Status Update on PCI Express Support in QEmu

Isaku Yamahata, VA Linux Systems Japan K.K.
<yamahata@valinux.co.jp>

Xen Summit North America
April 28, 2010
Agenda

- Introduction
- Usage and Example
- Implementation Details
- Future Work
- Considerations on further development issues
Introduction

From http://en.wikipedia.org/wiki/PCI_Express
PCI Express native Hotplug

Electro Mechanical Lock (EMI)

Slot Number

Manual Release Latch Closed

Manual Release Latch Open

OL* Attention Button

Power LED (Green)

Attention LED (Yellow)

From http://docs.hp.com/
Eventual Goal

Error Message

PCI Express native passthrough
With native hot plug support

Inject the error

Interrupt to notify the error

PCI Express bus

DomU

Dom0

qemu-dm

Interrupt

Virtual PCIe Bus

root

down

up

PCi Express root port

PCi Express upstream port

PCi Express downstream port

PCi Express device
Eventual Goal

- More PCI features/PCI express features
  - The current emulated chipset (I440FX/PIIX3) is too old.
  - So new Chipset emulator is wanted.

- Xen PCI Express support
  - PCI Express native hotplug
  - PCI Express native passthrough
    - When error is detected via AER (Advanced Error Reporting), inject the error into the guest.

- these require several steps, so the first step is...
First Phase Goal

- Make Qemu PCI Express ready
  - Introduce new chipset emulator (Q35)
- PCI Express native hot plug
- Implement PCI Express port emulators, and make it possible to inject errors into guest
## Current status

### Qemu/PCI express

<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCIe MMCONFIG</td>
<td>Merged.</td>
</tr>
<tr>
<td>Q35 chipset base</td>
<td>working</td>
</tr>
<tr>
<td>PCIe portemulator</td>
<td>working</td>
</tr>
<tr>
<td>PCIe native hotplug</td>
<td>working</td>
</tr>
<tr>
<td>PCIe AER</td>
<td>WIP</td>
</tr>
<tr>
<td>PCIe error injection</td>
<td>WIP</td>
</tr>
<tr>
<td>VBE paravirtualization</td>
<td>working</td>
</tr>
</tbody>
</table>

### seabios

<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcfg</td>
<td>working</td>
</tr>
<tr>
<td>e820</td>
<td>working</td>
</tr>
<tr>
<td>host bridge</td>
<td></td>
</tr>
<tr>
<td>initiazatlin</td>
<td>working</td>
</tr>
<tr>
<td>pci io/memory</td>
<td></td>
</tr>
<tr>
<td>space initialization</td>
<td>working</td>
</tr>
<tr>
<td>passing acpi table outside qemu</td>
<td>working</td>
</tr>
</tbody>
</table>

### vgabios

<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBE paravirtualization</td>
<td>working</td>
</tr>
</tbody>
</table>

### Xen

<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rebase qemu base</td>
<td>Not started</td>
</tr>
<tr>
<td>PCIe native passthrough</td>
<td>Not started</td>
</tr>
<tr>
<td>seabios update</td>
<td>Not started</td>
</tr>
</tbody>
</table>

- The qemu/guest firmware enhancement is almost done.
- The next step is qemu upstream merge.
HotPlug functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Supported?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attention Button</td>
<td>yes</td>
</tr>
<tr>
<td>Power Controller</td>
<td>No</td>
</tr>
<tr>
<td>MRL Sensor</td>
<td>No</td>
</tr>
<tr>
<td>Attention Indicator</td>
<td>Yes</td>
</tr>
<tr>
<td>Power Indicator</td>
<td>Yes</td>
</tr>
<tr>
<td>Hot-Plug Surprise</td>
<td>Yes</td>
</tr>
<tr>
<td>EMI</td>
<td>Yes</td>
</tr>
</tbody>
</table>

• Is MRL (Manually Operated Retention Latch) wanted?
• Is EMI (Electro Mechanical Lock) needed?
Usage and example
PCI Express native hotplug

- From qemu monitor command line
- pci_add/pci_del
  - Or device_add/device_del
  - This is same to PCI hot plug.
  - Internally it calls back bus specific function. So it eventually pci express hotplug logic.
- pcie_abp [chassis.]slot
  - push PCI express attention button of a given domain and chassis number.
Error injection

