VGA Assignment Using VFIO

Alex Williamson
alex.williamson@redhat.com
October 21st, 2013
Agenda

- Introduction to
  - PCI & PCIe
  - IOMMUs
  - VFIO
- VGA
- VFIO VGA support
- Quirks, quirks, quirks
- Status and future
- Performance
A brief look at PCI

- **Peripheral Component Interconnect**
- Initially developed by Intel, 1992
- Replaced VLB, MCA, EISA, and ISA
- Hierarchical, self describing, bus-based architecture
- Buses provide 32 device slots, 8 functions per device
- Each function has 256 bytes of configuration space
  - Allows device discovery
  - Describes device and features
  - Programmable base address registers (BARs)
- Single interrupt line (INTx) evolved to MSI/MSI-X
PCI-Express improvements

- Software compatible with conventional PCI
- No longer a shared bus
  - Point-to-point interface
- Requester IDs for transaction routing
  - Similar to a network
- Additional configuration space (4k)
IOMMUs

• Initially used to provide DMA *translation* only
  • Allowed 32bit devices to avoid bounce buffers
  • Popularized on x86 with AMD GART
• Fine granularity *isolation* support added with PCIe
  • Enabled by Requester IDs
  • AMD-Vi & Intel VT-d
  • Devices are confined and operate within their own address space
A platform for VM device assignment

- Discoverable
- Self describing
- Self contained
- Fully programmable
- Isolated
- DMA translation
- Minimal configuration space emulation required
Introduction to VFIO

- **Virtual Function I/O**
- Linux userspace driver infrastructure
- Enforces IOMMU protection
- Provides:
  - Device access
  - IOMMU programming interface
  - High performance interrupt support
VFIO for device assignment

- Guests are userspace drivers
- VFIO decomposes the device to userspace
- QEMU recomposes the device to PCI for guest
- KVM provides acceleration
VFIO & PCI

- VFIO abstracts devices as Regions and IRQs
- Regions include:
  - PCI configuration space
  - MMIO and I/O port BAR spaces
  - MMIO PCI ROM access
- IRQs include:
  - INTx (legacy interrupts)
  - Message Signaled Interrupts (MSI & MSI-X)
VFIO Device Decomposition

Memory space

I/O port space

I/O port 4

Physical Device

VFIO Device File Descriptor

VFIO
QEMU & VFIO

- QEMU creates guest mappings to each region
  - PCI config region mapped to guest Bus/Slot/Func
    - Some emulation in QEMU, some in VFIO
  - BARs & ROM mapped to emulated BAR address
    - `in/out` or `read/write` to BAR mapping results in `read/write` to VFIO device file descriptor
    - `mmap` support for sufficiently sized MMIO BARs
- VFIO signals interrupts via `eventfd`
  - QEMU receives interrupts via `poll(2)`
  - KVM irqfd support allows userspace bypass
QEMU Device Re-composition

VFIO
Device File Descriptor

I/O port 4
MMIO3
MMIO2
MMIO1

Guest
Memory
space

Guest
I/O port
space

I/O port 4

Virtual
Device

QEMU

Guest
PCI config
space

Device File Descriptor

Virtual Device
Great, can I assign my graphics card yet?

- **Video Graphics Array**
- Introduced on IBM PS/2 around 1987
- Age of fixed address I/O devices and 20-bit addresses
- Relies on fixed, system-wide addresses
  - MMIO: `0xa_0000–0xb_ffff`
  - I/O port: `0x3b0–0x3bb, 0x3c0–0x3df`
- VBIOS provides:
  - Device initialization
  - Runtime services: int10 & VESA
Fixed VGA resources vs PCI

- VGA resources are not exposed through PCI BARs
  - Assumed based on PCI class code
- Routing of VGA addresses controlled by:
  - I/O port & MMIO enabled bits on devices
  - VGA Enable bit on PCI bridges
  - Chipset specific registers
- VGA routed to a single “bus”
  - Single device per bus on PCIe
- Shared resources require arbitration
VGA Arbiter

