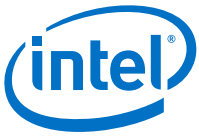
Using Intel® Virtualization Technology (Intel® VT) with Intel® QuickAssist Technology

Application Note

July 2014

Document Number: 330689-001



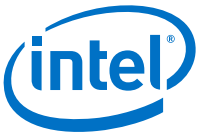
Contents

Tables

Revision History

Date	Revision	Description
July 2014	001	Updates include: • 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 1.3.1, "Where to Find Current Software and Documentation" on page 5. • Removed Fedora 14 information from Section 1.4, "Software Requirements" on page 6 and Section 2.4.1, "Using the libvirt* Virtual Machine Manager GUI" on page 9. • Added new step at the end of Section 2.2, "Installing and Configuring the Host Operating System" on page 7. • Updated Section 2.5, "Running PF/VF Simultaneously in Host and Guest" on page 17. Change bars indicate areas of change.
March 2014	1.4	Updates include: • Modified step 8 in Section 2.3, "Installing the Guest OS Image" on page 8. • Added Section 2.5, "Running PF/VF Simultaneously in Host and Guest" on page 17.
June 2013	1.3	Updates to make applicable to multiple platforms that use Intel® QuickAssist Technology.
February 2013	1.2	• Added new FAQ items: 1 and 1; deleted outdated FAQ items.
October 2012	1.1	Added Limitation.
September 2012	1.0	Initial release of this document.

§ §



1.0 Introduction

1.1 About this Document

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

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 About the Software

This section lists the features and limitations.

1.2.1 Features

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

1.2.2 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 7. KVM will not allow a VM to start with more than 7 VFs.

1.3 Documentation

1.3.1 Where to Find Current Software and Documentation

Associated software and collateral can be found on the open source website: https://01.org/packet-processing/intel%C2%AE-quickassist-technologydrivers-and-patches

Table 1 includes a list of related documentation.

1.3.2 Product Documentation

Documentation includes:

- Using Intel® Virtualization Technology (Intel® VT) with Intel® QuickAssist Technology Application Note (this document)
- Additional related documents listed in Table 1, which may be accessed as described in Section 1.3.1.

Table 1. Related Documents

Document Name	Number
Intel® QuickAssist Technology Software Release Notes	330683
Intel® Communications Chipset 8900 to 8920 Series Software for Linux* Getting Started Guide	330752
Intel® Communications Chipset 8925 to 8955 Series Software for Linux* Getting Started Guide	330750
Intel® Communications Chipset 8900 to 8920 Series Software Programmer's Guide	330753
Intel® Communications Chipset 8925 to 8955 Series Software Programmer's Guide	330751
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® Communications Chipset 89xx Series Datasheet	327879

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

1.3.3 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

1.4 Software Requirements

Software requirements will vary by the particular use case.

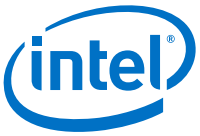
- Required: the Intel® QuickAssist Technology Software for Linux*
  It is recommended to use the same version of the QuickAssist driver on host and guest. However, changes from QATmux1.1.0 onwards are generally done in a backwardly compatible manner and there is a version compatibility check between drivers running on Host and Guest when bringing up the driver on the Guest.

These instructions were tested against the following Linux distribution:

- Fedora* 16 64-bit version, Kernel: GNU*/Linux* 3.1.0-7

2.0 Using Intel® QuickAssist Technology 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 is available under:
   Advanced > System Agent (SA) Configuration > VT-d
3. Enable the SR-IOV parameter in BIOS. The option is 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 Fedora* 14 64-bit version or Fedora* 16 64-bit version. Consult the *Getting Started Guide* section "*Installing the OS on a Development Board*" if necessary, taking note that this guide assumes Fedora* 16 64-bit version as the host OS when SR-IOV is used. The software will not work "out of the box" with SR-IOV and Linux* versions that are much older.

   Note: The *Getting Started Guide* uses the kernel boot parameter `intel_iommu=off`. That kernel boot parameter must not be included for the purposes of this guide. If this kernel boot parameter is set, remove it, using the *Getting Started Guide* as a reference to update grub2, and reboot.