- `pcie_error_inject [[domain:]bus:]dev.fn is_correctable error_status number number number number number [number [number [number [number]]]]]
  - `is_correctable`: bool
    - Correctable or uncorrectable
  - `error_status`: `uint32_t`
    - Specify error type
  - `number`: `uint32_t*4`: TLP header
  - `number`: `uint32_t*{0-4}`: TLP header prefix
Example from Linux boot log

ACPI: RSDP 00000000000f7ae0 00014 (v00 BOCHS )
ACPI: RSDT 000000001ff78f90 00038 (v01 BOCHS BXPCRSDT 00000001 BXPC 00000001)
ACPI: FACP 000000001ffffe70 00074 (v01 BOCHS BXPCFACP 00000001 BXPC 00000001)
ACPI: DSDT 000000001ff78fd0 86C82 (v01 BXPC BXDSDT 00000002 INTL 20100121)
ACPI: FACS 000000001ffffe00 00040
ACPI: SSDT 000000001ffffdc0 00037 (v01 BOCHS BXPCSSDT 00000001 BXPC 00000001)
ACPI: APIC 000000001ffffce0 00072 (v01 BOCHS BXPCAPIC 00000001 BXPC 00000001)
ACPI: HPET 000000001ffffca0 00038 (v01 BOCHS BXPCHPET 00000001 BXPC 00000001)
ACPI: MCFG 000000001ffffc60 0003C (v01 BOCHS BXPCMCFG 00000001 BXPC 00000001)
...
ACPI: bus type pci registered
PCI: MMCONFIG for domain 0000 [bus 00-ff] at [mem 0xe0000000-0xefffffff] (base 0xe0000000)
PCI: MMCONFIG at [mem 0xe0000000-0xefffffff] reserved in E820
pcieport 0000:00:04.0: setting latency timer to 64
pcieport 0000:00:04.0: 0000:00:04.0 lo: 0xfee0100c hi: 0x0 data 0x4129
pcieport 0000:00:04.0: irq 24 for MSI/MSI-X
pcieport 0000:00:04.0: Requesting control of PCIe PME from ACPI BIOS
pcieport 0000:00:04.0: 0000:00:04.0 lo: 0xfee0100c hi: 0x0 data 0x4129
pcieport 0000:00:04.0: Signaling PME through PCIe PME interrupt
pcie_pme 0000:00:04.0:pcie01: service driver pcie_pme loaded
pci_hotplug: PCI Hot Plug PCI Core version: 0.5
acpi_pcihp: acpi_get_hp_hw_control_from_firmware: Trying to get hotplug control for \\SB\_.PCI0
acpi_pcihp: acpi_get_hp_hw_control_from_firmware: Gained control for hotplug HW for pci 0000:00:04.0 (\\SB\_.PCI0)
acpi_pcihp: acpi_get_hp_hw_control_from_firmware: Trying to get hotplug control for \\SB\_.PCI0
acpi_pcihp: acpi_get_hp_hw_control_from_firmware: Gained control for hotplug HW for pci 0000:00:04.0 (\\SB\_.PCI0)
pchiehp 0000:00:04.0:pcie04: Hotplug Controller:
pchiehp 0000:00:04.0:pcie04:   Seg/Bus/Dev/Func/IRQ : 0000:00:04.0 IRQ 24
pchiehp 0000:00:04.0:pcie04:   Vendor ID            : 0x8086
pchiehp 0000:00:04.0:pcie04:   Device ID            : 0x3420
pchiehp 0000:00:04.0:pcie04:   Subsystem ID         : 0x0000
pchiehp 0000:00:04.0:pcie04:   Subsystem Vendor ID  : 0x8086
pchiehp 0000:00:04.0:pcie04:   PCIe Cap offset      : 0x90
pchiehp 0000:00:04.0:pcie04:   PCI resource [7]     : [io 0x1000-0x1fff]
pchiehp 0000:00:04.0:pcie04:   PCI resource [8]     : [mem 0x20000000-0x201fffff]
pchiehp 0000:00:04.0:pcie04:   PCI resource [9]     : [mem 0x20200000-0x203fffff pref]
pchiehp 0000:00:04.0:pcie04: Slot Capabilities      : 0x00020079
pchiehp 0000:00:04.0:pcie04:   Physical Slot Number : 0
pchiehp 0000:00:04.0:pcie04:   Attention Button     : yes
pchiehp 0000:00:04.0:pcie04:   Power Controller     : no
pchiehp 0000:00:04.0:pcie04:   MRL Sensor           : no
pchiehp 0000:00:04.0:pcie04:   Attention Indicator  : yes
pchiehp 0000:00:04.0:pcie04:   Power Indicator      : yes
pchiehp 0000:00:04.0:pcie04:   Hot-Plug Surprise    : yes
pchiehp 0000:00:04.0:pcie04:   EMI Present          : yes
pchiehp 0000:00:04.0:pcie04:   Command Completed    : yes
pchiehp 0000:00:04.0:pcie04:   Slot Status            : 0x0000
pchiehp 0000:00:04.0:pcie04:   Slot Control           : 0x05c0
pchiehp 0000:00:04.0:pcie04: HPC vendor_id 8086 device_id 3420 ss_vid 8086 ss_did 0
pchiehp 0000:00:04.0:pcie04: Registering domain:bus:dev=0000:20:00 sun=0
pci_bus 0000:20: dev 00, created physical slot 0
pci_hotplug: __pci_hp_register: Added slot 0 to the list
pchiehp 0000:00:04.0:pcie04: pcie_isr: intr_loc 10
pchiehp 0000:00:04.0:pcie04: pciehp_get_power_status: SLOTCTRL a8 value read 3f9
pchiehp 0000:00:04.0:pcie04: service driver pchiehp loaded

