168898]
DMAR: No ATSR found
[    2.169358] IOMMU 0 0xfbffc000: using Queued invalidation [
     2.169728] IOMMU: Setting RMRR:
[    2.170091] IOMMU: Setting identity map for device 0000:00:1d.0
[0x7a23f000 - 0x7a241fff]
[    2.170767] IOMMU: Prepare 0-16MiB unity mapping for LPC
[    2.171133] IOMMU: Setting identity map for device 0000:00:1f.0
[0x0 - 0xffffff]
```

Note: If the above command fails, a BIOS update or kernel reconfiguration may be required.

2.3 Installing the Guest OS Image

This section describes how to use the libvirt* Virtual Machine Manager GUI to create the guest OS installation.

Note: The instructions in this section use the Graphical User Interface (GUI) approach; information on using the command line interface (CLI) is available at: http://libvirt.org/virshcmdref.html

Note: Using the steps below, enter the root password when prompted.

1. Start the Virtual Machine Manager GUI by selecting it from the top main menu:

   Applications > System Tools > Virtual Machine Manager.
2. Open a connection to a Hypervisor by choosing **File > Add Connection**.
3. Choose **QEMU/KVM** for Hypervisor.
4. Make sure **Connect to remote host** is NOT checked.

5. Make sure **Autoconnect** is checked.
6. Click the **Connect** button.
7. After a connection is opened, select the **localhost (QEMU)** and right click to select **New**.

   The New VM window is displayed.

   a. **Create a new virtual machine (Step 1 of 5)**: Enter the **Name** for the Guest virtual machine, select **Local install media (ISO image or CDROM)**, and click the **Forward** button.

   b. **Create a new virtual machine (Step 2 of 5)**: Select **Use CDROM or DVD**, insert the OS installation CDROM/DVD into the CDROM/DVD drive, and make sure that the mounted CDROM appears in box [**Media Unknown (dev/sr0)**]. Select the OS type and version, and click the **Forward** button.

   c. **Create a new virtual machine (Step 3 of 5)**: Choose **Memory (RAM)** in MB and number of **CPUs** settings (assign a sufficient amount, but it should not affect the Host OS, e.g., for 4 GB RAM and 8 cores, allocate Guest OS < 2 GB RAM and 4 cores CPU). Click the **Forward** button.

   Note: Many platforms will show twice the actual number of cores due to simultaneous multithreading.

   d. **Create a new virtual machine (Step 4 of 5)**: Make sure **Enable storage for this virtual machine** is checked. Select **Create a disk image on the computer's hard drive** and specify a sufficient amount of hard drive space in GB (20 GB is recommended, and at least 18 GB may be required). Make sure **Allocate entire disk now** is checked. Click the **Forward** button.

   e. **Create a new virtual machine (Step 5 of 5)**: Review the information from Steps 1 through 4. Note the **Ready to begin installation of <Name>** and the **Storage** path to the Guest virtual machine image (this will be used if using the QEMU CLI). Click the **Finish** button to begin the installation of the Guest OS.
8. Follow the steps provided in the "Installing CentOS" section of the appropriate *Getting Started Guide* to install the Guest OS.
9. Shut down the guest OS.

   By default, the guest image is created in the `/var/lib/libvirt/images` directory. This image can be used by libvirt APIs (virsh tools) and qemu-kvm to run the guest.

2.4 Installing and Configuring Intel® QuickAssist Technology Software

The following sections detail the steps to use the libvirt* Virtual Machine Manager GUI, though similar steps are possible using the command line interface.

2.4.1 Installing Intel® QuickAssist Technology Software on Host

Note: If you are not using SR-IOV and are instead passing through a Physical Function (PF) for acceleration services on one guest only, it is not required to install the Intel® QuickAssist Technology Software package on the host.

Note: The `installer.sh` or configure script included with the software package will automatically take care of certain build environment details, including setting any SRIOV environment variable to the correct value and copying over the correct sample configuration files. If you are not using an included script to build and install the software, you must perform these operations yourself, using the included script as a guide.

1. If the Intel® QuickAssist Technology software package has a configure script, enable the SR-IOV build on the host by using:
   ```
   # ./configure --enable-icp-sriov=host
   ```
2. Check the log file for any error messages. `InstallerLog.txt` is appended after each installation with the time/date and the output of the build/install. If any issues were seen during the installation, check the log file for details.