   Note: Some Linux distributions may require the kernel boot parameter `intel_iommu=on` if using SR-IOV.
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, python-virtinst, qemu, virt-manager, virt-viewer and all of the dependencies that are needed.
3. Start the libvirtd service using the commands:

   ```
   # chkconfig libvirtd on
   # service libvirtd start
   ```
4. Verify SR-IOV hardware capabilities using the command:

   For Intel® Communications Chipset 8900 to 8920 Series:

   ```
   # lspci -vnd 8086:0434
   ```

   For Intel® Communications Chipset 8925 to 8955 Series:

   ```
   # lspci -vnd 8086:0435
   ```

   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. Shutdown the system:

```
# shutdown –h now
```

8. Power on the system and proceed with the instructions in 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 00000000bdf1d918 00110 (v01 INTEL  ROMLEY   06222004
INTL 20090903)
[    0.000000] Intel-IOMMU: enabled
[    0.065775] DMAR: Host address width 46
[    0.065778] DMAR: DRHD base: 0x000000fbffe000 flags: 0x0
[    0.065807] IOMMU 0: reg_base_addr fbffe000 ver 1:0 cap d2078c106f0466 ecap
f020de
[    0.065810] DMAR: DRHD base: 0x000000ebffc000 flags: 0x1
[    0.065835] IOMMU 1: reg_base_addr ebffc000 ver 1:0 cap d2078c106f0466 ecap
f020de
[    0.065837] DMAR: RMRR base: 0x000000bddc8000 end: 0x000000bddd7fff
[    0.065839] DMAR: ATSR flags: 0x0
[    0.065964] IOAPIC id 2 under DRHD base  0xfbffe000 IOMMU 0
[    0.065967] IOAPIC id 0 under DRHD base  0xebffc000 IOMMU 1
[    0.065968] IOAPIC id 1 under DRHD base  0xebffc000 IOMMU 1
[    4.601550] IOMMU 0 0xfbffe000: using Queued invalidation
[    4.601553] IOMMU 1 0xebffc000: using Queued invalidation
[    4.601565] IOMMU: Setting RMRR:
[    4.601591] IOMMU: Setting identity map for device 0000:00:1d.0 [0xbddc8000
- 0xbddd7fff]
[    4.601632] IOMMU: Prepare 0-16MiB unity mapping for LPC
[    4.601654] 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: These instructions were tested with Virtual Machine Manager, version 0.9.1. Earlier versions of Virtual Machine Manager (for example, version 0.8.7 which is included with Fedora* 14) will not work with SR-IOV.

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: During 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 on 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 sufficient amount, but it should not affect the Host OS, e.g., for 4 GB RAM and 8 cores, allocate Guest OS < 2GB 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 the **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** the 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 Guest OS.
8. Follow the steps provided in the "*Installing Fedora*" section of the appropriate Getting Started Guide to install the Guest OS.

   When installing Fedora, if the following error message appears:

   "Cannot display graphical console type 'spice': /usr/lib64/libspice-client-gLib-2.0.so.1: Undefined symbol: usbredirhost_get_guest_filter"

   Install the `usbredir` package with the following command:

   ```
   # yum install usbredir
   ```
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 Using the libvirt* Virtual Machine Manager GUI

Intel recommends you use Fedora* 16 as the Host OS when using the libvirt* Virtual Machine Manager GUI.

2.4.1.1 Installing Intel® QuickAssist Technology Software on Host

Note: If you are not using SR-IOV and are instead passing through a 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` included with the software package will automatically take care of certain build environment details, including setting `ICP_SRIOV` and `ICP_HOST_SRIOV` to the correct value and copying over the correct sample configuration files. If you are not using the included `installer.sh` to build and install the software, you must perform these operations yourself, using the included `installer.sh` as a guide.

1. Install the Intel® QuickAssist Technology Software package on the host, choosing the **Install SR-IOV Host Acceleration** option. Consult the *Getting Started Guide* (see Table 1 on page 6) for more information.
2. Check the log file for any error messages. `InstallerLog.txt` is appended to 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.

2.4.1.2 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 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 Intel® Communications Chipset 8900 to 8920 Series SKU4 in the system, the output would have 16 0442 devices, as shown below:

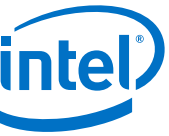
```
# lspci | grep 0442
01:01.0 Co-processor: Intel Corporation Device 0442 (rev 21)
01:01.1 Co-processor: Intel Corporation Device 0442 (rev 21)
01:01.2 Co-processor: Intel Corporation Device 0442 (rev 21)
...
01:01.7 Co-processor: Intel Corporation Device 0442 (rev 21)
01:02.0 Co-processor: Intel Corporation Device 0442 (rev 21)
01:02.1 Co-processor: Intel Corporation Device 0442 (rev 21)
...
01:02.6 Co-processor: Intel Corporation Device 0442 (rev 21)
01:02.7 Co-processor: Intel Corporation Device 0442 (rev 21)
```