Enabled debug message via kernel command line
pci_hotplug.debug=1
pci_hotplug.debug_acpi=1
pchiehp.pchiehp_debug=1
pci_slot.debug=1
lspci

# lspci -vt
-

- [0000:00]--+-00.0  Intel Corporation 82G33/G31/P35/P31 Express DRAM Controller
  +-01.0  Cirrus Logic GD 5446
  +-04.0-[20]--
  +-18.0-[21]--
  +-18.1-[22]--
  +-18.2-[23]--
  +-18.3-[24]--
  +-18.4-[25]--
  +-18.5-[26]--
  +-19.0-[36-bf]--+-00.0-[37-47]--+-00.0-[38]--

......

|       |       | +-0e.0-[46]--
|       |       | `-0f.0-[47]--
|       |       | +-00.1-[48-58]--+-00.0-[49]--
|       |       | +-01.0-[4a]--
|       |       | +-02.0-[4b]--

......
00:04.0 PCI bridge: Intel Corporation 5500 Non-Legacy I/O Hub PCI Express Root Port 0 (rev 02) (prog-if 00 [Normal decode])
  Physical Slot: 4
  Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B-
  Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR-
  Latency: 0
  Bus: primary=00, secondary=20, subordinate=20, sec-latency=0
  I/O behind bridge: 00001000-00001fff
  Memory behind bridge: 20000000-201fffff
  Prefetchable memory behind bridge: 20200000-203fffff
  Secondary status: 66MHz- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- <SERR- <PERR-
  BridgeCtl: Parity- SERR- NoISA- VGA- MAbort- >Reset- FastB2B-
  PriDiscTmr- SecDiscTmr- DiscTmrStat- DiscTmrSERREn-
  Capabilities: [40] Subsystem: Intel Corporation Device 0000
  Capabilities: [60] MSI: Enable+ Count=1/2 Maskable+ 64bit-
  Address: 0000100c  Data: 4129
  Masking: 00000003  Pending: 00000000
  Capabilities: [90] Express (v2) Root Port (Slot+), MSI 00
  DevCap: MaxPayload 128 bytes, PhantFunc 0, Latency L0s <64ns, L1 <1us
  ExtTag- RBE- FLReset-
  DevCtl: Report errors: Correctable- Non-Fatal- Fatal- Unsupported-
  RlxdOrd- ExtTag- PhantFunc- AuxPwr- NoSnoop-
      MaxPayload 128 bytes, MaxReadReq 128 bytes
  DevSta: CorrErr- UncorrErr- FatalErr- UnsuppReq- AuxPwr- TransPend-...
(qemu) pci_add 20:0 nic model=e1000
OK domain 0, bus 32, slot 0, function 0