2.4.2 Verifying SR-IOV on the Host

Note: If you are not using SR-IOV, skip this section.

Note: Sample configuration files have been included in the software package.

1. Optional: View the sample SR-IOV configuration files that were copied to the `/etc` directory. Note that any software instances that are specified in the PF (non-VF) configuration files will not be created. The sample SR-IOV configuration file has `SRIOV_Enabled = 1` and it sets the number of kernel service instances to 0.
2. Verify the VFs by running the following command in the host OS. As an example, with one high-end Intel® C62X Chipset in the system, the output would have 16 or more 37c9 devices, as shown below:

   ```
   # lspci | grep 37c9

   01:01.0 Co-processor: Intel Corporation Device 37c9
   01:01.1 Co-processor: Intel Corporation Device 37c9
   01:01.2 Co-processor: Intel Corporation Device 37c9
   ...
   01:01.7 Co-processor: Intel Corporation Device 37c9
   01:02.0 Co-processor: Intel Corporation Device 37c9
   01:02.1 Co-processor: Intel Corporation Device 37c9
   ...
   01:02.6 Co-processor: Intel Corporation Device 37c9
   01:02.7 Co-processor: Intel Corporation Device 37c9
   ```

   As another example, with one Intel® Communications Chipset 8925 to 8955 Series device in the system, the output would have 32 0443 devices, as shown below:

   ```
   # lspci| grep 0443
   01:01.0 Co-processor: Intel Corporation Device 0443
   01:01.1 Co-processor: Intel Corporation Device 0443
   01:01.2 Co-processor: Intel Corporation Device 0443
   ...
   01:01.7 Co-processor: Intel Corporation Device 0443
   01:02.0 Co-processor: Intel Corporation Device 0443
   01:02.1 Co-processor: Intel Corporation Device 0443
   ...
   ```

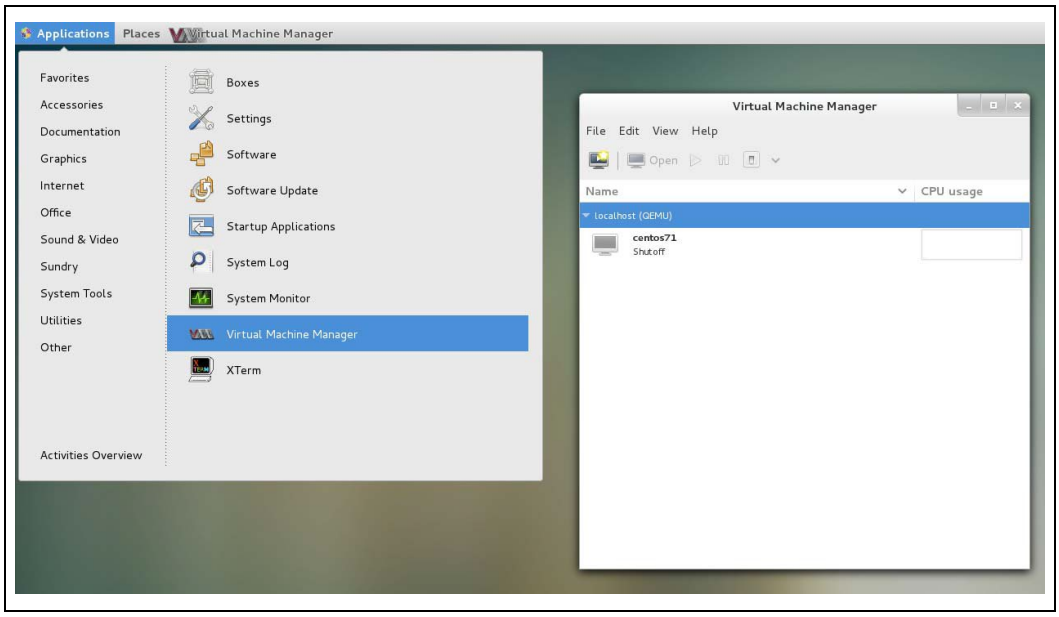
```
01:02.6 Co-processor: Intel Corporation Device 0443
01:02.7 Co-processor: Intel Corporation Device 0443
01:03.0 Co-processor: Intel Corporation Device 0443
01:03.1 Co-processor: Intel Corporation Device 0443
...