Note: Do not detach 8086:043e; it is used for debug purposes.

```
Co-processor [0b40]: Intel Corporation Device [8086:043e]
```

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:

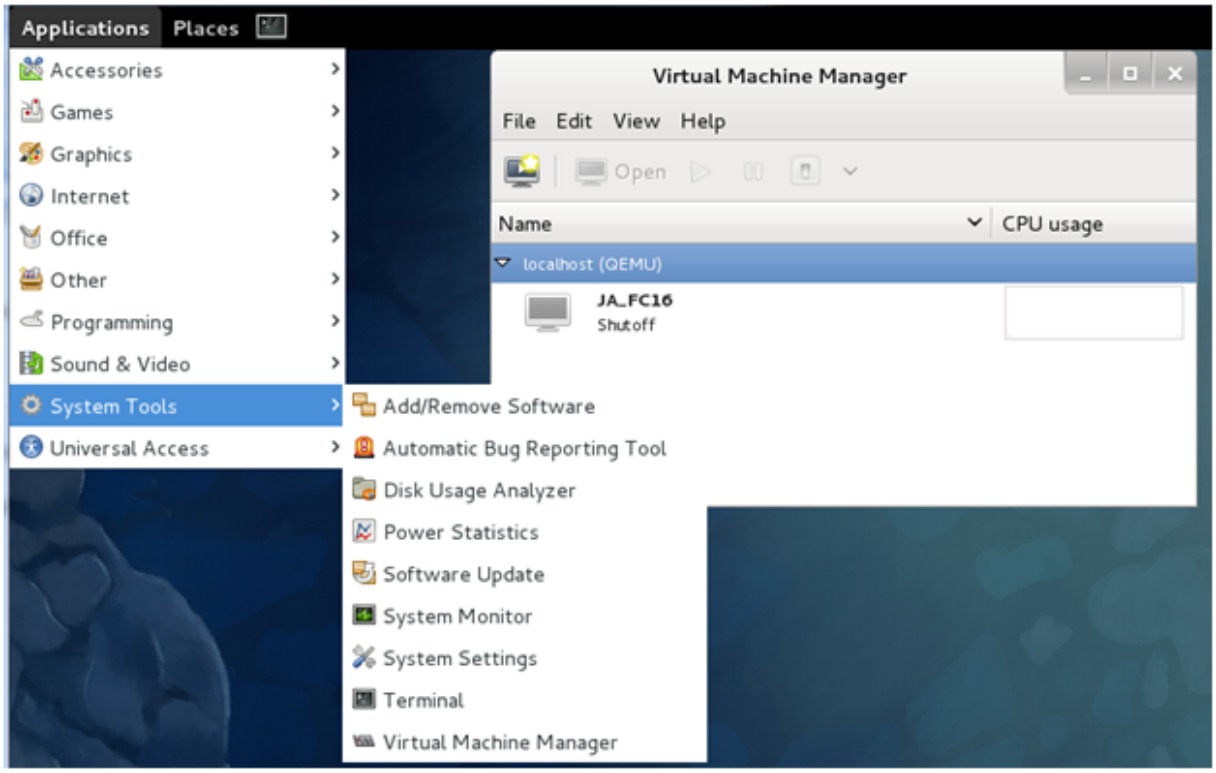
```
# lspci | grep 0443
01:01.0 Co-processor: Intel Corporation Device 0443
01:01.1 Co-processor: Intel Corporation Device 0443
01:01.2 Co-processor: Intel Corporation Device 0443
...
01:01.7 Co-processor: Intel Corporation Device 0443
01:02.0 Co-processor: Intel Corporation Device 0443
01:02.1 Co-processor: Intel Corporation Device 0443
...
01:02.6 Co-processor: Intel Corporation Device 0443
```