pci 0000:20:00.0: found [8086:100e] class 000200 header type 00
pci 0000:20:00.0: reg 10: [mem 0x00000000-0x0001ffff]
pci 0000:20:00.0: reg 14: [io  0x0000-0x003f]
pci 0000:20:00.0: reg 30: [mem 0x00000000-0x0001ffff pref]
pci 0000:20:00.0: calling quirk_resource_alignment+0x0/0x1b5
pci 0000:20:00.0: calling pci_fixup_transparent_bridge+0x0/0x2a
pci 0000:20:00.0: BAR 0: assigned [mem 0x20000000-0x2001ffff]
pci 0000:20:00.0: BAR 0: set to [mem 0x20000000-0x2001ffff] (PCI address [0x20000000-0x2001ffff]
pci 0000:20:00.0: BAR 6: assigned [mem 0x20200000-0x203fffff pref]
pci 0000:20:00.0: BAR 1: assigned [io  0x1000-0x103f]
pci 0000:20:00.0: BAR 1: set to [io  0x1000-0x103f] (PCI address [0x1000-0x103f]
pcieport 0000:00:04.0: PCI bridge to [bus 20-20]
pcieport 0000:00:04.0: bridge window [io  0x1000-0x103f]
pcieport 0000:00:04.0: bridge window [mem 0x20000000-0x201fffff]
pcieport 0000:00:04.0: bridge window [mem 0x20200000-0x203fffff pref]
pcieport 0000:00:04.0: setting latency timer to 64
pci 0000:20:00.0: no hotplug settings from platform
pci 0000:20:00.0: using default PCI settings
e1000 0000:20:00.0: enabling device (0000 -> 0003)
e1000 0000:20:00.0: PCI INT A -> GSI 18 (level, low) -> IRQ 18
e1000 0000:20:00.0: enabling bus mastering
e1000 0000:20:00.0: setting latency timer to 64
e1000: 0000:20:00.0: e1000_probe: (PCI:33MHz:32-bit) 52:54:00:12:34:57
e1000: 0000:20:00.0: e1000_reset: Hardware Error
e1000: eth1: e1000_probe: Intel(R) PRO/1000 Network Connection
pciehp 0000:00:04.0:pcie04: pcie_isr: intr_loc 8
pciehp 0000:00:04.0:pcie04: Presence/Notify input change
pciehp 0000:00:04.0:pcie04: Card present on Slot(0)
pciehp 0000:00:04.0:pcie04: Surprise Removal
pciehp 0000:00:04.0:pcie04: pcie_isr: intr_loc 10
pciehp 0000:00:04.0:pcie04: pciehp_green_led_blink: SLOTCTRL a8 write cmd 200
pciehp 0000:00:04.0:pcie04: pciehp_check_link_status: lnk_status = 11
pciehp 0000:00:04.0:pcie04: pcie_isr: intr_loc 10
pciehp 0000:00:04.0:pcie04: pciehp_green_led_on: SLOTCTRL a8 write cmd 100
Push attention button

(qemu) pcie_abp 0
OK chassis 0, slot 0

pciehp 0000:00:04.0:pcie04: pcie_isr: intr_loc 1
pciehp 0000:00:04.0:pcie04: Attention button interrupt received
pciehp 0000:00:04.0:pcie04: Button pressed on Slot(0)
pciehp 0000:00:04.0:pcie04: pciehp_get_power_status: SLOTCTRL a8 value read 1f9
pciehp 0000:00:04.0:pcie04: PCI slot #0 - powering off due to button press.
pciehp 0000:00:04.0:pcie04: pcie_isr: intr_loc 10
pciehp 0000:00:04.0:pcie04: pciehp_green_led_blink: SLOTCTRL a8 write cmd 200
pciehp 0000:00:04.0:pcie04: pciehp_set_attention_status: SLOTCTRL a8 write cmd c0
pciehp 0000:00:04.0:pcie04: pcie_isr: intr_loc 10
pciehp 0000:00:04.0:pcie04: Disabling domain:bus:device=0000:20:00
pciehp 0000:00:04.0:pcie04: pciehp_unconfigure_device: domain:bus:dev = 0000:20:00
e1000: eth1: e1000_reset: Hardware Error
e1000 0000:20:00.0: PCI INT A disabled
pciehp 0000:00:04.0:pcie04: pcie_isr: intr_loc 10
pciehp 0000:00:04.0:pcie04: pciehp_green_led_off: SLOTCTRL a8 write cmd 300
Hot unplug

(qemu) pci_del 20:0

pciehp 0000:00:04.0:pcie04: pcie_isr: intr_loc 8
pciehp 0000:00:04.0:pcie04: Presence/Notify input change
pciehp 0000:00:04.0:pcie04: Card not present on Slot(0)
pciehp 0000:00:04.0:pcie04: Surprise Removal
pciehp 0000:00:04.0:pcie04: Disabling domain:bus:device=0000:20:00
pciehp 0000:00:04.0:pcie04: pciehp_unconfigure_device: domain:bus:dev = 0000:20:00
pciehp 0000:00:04.0:pcie04: pcie_isr: intr_loc 10
pciehp 0000:00:04.0:pcie04: pciehp_green_led_off: SLOTCTRL a8 write cmd 300

