Graphics Virtualization Challenges

April-29-2010

Allen Kay

allen.m.kay@intel.com
Agenda

- **Background**
  - High Level PCI Driver Operation Flow
  - Native Device Initialization
  - QEMU IO Virtualization
  - Device Pass-through

- **Graphics Pass-through**
  - Base Changes
  - Discrete Graphics
  - Integrated Graphics Device (IGD)

- **Current Status**

- **Future Work**
High Level PCI Driver Operation Flow

- Initialize device base on vendor_id and device_id
- Map device MMIO area containing device control registers
- Using device control registers to initiate DMA
- Device interrupts CPU when DMA completes
Background: Native Device Initialization

- Device driver finds matching device with vendor/device ID
  - Reads device’s PCI configuration space
  - Using IO ports 0xCF8 and 0xCFC

- HW will respond when IO ports 0xCF8/0xCFC are accessed
  - OS writes PCI bus:dev:func:offset into 0xCF8
  - HW returns content into 0xCFC

- IO PORT and MMIO accesses are handled by HW

- DMA operation
  - Driver programs DMA physical address in DMA HW
  - No virtual-to-physical translation from the IO side

- HW generates interrupt after DMA completion
Background: QEMU IO Virtualization

- QEMU provides virtual platform for HVM guests
  - Virtual PCI bus 0
  - Virtual PCI devices hangs off on virtual PCI bus 0

- Virtual PCI vendor/device ID’s in PCI configuration space are hardcoded to match the real HW
  - i.e. E1000 NIC has vendor_id = 0x8086, device_id = 0x100E
  - Guest pci config info stored in pci_dev->config[4096]

- Guest PCI config accesses are trapped and emulated
  - Reading virtual E1000 NIC’s device_id returns 0x100E
  - This allows E1000 driver in guest OS to initialize successfully

- IO ports and MMIO accesses are trapped and handled by QEMU

- DMA operations and interrupts are also emulated
Background: QEMU IO Virtualization

Diagram:
- Device Model
- Native Driver
- Apps
- Driver
- Hypervisor
- Device

Flow:
- IRQ Emu
- VM Exit
- Virtual Interrupt
Background: PCI Device Pass-through

- QEMU still provides virtual platform as before
  - Virtual PCI bus 0
  - Virtual PCI devices and pass-through devices hang off virtual bus 0

- PCI config handling
  - Pass-through device config space values are read off from real HW
  - Vendor and device IDs reflects pass-through device values
  - Native driver in the guest can be successfully initialized
  - Most registers are emulated by QEMU as before
  - Read/Write of some registers are pass-through to HW (i.e. COMMAND)
Background: PCI Device Pass-through

- **IO port handling**
  - Xen hypervisor intercepts guest IO port accesses
  - Converts guest IO port to host IO port
  - Check for host IO port access permission
  - Read/Write host IO port on guest’s behalf

- **MMIO handling**
  - Maps GPA to HPA mapping in hypervisor
  - Guest can access device MMIO without causing VM exits

- **DMA operations can be done without VM exits**
  - Guest device driver programs GPA to the device DMA engine
  - IOMMU HW translates GPA to HPA

- **Interrupt handling**
  - Interrupts from pass-through device is intercepted by hypervisor
  - Re-injected to the guest via vioapic->vlapic->vmcs mechanism
Graphics Pass-through: Base Changes

- **All changes are in user mode**
  - QEMU and hvmloader

- **Need to execute video BIOS in guest**
  - Copy content of physical address 0xC0000 to guest memory
  - Execute at guest BIOS startup

- **Allow access to legacy video specific IO Ports**
  - 0x3B0 - 0x3BC
  - 0x3C0 - 0x3E0

- **Setup GPA to HPA mapping of legacy video MMIO range**
  - 0xA0000 - 0xBFFFF
Graphics Pass-through: Discrete Graphics

- Targeting virtualization friendly discrete devices
  - nVidia Quadro FX3800 and ATI FireproV3750