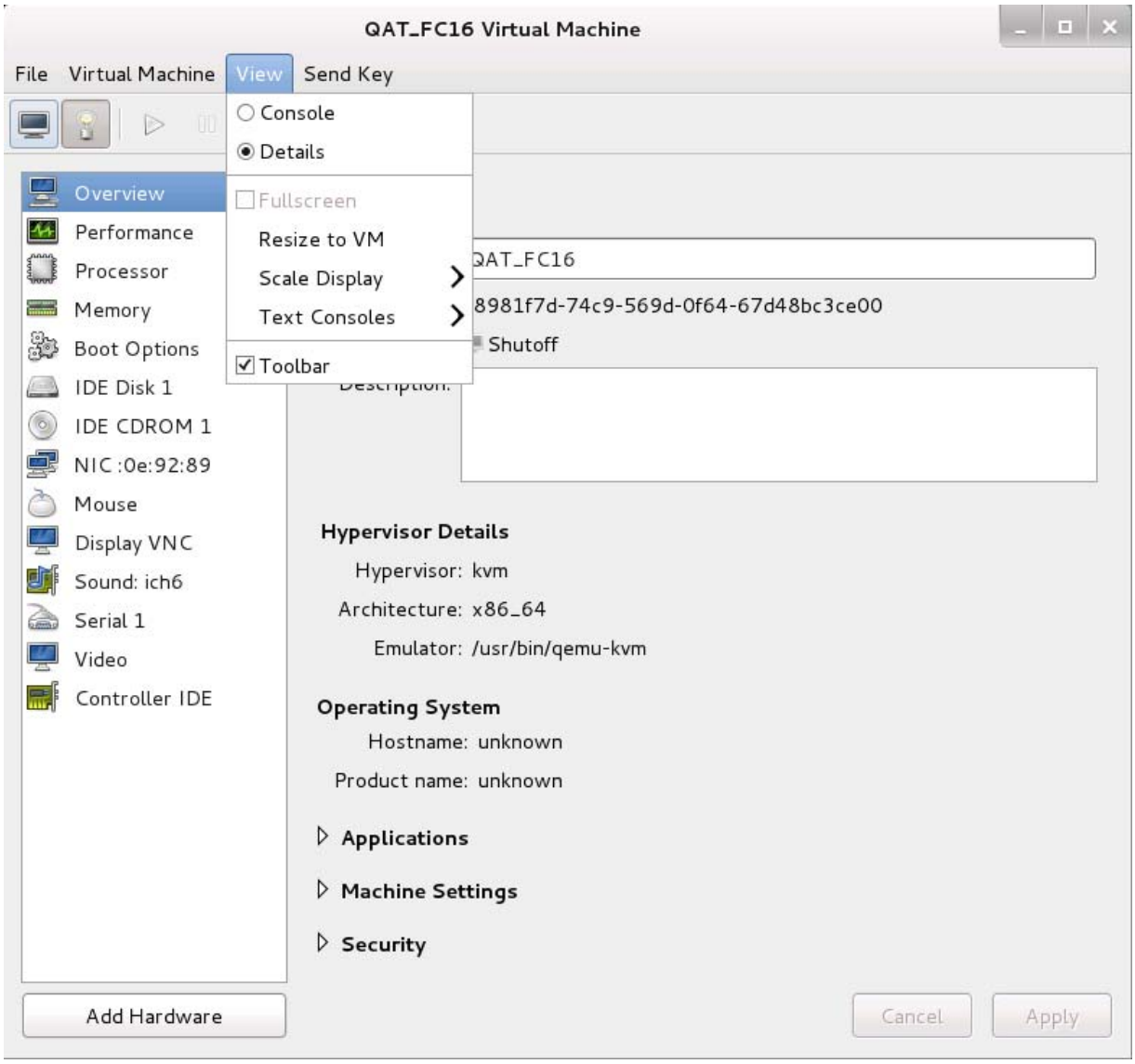
```
01:02.7 Co-processor: Intel Corporation Device 0443
01:03.0 Co-processor: Intel Corporation Device 0443
01:03.1 Co-processor: Intel Corporation Device 0443
...
01:03.6 Co-processor: Intel Corporation Device 0443
01:03.7 Co-processor: Intel Corporation Device 0443
01:04.0 Co-processor: Intel Corporation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

2.4.1.3 Pass-through the PCI Device

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

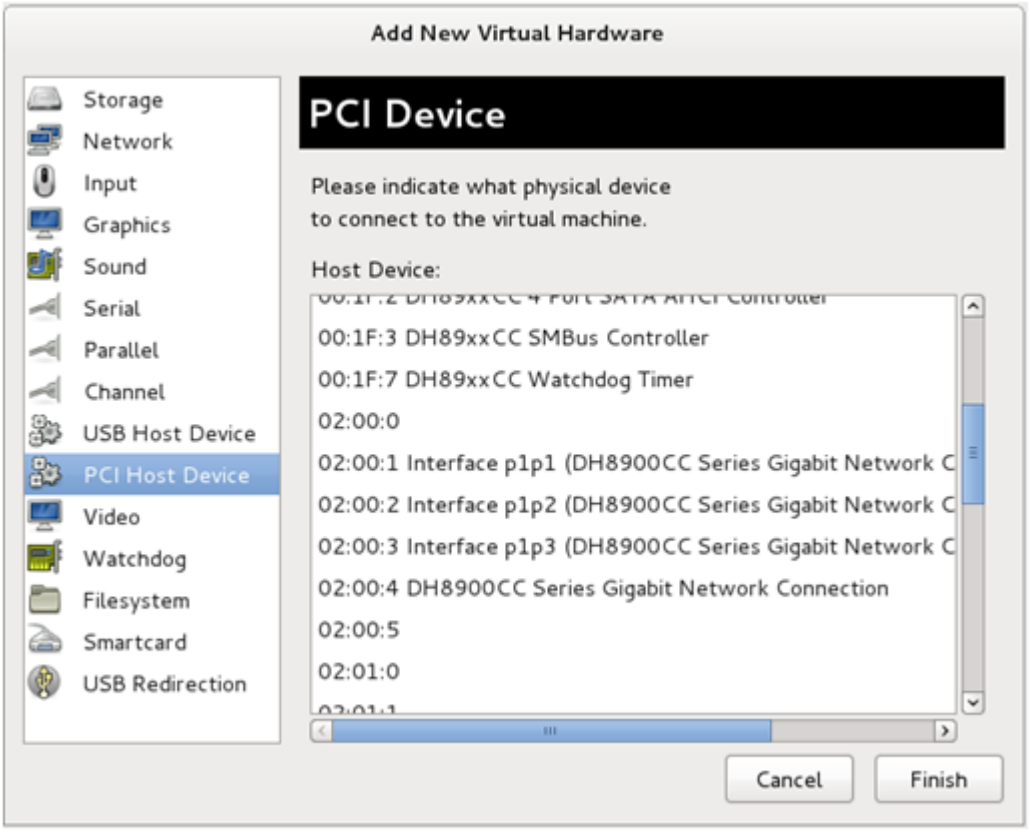


2. Right-click on the guest and click **Open** (Do not run the guest).
   A new window for the virtual machine is displayed. Go to **View > Details**



You can configure the processor, memory, boot options, virtual hardware for the guest.

3. To add co-processor virtual functions (0442 or 0443) or GigE ports, click **Add Hardware** in bottom-left corner and click **PCI Host Device**.



   Select the appropriate PCI device (for instance, in the figure above, 02:01:1 is one of the 0442 devices) to attach to Guest and click **Finish**. This newly added device should appear on left column of details for the Guest.

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

   Note: Do not use pass through for any 0442 devices with function 0 (for instance, "02:01.0"). These virtual functions are used by the physical functions.

4. Optional: To detach a PCI device from the guest, click the PCI device to be detached from details page left column and click **Remove** (bottom row). If you are using Fedora* 16, click **Finish**.

   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 **Play** radio button on Menu bar.

6. To view the guest console, go to **View > Console**.

2.4.1.4 Installing Intel® QuickAssist Technology Software on the Guest

1. In the Guest OS, verify that the appropriate device has been passed through (see Section 2.4.1.3), as evidenced by the `lspci` command.
   a. For passthrough of Virtual Function (VF) using SR-IOV, this should be one or more 0442 or 0443 devices.
   b. For passthrough of Physical Function (PF), this should be a 0434 or 0435 device.
2. Install the Intel® QuickAssist Technology Software package on the Guest.
   a. For passthrough of VF using SR-IOV, choose the **Install SR-IOV Guest Acceleration** option.
   b. For passthrough of PF, choose the **Install Acceleration** option.

   Consult the Getting Started Guide (Table 1) for more information.
3. 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.

Note: View the sample VM configuration file `/etc/dh<device>cc_qa_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. See the relevant Programmer's Guide (Table 1) and Appendix A, "FAQ" for more information on the configuration file formats. More 0442 or 0443 devices can be passed through if more service instances are required.

2.4.2 Using QEMU* KVM Command Line Interface

Note: This section can be skipped if you are using the libvirt* Virtual Machine Manager GUI to pass-through a PCI device as described in Section 2.4.1.

Note: This section uses sample device IDs which will not be on all platforms.

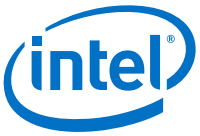
2.4.2.1 Installing Updated QEMU* KVM