From /proc/interrupts

24: 13 PCI-MSI-edge PCIe PME, pciehp
Some notes on Linux

- Linux doesn't touch electro mechanical interlock (EMI).
- Linux pcie aer doesn't log TLP header prefix.
Implementation Details
Q35 based new chipset emulator

• Why new chipset?
  – Keep the currently supported chipset (I440FX/PIIX3) for legacy compatibility.
  – Add new features for modern OSes without legacy compatibility.

• Many clean ups to coexist with the existing chipset (mch/ich9)
  – Factor out i440 specific code to coexist.
PCI Express port emulator

- Root port/upstream port/downstream port
  - All of three ports are needed.
- At first, PCI bridge code had to be cleaned up.
  - It was just a stub, had to implement it first.
  - PCI bus numbering paravirtualization
SeaBIOS modifications

- Multi chipset support
  - factor out i440 specific code
- PCI Bus initialization
  - Bus numbering paravirtualization
- 64bit BAR
- MCFG
- Passing DSDT from qemu command line to guest bios
VGABios VBE paravirtualization

- Currently qemu mapes VBE base address to the hard coded address.
- Which conflicts MMCONFIG area.
- Make it dynamic by paravirtualization
- However I did it differently from boch one.
  - I knew bochs work after creating the patch
Future Work
Future Work

- Complete error injection
- Qemu Upstream Merge
- Xen support
  - rebasing/bios change
  - PCI Express native passthrough
  - AER injection
Qemu Release Plan

• The next major release 0.13
  – Planned target: June 1\textsuperscript{st}
    – http://www.mail-archive.com/qemu-devel@nongnu.org/msg22485.html

• Planned features
  – http://wiki.qemu.org/Features/0.13
Call to action

- Help for upstream merge.
- Rebase qemu-dm to the upstream qemu
- Switch guest bios from bochs bios to seabios
Considerations on further development issues
Qemu PCI Express

- There are no interesting emulated PCI Express native devices at the moment.
  - PCIe switch port isn't interesting.
- Any good candidate?
  - igb, igbvf
  - ixgb, ixgbvf
Qemu AER emulation and error injection

- Is multiple header recording wanted?
  - Maybe yes.

- Is multiple errors injection wanted
  - Maybe this is wanted because Linux aer error injection tool supports it.
PCI Express native passthrough

- Xen qemu-dm rebasing
  - Back porting patches is impractical.
    - They heavily depends on new qemu device framework. (QDev)
- PCI Express native passthrough
  - Enhancing the configuration space access is trivial.
  - Necessary to virtualize extended capabilities according to the current framework.
  - root/downstream port should also be passthroughed?
- Multifunction
  - It is difficult to make non-ARI function have ARI.
AER Error injection into guest

- Capturing AER error
- Error injection
  - TLP header mapping (address, device id)
  - Bus topology mapping
- Error recovery and reset
  - Bus topology mapping
Error capturing

- Root ports reports all the errors generated by all the device under the bus
- The current PCIe port service driver of AER captures all errors.
- Enhance PCIe port service driver of AER to capture wanted errors.
- Multiple errors can occur
Error Injection

- TLP header conversion (address, device id)
- Error can occur intermediate ports.
  - PCIe bus topology mapping isn't trivial
- Multiple error injection.

Where to inject?

Interrupt

Error message
Recovery and device/bus reset

- PCIe bus can be shared by domains.
  - Or PCIe functions are passed through to different domUs
- Error can occur intermediate ports
- OS would reset those ports as error recovery
  - Which real device to reset?
- Bus reset is propagated to all the under devices
- What should do other domains?
  - Notify reset?

interrupt
packet
reset

 PCI Express root port
 PCI Express upstream port
 PCI Express downstream port
 PCI Express device

Domain
Domain
Domain

passsthrough or dom0 uses devices
...
Summary

- **Enhanced Qemu**
  - New chipset emulator (mch/ich9)
  - PCI enhancement
  - PCI express
    - MMCONFIG
    - Native hot plug
    - Error injection
- **SeaBIOS enhancement**
- **VGABIOS modification**
Thank you

Questions?