- nVidia Quadro FX3800
  - Cheapest in the product family at $900 each
  - Video BIOS cannot be re-executed dynamically in the guest
  - Manual workaround is not reasonable
  - Cannot see boot messages
  - Video works after OS video driver is up and running

- ATI Firepro V3750
  - A more reasonable alternative at $150 each
  - Work In Progress

- Plan to use dual QEMU VGA and pass-through graphics
  - OS loader messages will be display in QEMU VGA
  - Once OS is up and running, display is switched to the pass-through graphics
Graphics Pass-through: Integrated Graphics

- **Integrated Graphics Device (IGD) OpRegion**
  - MMIO memory used for runtime driver/BIOS communication
  - Reported at vendor specific offset 0xFC in PCI config space
  - Not reported in PCI BAR
  - Need additional function call to map this region

- **Non-IGD device accessing**
  - IGD historically was part of the chipset
  - Driver accesses IOH registers for such info as Top Of Memory
    - Registers accesses are HW implementation dependent
    - Future HW will shadow these registers in IGD device (0:2.0)
  - Driver also accesses PCH on newer platforms with on die graphics
    - Windows driver hard codes device 00:1f.0
  - QEMU needs to allow such accesses when there is IGD pass-through
Current Status

- **Run Environment**
  - Guest owns the graphics device
  - Access dom0 via VNC

- **Xen 4.0 and xen-unstable supports**
  - Quadro FX3800 (with no boot messages)
  - Q35/Q45/GM45 integrated graphics

- **Working and need to submit upstream**
  - Core i3/i5
  - Sandybridge

- **Work In Progress**
  - Enabling ATI Firepro V3750 discrete graphics
Future Work

- **Dual IGD and discrete graphics support**
  - Dom0 and guest can each have a device

- **Restarting guests with graphics pass-through**

- **Dual QEMU VGA and graphics pass-through support**
  - Workaround video BIOS init problem in discrete graphics

- **Keyboard/mouse support for OS loader boot menu**
  - Allow user to select which kernel to boot

- **Raise virtualization awareness in graphics community**
  - Windows driver validation team will be using Xen for validating drivers in virtualization environment
  - Shadow IOH and PCH register accesses in IGD device
  - Report IGD OpRegion address in PCI BAR
Questions?
Background: PCI Pass-through Example

- Pass-through E1000 NIC device at BDF 1:0.0
- QEMU reads PCI configuration space register using libpci
- QEMU constructs a virtual PCI configuration space with same content as 1:0.0 on real platform
  - Example: vendor_id = 0x8086, device_ID = 0x10b9
  - Example: 1:0.0 on real platform is mapped to 2:0.0 on virtual platform
- Guest PCI configuration accesses to device 2:0.0 is handled as pass-through device
  - Most read only registers are handled same as QEMU device
  - Some registers are passed through to HW: command register
- Access to any non pass-through devices are emulated
  - Example: host bridge device 0:0.0
VT-d: Hardware Overview

- DMA Requests
- Device ID | IO Virtual Address | Length | ...
- Fault Generation
- VT-d Hardware
- Translation Cache
- Context Cache
- Memory Access with Host Physical
- Device Context Structures
  - Bus 255
  - Bus N
  - Bus 0
  - Dev P, Func 2
  - Dev 31, Func 7
  - Dev P, Func 1
  - Dev 0, Func 0
- Address Translation Structures for Domain A
- Address Translation Structures for Domain B
- 4KB Page Tables
- 4KB Page Frame
- Memory-resident IO Partitioning & Translation Structures
- Intel Confidential
• VT-d specification:

• Xen VT-d wiki:
  – http://wiki.xensource.com/xenwiki/VTdHowTo

• SR-IOV Specification:

• ATS 1.1 Specification:
LEGAL INFORMATION

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. EXCEPT AS PROVIDED IN INTEL’S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY RELATING TO SALE AND/OR USE OF INTEL PRODUCTS, INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT.

- Intel may make changes to specifications, product descriptions, and plans at any time, without notice.

- All dates provided are subject to change without notice.

- Intel is a trademark of Intel Corporation in the U.S. and other countries.

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

- Copyright © 2007, Intel Corporation. All rights are protected.