- Kernel software construct
- Allows locking of VGA resources
  - Acquire lock before access
  - Release lock after access
  - VGA routing included with lock
- Cooperative usage model
VFIO & VGA

- Provide a new VFIO “region” for accessing VGA
- Wrap guest VGA access in VGA arbiter callbacks
  - Slow, but functional
  - Assume guest won't use VGA long term
- Problem solved! Right?
VFIO VGA

- Memory space
- I/O port space
- Physical Device

VFIO VGA

- PCI config space
- VGA Arbiter
- VFIO

VFIO Device File Descriptor
If only...

- VBIOS uses “back doors” to access PCI regions
  - Physical BAR addresses hidden in VGA space
  - PCI config space access through MMIO BARs
  - I/O port BARs provide windows to MMIO BARs
- Try to access Host addresses
  - Device is still isolated
  - But it doesn't work
VGA bootstrap example

- AMD/ATI Radeon has a PCI I/O port region at BAR4
- The VBIOS discovers the 2\textsuperscript{nd} byte of the address of BAR4 through the byte at VGA I/O port 0x3c3
- I/O port BAR4 provides an address and data window register access to PCI MMIO BAR2
- At offset 0x4000 into BAR2, PCI configuration space for the device is available
Another example

- Nvidia cards have an I/O port region at BAR5
- Provides a data and address window to all the other MMIO BARs
- BAR0 provides access to 256 bytes of conventional PCI config space at offset 0x1800 and the full 4k of PCIe config space at offset 0x88000
- Envytools documentation suggests other offsets provide access to PCI configuration space of the companion audio device and parent PCI bridge
How do we handle this?

- Possible solution: Identity map graphics cards
  - Imposes difficult restrictions on guest memory map
- Possible solution: Quirks
  - QEMU Memory API allows for fine granularity handling of sub-ranges for devices
  - Quirks are device specific MemoryRegions for handling these back doors
  - Hopefully not performance paths
  - Hopefully static
Identifying quirks

- It's hard... but not as hard as it seems
- QEMU VFIO has very good logging
- Unassigned memory access logging is useful
- Be careful of BARs that randomly get 1:1 mapped
  - It will work for you and not others
  - Configuration dependent
Status

- Making progress
- Examples of working Radeon and Geforce cards
- Ongoing improvements:
  - Linux v3.12
    - Includes PCI bus & slot reset interface
  - QEMU 1.7
    - VFIO co-assigned device reset
    - Better ROM handling
    - Coherency/NoSnoop fixes (Windows Code 43 fix)
VGA vs GPU

- VGA is not the only way to assign a graphics card
- Configure guest with emulated VGA + secondary assigned GPU
  - Just another assigned PCI device
- Proprietary drivers often required to initialize device
- May or may not need quirks
- No VGA arbiter dependencies, no VGA routing issues
- Actually some vendor support for this
  - Nvidia: Quadro, GRID, and Tesla
Future

- Close to having discrete AMD/Nvidia support
- Some remaining driver issues
  - Not all host drivers unload cleanly
  - Nvidia driver hogs VGA arbiter lock
  - i915 VGA arbiter support is broken
  - Uncertain future for VGA arbiter
  - Basic VGA console drivers don't use VGA arbiter
- IGD Assignment
  - Work in progress by Andy Barnes
  - Not self contained like discrete graphics
Performance

- 95%+ for accelerated drivers
- Poor VGA performance
  - Abysmal if contended
- Bootstrap only
- Tune for virtualization
- INTx vs MSI
  - NVreg_EnableMSI=1
  - GeForce vs Quadro
- Configuration space polling
  - NVreg_CheckPCIConfigSpace=0
Relative Windows 7 VM Graphics Performance

Intel Core i5 4430

Nvidia GTX 660
Radeon HD 7850
Nvidia K4000

- Gaming
- Windows 7 Experience
- Desktop
- 3DMark 11
- Basic
- Unigine Heaven 4.0
- Extreme
Thanks