01:03.6 Co-processor: Intel Corporation Device 0443
01:03.7 Co-processor: Intel Corporation Device 0443
01:04.0 Co-processor: Intel Corporation Device 0443
01:04.1 Co-processor: Intel Corporation Device 0443
...
01:04.6 Co-processor: Intel Corporation Device 0443
01:04.7 Co-processor: Intel Corporation Device 0443
```

Refer to Table 3 for supported devices and their device IDs.

2.4.3 Pass-through the PCI Device

1. Start Virtual Machine Manager using **Application > System Tools > Virtual Machine Manager**.

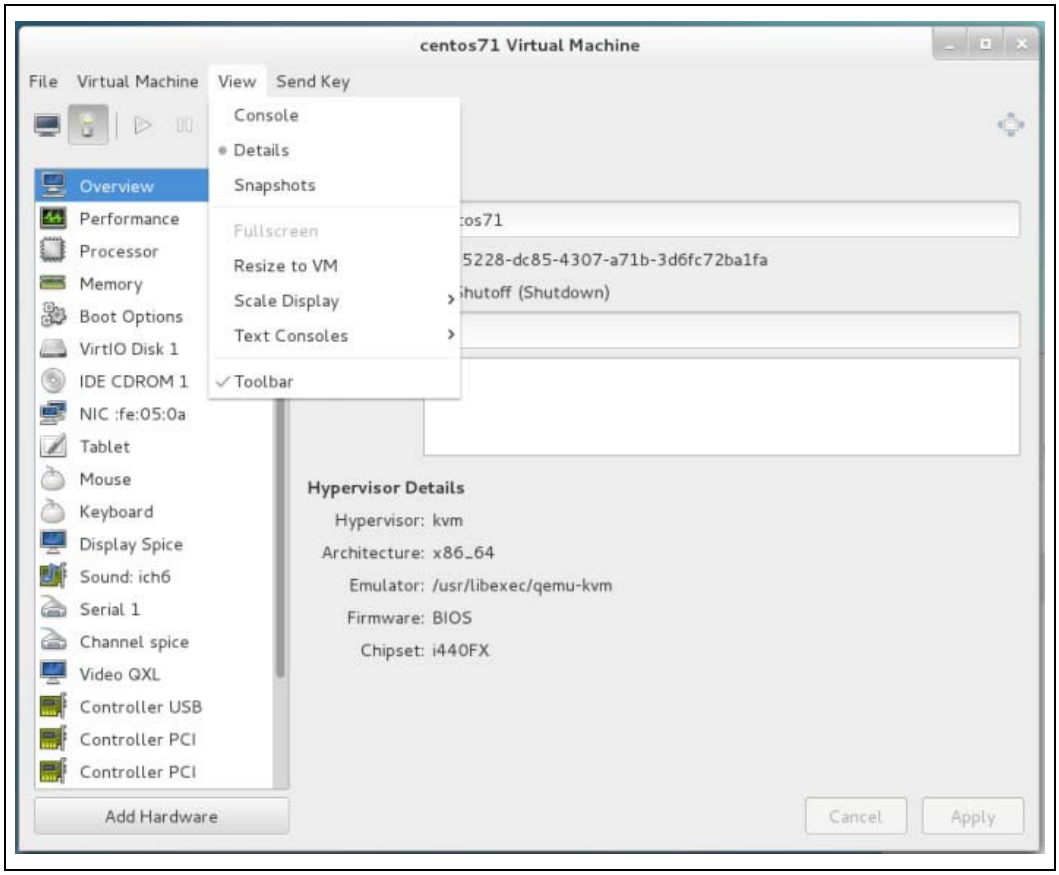
Figure 1. Virtual Machine Manager



2. Right-click on the guest and click **Open** (Do not run the guest).

   A new window for the virtual machine is displayed. Go to **View > Details**.