Note: The latest stable qemu-kvm version is recommended, though these steps were tested with Fedora* 16 as the host OS and were verified to work with `qemu-kvm version 1.2.0`, available here:

http://sourceforge.net/projects/kvm/files/qemu-kvm/1.2.0/

1. Download `qemu-kvm-1.2.0` to `/root` (or another directory of your choosing).
2. Extract the software using the command:
   ```
   # tar –xvzf qemu-kvm-1.2.0.tar.gz
   ```
3. Install the software using the command:
   ```
   # cd qemu-kvm-1.2.0
   # ./configure
   # make
   # make install
   ```
4. Verify the software using the command:
   ```
   [root@localhost bin]# /usr/local/bin/qemu-system-x86_64 --version
   QEMU emulator version 1.2.0 (qemu-kvm-1.2.0), Copyright (c) 2003-2008 Fabrice Bellard
   ```
   Ensure that the version string shows 1.2.0.

When running the `qemu-system-x86_64` commands, you must use the complete path (`/usr/local/bin`) not the system path.



2.4.2.2 Pass-through the PCI Device

Note: Some of these instructions were taken from:
http://www.linux-kvm.org/page/How_to_assign_devices_with_VT-d_in_KVM

1. Identify the Bus/Device/Function numbers using the command:

   ```
   # virsh nodedev-list --tree
   ```

   The display printout will be something like this:

   ```
   computer
    |
     +- net_lo_00_00_00_00_00_00
     +- pci_0000_00_00_0
     +- pci_0000_00_01_0
     +- pci_0000_00_01_1
     |   |
     |   +- pci_0000_02_00_0
     |   +- pci_0000_02_00_1
     |   |   |
     |   |   +- net_eth0_00_13_20_f9_9f_7f
     |   |
     |   +- pci_0000_02_00_2
     |   |   |
     |   |   +- net_eth1_00_13_20_f9_9f_80
   ```

2. Select the required PCI device. In this example, we are selecting `pci_0000_02_00_1`, an Ethernet port and the acceleration complex is `pci_0000_02_00_0`.

3. Look for the same PCI device using the command:

   ```
   # lspci -nn
   :
   02:00.1 Ethernet controller [0200]: Intel Corporation DH8900CC Series Gigabit
   Network Connection [8086:0438]
   :
   ```

   Note: The Ethernet port may be `Intel Corporation DH8900CC Series Gigabit Network Connection [8086:0436]` if the latest firmware has not been installed.

   Record the `bus:device.function`, which is `02:00.1` in this example. You will need this parameter when you start the guest and pass through this device.

   Note: The commands that follow use the following conventions:

   — `<bb:dd.f:Ethernet>` should be replaced by the specific bus/device/function that applies to the specific Ethernet port chosen (for instance, `02:00.1` or `01:00.1`).

   — `<bb_dd_f:Ethernet>` will be used in some commands that use underscores (_) and applies to the specific Ethernet port chosen (for instance, `02_00_1` or `01_00_1`).

   — `<bb:dd.f:accel>` should be replaced by the specific bus/device/function that applies to the accelerator (`8086:0434`).

   — `<bb_dd_f:accel>` will be used in some commands that use underscores (_) and applies to the accelerator (`8086:0434`).

4. **Detach the Ethernet port from the host OS**, replacing the placeholders with the appropriate bus/device/function text:

   ```
   # virsh nodedev-dettach pci_0000_<bb_dd_f:Ethernet>
   Device pci_0000_<bb_dd_f:Ethernet> dettached
   ```

5. **Detach the acceleration complex from the host OS**, replacing the placeholders with the appropriate bus/device/function text:

