288 vCPUs Support On Xen

Chao Gao
chao.gao@intel.com

Goals And Current Status

Goals

- Bump up the maximum number of vCPUs of a HVM guest to 288
  - To improve the computing power of one VM
- Make guest CPU/cache topology configurable
  - Intel® Xeon Phi™ processor has up to 72 cores. Each core has 4 logical processors.
  - Proper configuration greatly benefits VM's performance

Current Status

- Xen supports up to 128 vCPUs in a HVM guest
- Guest CPU/cache topology isn't configurable

A Virtual Intel® VT-d In Xen

- Emulated by Xen rather than by qemu
  - PVH guest may use this emulated VT-d
  - Injecting interrupts of pass-through devices doesn’t involve qemu
- For simplicity, not all features are emulated
  - Interrupt Remapping (IR), Queue Invalidation (QI) and fault report
  - DMA remapping isn’t emulated.

How Virtual VT-d Works

1. Libacpi builds DMAR for virtual VT-d

2. Tool Stack creates virtual IOMMU in Xen via domctl hypercall
   a) Pass the base address and length of register region to Xen
   b) Xen traps accesses to that region.

3. Guest parses ACPI DMAR and initialize virtual VT-d
   a) Set up Interrupt Remapping Table (IRT) in guest memory, and enable IR, QI and Fault report.
   b) Program Interrupt Remapping Table Entries (IRTEs) when programming vIOAPIC and vMSI.
      - Guest OS needs to invalidate Interrupt Entry Cache (IEC) via QI.

How Virtual VT-d Works (continued)

4. Xen emulates accesses to the register region of virtual VT-d
   a) Set up IRT: map IRT to Xen.
   b) Enable IR: start to hook delivery of guest interrupts
   c) Queue invalidation request: flush in-flight interrupts and then notify guest the completion.

5. Virtual VT-d handles guest interrupts per IRTE
   a) Interrupts of virtual devices injected by qemu
   b) Interrupts from pass-through devices

Other Changes For Virtual VT-d

- Extend a hypercall to bind guest interrupts of pass-through device with physical interrupts
  - The existing hypercall accepts a filtered interrupt request.
    - The fields filtered out are reserved or unalterable for Compatibility Format Interrupt Request
    - However, they are meaningful to Remappable Format Interrupt Request
  - Introduce a new type to indicate interrupt requests passed down is unfiltered.

- A new hypercall to manage vIOMMU
  - At present, only support vIOMMU creation
  - Can be extended for other vIOMMUs, like AMD IOMMU.

- Migration support

Other Changes For 288 vCPUs Support

- Declaring processors in ACPI
  - Declare x2apic structure in MADT, x2apic affinity structure in SRAT and etc.
  - Declare processors in DSDT
    - ASL Processor statement (deprecated in ACPI 6.2, using this method for CPU with APIC ID <= 254 for compatibility)
    - ASL Device statement (for APIC ID > 254, added in ACPI 3.0)

- Add multiple IOREQ pages support
  - Currently, only support one IOREQ page, supporting up to 128 vCPUs.
  - Paul Durrant's "mapping guest resources" (merged) simplifies the changes needed.
  - Qemu computes the number of IOREQ page needed and requests Xen to set up for these IOREQ pages.

Other Changes For 288 vCPUs Support (continued)

- Switch vCPUs to x2APIC mode before passing control to guest OS
  - On bare-metal, BIOS do this, if any CPU reports an APIC ID of 254 or greater.
  - Will be done in hvmloader.

- Make guest CPU/cache topology configurable
  - Introduce several options in a domain configuration file, i.e. #core, and #thread in each core
  - Tool Stack computes return value of CPUID 0x4 and CPUID 0xb and sets them into Xen when applying a CPUID policy to the guest

Thank you!

Backup

Two interrupt request formats

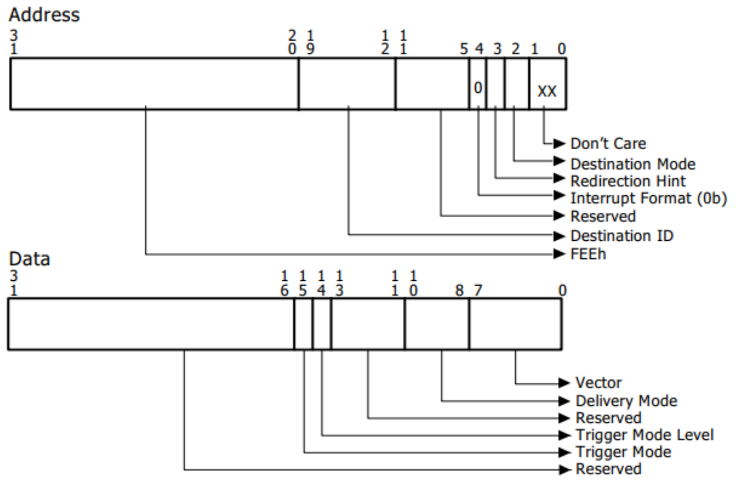

Figure 5-14. Compatibility Format Interrupt Request

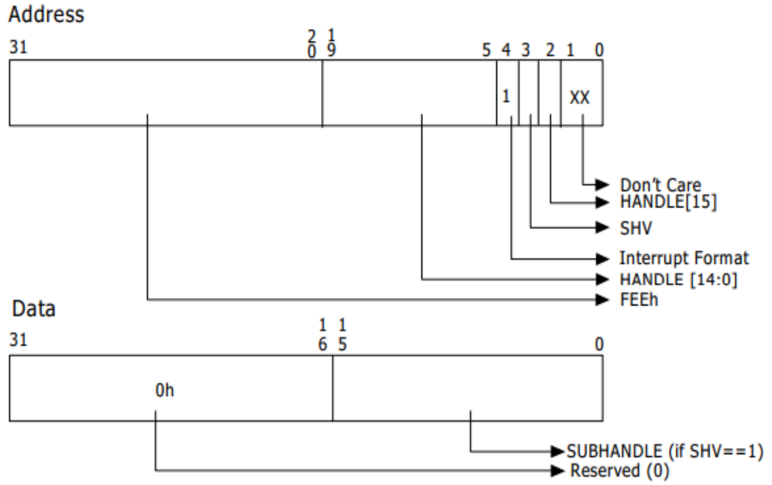

Figure 5-15. Remappable Format Interrupt Request