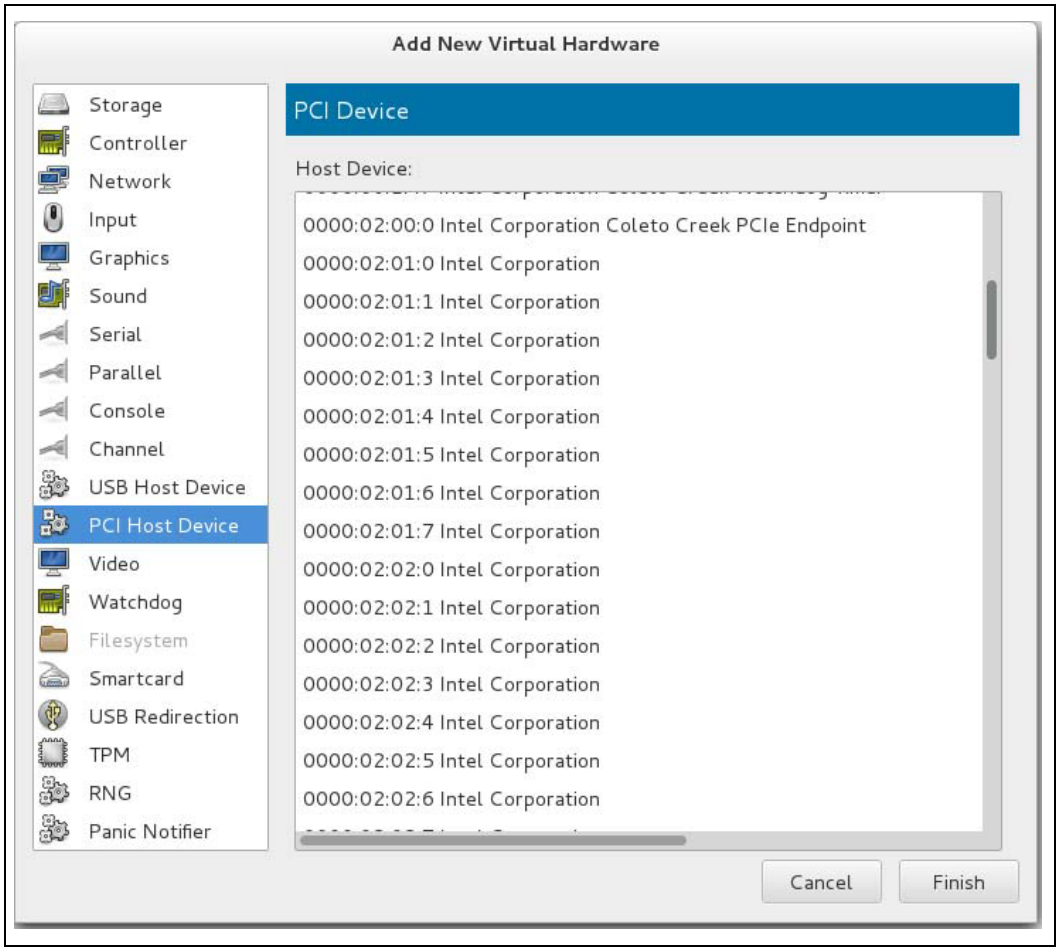
Figure 2. View Virtual Machine Details



Configure the processor, memory, boot options, and virtual hardware for the guest.

3. To add co-processor virtual functions (refer to Table 3 for supported devices and their device IDs) or GigE ports, click **Add Hardware** in the bottom-left corner and click **PCI Host Device**.

Figure 3. Add New Virtual Hardware



Select the appropriate PCI device (for instance, in the figure above, `02:01:1` is one of the 0443 devices) to attach to Guest and click **Finish**. This newly added device should appear in the left column of details for the Guest.

Note: This action will internally unbind the PCI device from the Host driver currently being used and bind it to vfio-pci (CentOS 7.1). If using a CLI, a similar sequence is: `virsh-detach <pci_func>` and `virsh-attach <domain> <pci_func>` .

4. Optional: To detach a PCI device from the guest, click the PCI device to be detached from the details page left column and click **Remove** (bottom row).

Note: You can add and remove some PCI devices while the guest is running.

5. To run the guest, go to **Virtual Machine > Run** or click the **Play** radio button on the Menu bar.
6. To view the guest console, go to **View > Console**.

2.4.4 Installing Intel® QuickAssist Technology Software on the Guest

1. In the Guest OS, verify that the appropriate device has been passed through (see Section 2.4.3), as evidenced by the `lspci` command. Refer to Table 3 for PF and VF device IDs.