```
# virsh nodedev-dettach pci_0000_<bb_dd_f:accel>
Device pci_0000_<bb_dd_f:accel> dettached
```

6. Verify that both devices are detached (optional):
   For the selected detached device, the command `lspci –vv` returns the line: `kernel driver in use: pci-stub`. The most effective way to find this is to use the command `lspci -vv | less` and search for pci-stub and scroll up to find the 0434 or 0438 keyword. Alternatively, use the command: `lscpi -vv | grep pci-stub`, which should return two lines.

2.4.2.3 Creating a Shared Folder

The Intel® QuickAssist Technology Software and certain test suites (such as Canterbury Corpus) must be accessible to the guest. One way (if the guest cannot be assigned a shared or dedicated network connection) is to create an ISO image of the folder with the software that we want to share and mount it as a cdrom in the qemu-system-x86_64 CLI.

In this example, the folder name is `/tmp/QAT`:

```
[root@localhost ~]# mkisofs -o /tmp/cd.iso /tmp/QAT
I: -input-charset not specified, using utf-8 (detected in locale settings)
Total translation table size: 0
Total rockridge attributes bytes: 0
Total directory bytes: 0
Path table size(bytes): 10
Max brk space used 0
1529 extents written (2 MB)
[root@localhost ~]#
```

2.4.2.4 Starting the Guest

Run the following command in the terminal window of the host, replacing the placeholders with the appropriate bus/device/function text:

