Enhancing pass through device support with IOMMU

Haitao Shan (haitao.shan@intel.com)
Yunhong Jiang
Allen M Kay
Eddie (Yaozu) Dong

Agenda

- Current Status
- Further enhancement
  - Hardening host
  - Improving functionality
  - Handling more corner cases
- Call to action

Current Status

Pass through device support with IOMMU is enabled in Xen since 3.2.0

- Full functional
- High throughput, minimal CPU utilization

Further enhancement

Further enhancements needed for error handling, more devices, hot add / removal etc.

- Hardening host from device failure
- Improving functionality
- Handling more corner cases

Hardening host from device failure

A guest device error may be propagated to host

- **PCI SERR# from guest assigned device may result in host platform reset.**
  - **SERR# can generate an SMI or NMI and host BIOS/OS may reset the physical platform**

Proposed solution

- **Need PCIe AER to recover from device error**
  - **Turn on AER for underlying PCIe device.**
    - Virtualize guest CMD access by pinning host SERR to 1
- **AER support in Linux is only after 2.6.19**
  - **Push for dom0 pv_ops!!!**

Improving functionality: Standardizing CFGS emulation

Current CFGS emulation policy

- Only support offset 0-256, 256-4096 are not in current Qemu
- Pass thru CMD only, virtualize BAR/MSI/MSI-X, writes to the rest are ignored, but read from physical
  - Device may not fully function since if driver's setting are ignored

Proposed solution

- Pass thru all registers except the ones with known behavior
  - Framework to provide additional filters (in QEMU) for certain capabilities
- Standardized CFGS emulation policy helps device vendors to implement virtualization friendly solutions

CFGS Emulation Summary

Register (s)	Current Policy	Proposed Policy
CMD	Pass-thru	Pass-thru with SERR pin to 1
BARs and Expansion Rom Base Address	Virtualized	Virtualized
CardBUS CIS pointer	Virtualized as RAM	Pass-thru
Interrupt Line (RW) (device does not use)	Virtualized as RAM	Virtualized as RAM
Interrupt Pin (RO)	Virtualized as RAM	Present base on virtual platform
Rest registers in PCI header	Virtualized as RAM	Pass-thru
MSI/MSI-x	Virtualized	Virtualized
Registers outside configuration space header	Write ignored, Read from physical device	Pass-thru

Handling more corner cases: Device reconfiguration

Xen uses initial BIOS PCI settings to build internal device list, and IOMMU page tables for dom0

- But PCIe devices can be reconfigured later on and even at rum-time
  - Dom0 may re-balance (pci=assign-busses) at its own initialization stage → Bus # changed
  - Devices can be dynamically added (VFs in SR-IOV and hot-plugged devices)

Proposed solution

- Add new hypercalls to instruct Xen to update IOMMU mappings
- Allow device list in Xen to be dynamically modified

Should the device list and IOMMU table be constructed in Xen startup stage or later on after dom0 instructs Xen?

Handling more corner cases: Qemu for PCIe Devices

Virtual chipset (PIIX3) in Qemu doesn't support PCIe

- Guest see PCIe devices in PCI chipset without bridge
- Qemu doesn't support PCIe extended capabilities
  - Device may not work appropriately if its setting are not passed through

Possible solution

- Enable PCIe extended capabilities in virtual platform
- Pushing Qemu/BIOS ahead to incorporate more advanced features and more recent chipset model

Call to action

Need community efforts

- Pushing dom0 to latest Linux tree
  - Any update of pv_ops support for dom0 ?
- AER enabling
- Device power management

Legal Information


intel
Leap ahead™