2. Install the Intel® QuickAssist Technology Software package on the Guest.
3. Enable the SR-IOV build on the host by using:
   ```
   # ./configure --enable-icp-sriov=guest
   ```
4. Check the log file for any error messages.

Note: View the sample VM configuration file `/etc/<device>_dev0.conf`. Note that this configuration file supports a limited number of service instances. Specifically, the limitations is a budget of 16 rings per VF. Refer to the relevant *Programmer's Guide* (Table 2) for more information on the configuration file formats. More devices can be passed through if more service instances are required.

2.5 Running Acceleration Services Simultaneously in Host and Guest

Follow the steps below to run acceleration services simultaneously.

Using SR-IOV, acceleration running on the Host does not use the PF; instead, it uses one or more VFs. These VFs cannot then be assigned to any Guest.

Note: Update the kernel boot parameters to include "`intel_iommu=on`". Refer to the relevant *Getting Started Guide* (Table 2) as a reference to update `grub2`, and reboot.

1. Install the SR-IOV Host acceleration on the Host machine as outlined in Section 2.4.1.
2. Edit the `<device>_dev0.conf` file in the host to add Crypto and Compression instances. Use the other configuration files that are included in the Intel® QuickAssist Technology Software package as a guide to ensure the correct syntax.

Note: If more than one acceleration device is present, the following commands will need to consider the configuration of `dev1`, `dev2`, etc.

The following sample shows the partial modifications to the USER section for establishing two crypto and one compression instance:

```
[SSL]
NumberCyInstances = 2
NumberDcInstances = 1
NumProcesses = 1
# Crypto - User instance #0
Cy0Name = "SSL0"
:
# Crypto - User instance #1
Cy1Name = "SSL1"
:
# Data Compression - User space instance #0
Dc0Name = "UserDC0"
```

3. Restart the acceleration service.

```
# service qat_service restart
```

4. Use any of the free VFs and passthrough to the guest.
5. After verifying that `qat_service` is running without any issues in Host and Guest, user space code can be executed simultaneously in the Host and Guest.

2.6 Enabling Virtual Functions in QAT

To enable the virtual functions for Intel® QuickAssist Technology endpoints designed to work with the Hardware Version 1.7 software packages, run the command

```
# service qat_service_vfs start
```

This command writes a `1` to the `/sys/bus/pci/devices/0000:$bdf/sriov_numvfs` file for any QAT endpoints in the system.

Virtual functions can now be used on the host (managed via `qat_service` or `adf_ctl`) as well as the guest.

§

Appendix A FAQ

A.1 Q: How can I pass through the QAT PF to a guest?

QAT1.7 devices are not fully compliant to PCI specs. For this reason, when a FLR is done on the device by a driver different than the QAT driver (e.g. `vfio-pci`), the value of MPS is restored to the reset value and not the previous value. This has an impact on full direct pass-through. Follow this procedure to assign a device (using 8086:37c8 as an example) to a guest using full direct passthrough:

On the host:

1. Load the `vfio-pci` driver.

   ```
   modprobe vfio-pci
   ```

2. Bind the `vfio` driver to qat devices.

   ```
   echo 8086 37c8 > /sys/bus/pci/drivers/vfio-pci/new_id
   ```

3. Read the device MPS.

   ```
   lspci -vvvnd 8086:37c8 | grep "MaxPayload [1-9]* bytes, Max"
   ```

On the guest:

Note: Make sure QAT driver is not installed in the guest!

Enter a command of the form:

`qemu-system-x86_64 -enable-kvm -hda` <path to your HD image> `-m` <memory in MB>`M -device vfio-pci,host=`<BDF of your QAT device>

For example:

```
qemu-system-x86_64 -enable-kvm -hda /var/lib/libvirt/images/f24.qcow2
-m 2048M -smp 16,cores=8,threads=1,sockets=2,maxcpus=16 -device vfio-
pci,host=03:00.0
```

On the host:

Set the MPS to its original value. For example, if the MPS in the upstream bridge is equal to 256, enter the command:

```
setpci -d 8086:37c8 0x7c.b=0x37
```

On the guest:

Install and use the driver.

Note: If the guest is rebooted, the MPS will be changed.

§