```
# /usr/local/bin/qemu-system-x86_64 \
-m 1024                 \
-boot c                 \
-net none               \
-hda /var/lib/libvirt/images/fedora-kvm.img \
-cdrom /tmp/cd.iso      \
-device pci-assign,host=<bb:dd.f:accel> \
-device pci-assign,host=<bb:dd.f:Ethernet>
```

Parameters are separated by lines for understanding and readability.

Note: The `m` option assigns memory to the guest (in MB). For testing purposes, Intel recommends at least 2 GB of system memory and at least 1 GB assigned to the virtual machine.

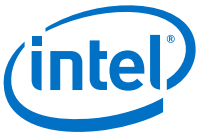
Note: The following status may be ignored:

```
Using raw in/out ioport access (sysfs - Input/output error)
```

2.4.2.5 Verifying Pass-through

Once the Guest starts, run the following command in the Guest:

```
#lspci -nn
```



Pass-through PCI devices should appear with the same description as the Host originally showed. For example, if the below was shown on the Host:

```
Intel Corporation DH8900CC Series Gigabit Network Connection [8086:0438]
```

It should show up on the Guest as:

```
Ethernet controller [0200]: Intel Corporation Device [8086:0438]
```

If the description is not same as the Host has, then shutdown the guest and reboot it again. For instance, the Ethernet controller may be listed in lspci as `Non-VGA unclassified device`. If the problem persists despite several shutdown and boot cycles, it may be necessary to detach and pass through one of the other Ethernet ports, but this is not required if no network access is required at this point.

This problem has not been observed with co-processor [8086:0434].

2.4.2.6 Installing Intel® Communications Chipset 8900 to 8920 Series Software in KVM Guest

A CDROM icon should be visible on the virtual machine desktop.

1. Double-click this icon to browse the files on the CDROM image. Some filenames may be truncated, so it may be necessary to re-extract the original tarball.
2. Right-click on `dh89x001.tgz` and select **Copy**.
3. Browse to a temporary directory (for instance the `Downloads` directory of the user account), and paste this file there.
4. Create a directory `/CRF_Release` (or another working directory of your choosing) and copy `dh<device>cc.tgz` there and extract it with the command:
   `tar xvzf dh<device>001.tgz`

Installation steps are the same as given in the Intel® Communications Chipset 8900 to 8920 Series Software for Linux* Getting Started Guide and summarized below:

- Install Acceleration Software

  Note: QEMU does not support extended PCI configuration space. On the guest with PF passthrough, PCI AER checking must be disabled before Acceleration Software compilation. Appendix A, “FAQ”, item 1 describes the solution.

- Compile and execute QuickAssist Acceleration sample application (Kernel space and User space versions)

Note: During the above steps, you will reboot the guest several times. Make sure the pass-through device appears properly in guest.

2.5 Running PF/VF Simultaneously in Host and Guest

1. Install the SR-IOV Host acceleration on the Host machine as outlined in Section 2.4.1.1.
2. Edit the `dh89xxcc_qa_dev0.conf` or `dh895xcc_qa_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. If using the "v1" config file (see Table 1 for related Programmer's Guide), note that the VFs will use bank numbers that correspond to their VF number. For instance, the first VF uses bank number 0, the second VF uses bank number 1, and so on. If the configuration file on the PF uses any resources from a given bank, it is not possible to use the VF that would have used that bank number.

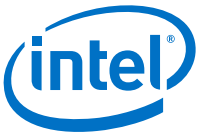
   If using the "v2" config file, the configuration file needs to set PF_bundle_offset in the configuration file to specify what bank numbers to reserve for PF use. See the section “General Parameters” in the related Programmer's Guide for more information.

Note: Do not use passthrough for any 0442 devices with function 0 (for instance,“02:01.0 and 02:02.0”); these virtual functions are used by the physical functions.

4. Restart the acceleration service.

   ```
   # service qat_service restart
   ```

5. Use any of the free VFs and passthrough to the guest as mentioned in Section 2.4.2.2.
6. 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.



Appendix A FAQ

1. How can I achieve load balancing when multiple VFs are being used?

   Note: This FAQ only applies to the Intel® Communications Chipset 8900 to 8920 Series. It does not apply to the Intel® Communications Chipset 8925 to 8955 Series.

   The top two PCH SKUs have two accelerators, each with two crypto engines and one compression engine. For the top SKU, VF numbers 1-7 use the hardware resources (i.e., engines) of the first accelerator, and VF numbers 9-15 use the hardware resources (i.e., engines) of the second accelerator. To achieve load balancing when using one or more VFs from one PCH device, it is necessary to take special consideration to select sensible values for the AcceleratorNumber parameter (plus the ExecutionEngine parameter, if using configuration file version 2) in the particular guest configuration file(s).

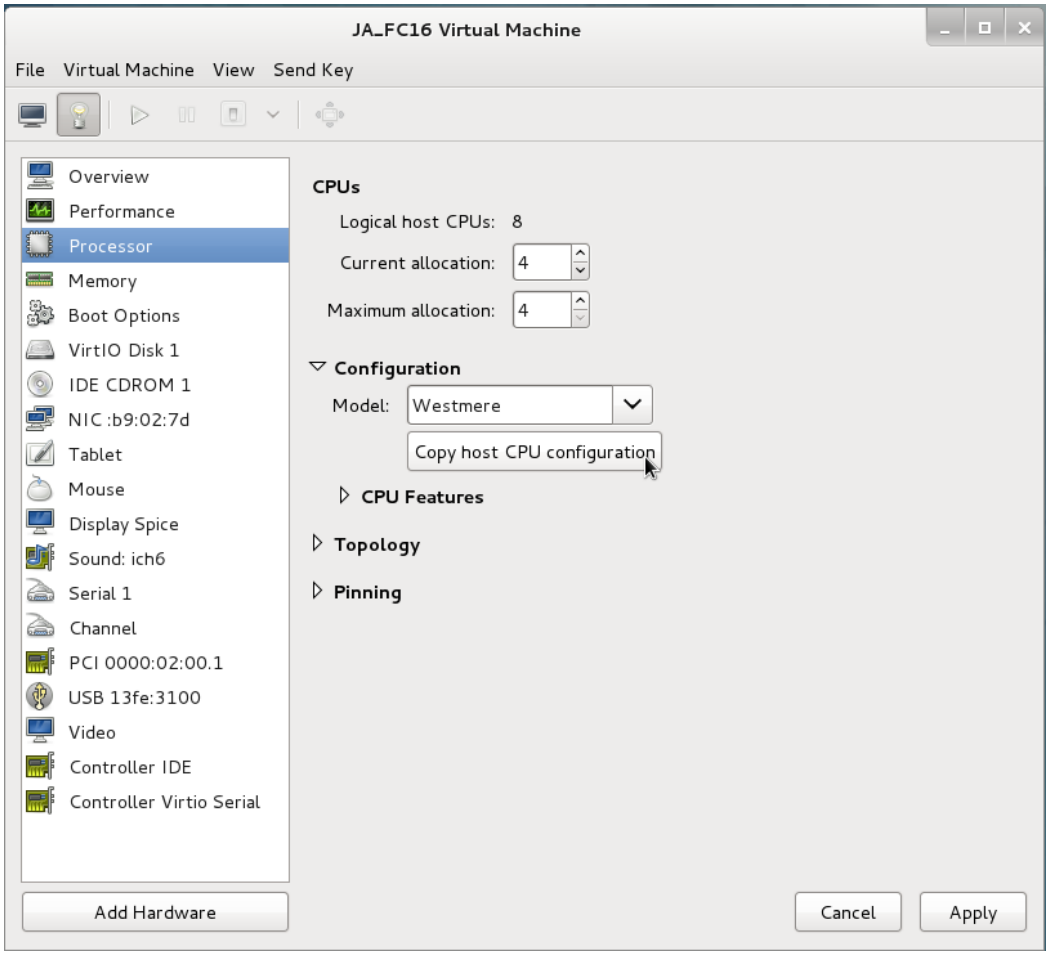
   For instance, take the case where VF#1 is passed through to a guest OS. By default, the configuration file for the guest has two crypto instances: one is assigned to the first crypto engine for the first accelerator, and one is assigned to the second crypto engine for the first accelerator. If running crypto on this VF, the application can load balance between these two engines. If two VFs are passed through, the second VF should be chosen from the range 9 to 15 for the best load balancing, since the second VF would be otherwise competing with the first VF for use of the crypto engines. Likewise, to have the best compression performance, an application should have access to at least one compression instance from a VF in the range 1-7 and at least one compression instance from a VF in the range 9-15.

   For more information on the configuration file options, see the appropriate Programmer's Guide (see Table 1).

2. **Can I get Host CPU in the Guest? How?**

   Yes. If you want to run Intel® QuickAssist Technology Software and Intel® DPDK in the guest you will need Host CPU. By default, the hypervisor provides emulated CPU to the guest. If you are using Virt-Manager GUI, go to **Guest Machine details**, then **Processor > Configuration** then select **Copy host CPU**

configuration. If you are using qemu command line, use `-cpu host` parameter to provide host CPU to guest.



§ §