Xen/IOMMU

Breaking IO in New and Interesting Ways

Muli Ben-Yehuda, Jon D. Mason, Orran Krieger, Jimi Xenidis

`muli@il.ibm.com, {jdmason,okrieg}@us.ibm.com, jimix@watson.ibm.com`

IBM

Table of Contents

Background/Motivation

I/O Memory Management Units are HW devices that translate addresses from “I/O space” to “machine space”.

- Bounce buffer avoidance - 32bit DMA capable, non-DAC, devices can access physical memory addresses higher than 4GB.
- OS RAS - prohibiting devices from DMA’ing into the wrong area of the OS’s memory.
- Domain isolation - prohibiting devices from DMA’ing into another domain’s memory, e.g., **untrusted** OS with direct device access.
- Improved I/O performance - programming IOMMUs so that the memory region appears to be contiguous to the device (aka scatter-gather coalescing or I/O merging).

Background/Motivation Cont’

- IOMMUs usually reside on a PHB or in the northbridge and translate DMA addresses when devices DMA to and from memory.
- IOMMUs exist on all major HW architectures.
- Utilizing the IOMMU might require platform (i.e., BIOS) support and definitely requires hypervisor and operating system support.
- Self-virtualizing devices (e.g., Infiniband HCAs) can be considered IOMMUs as well.
- IOMMU data structure divergence: format, size, internal structure (e.g., hierarchical page tables or hash table).
- IOMMU isolation capabilities, e.g., no device isolation (single shared I/O address space), isolation per bus, isolation per BDF, etc.

Current Linux and Xen support

- Linux:
  - The DMA-API abstracts the existence and details of the IOMMU from driver writers. Typically only the DMA-API implementation is aware of the existence and details of programming the IOMMU.
  - The implementation of the DMA-API is architecture specific; for x86-64 there are multiple implementations (gart, swiotlb, nommu), selected at compile time.
- Xen:
  - The Xen unstable tree does not support any HW IOMMUs.
  - swiotlb and grant tables can be considered SW implementations of IOMMU functionality.

Example IOMMU: IBM TCEs

- IBM’s Translation Control Entries (TCE) provide functionality to translate and isolate.
- Translation table entirely different from CPU page table: re-use, cachability...: data structures must be kept consistent
- Large page support (currently same as processor).
- Have shared I/O space (I/O addressability and protection independent): hypervisor must control.
- Currently supported in IBM’s pSeries servers, HW exists in IBM’s xSeries Summit chipsets, OS and hypervisor support are WIP.
- Protection depends on machine configuration (bus/slot).

Plan/Status

- Enable run time selection of IOMMU: AMD, Intel, IBM TCE, Xen
- Support IBM TCE in Linux x86-64 (isolation capable x86 IOMMU available in production systems)
- Define simple interface IOMMU support in Xen:
  - dedicating device to untrusted domain
  - driver domains
- Preliminary implementation targeting IBM TCE
- Integrate IOMMU implementations for AMD and Intel
- Integration interface/implementation with Grant tables & SWIOTLB
- Define responsibilities Dom0/Xen (e.g., I/O space creation/management)

Issues

- Who creates/destroys I/O spaces?
- We assume that the hypervisor directly updates.
- What is the right PV interface?
- Implementation issues in Xen: e.g., pinning, garbage collection on I/O device removal...
- Grant table convergence: Interface and implementation similarities, but differences, e.g., hypervisor must allocate I/O address

Issues Continued

- Do we expose granularity of isolation, or virtualize at finest granularity possible?
- Support for large pages in the IOMMU.
- Can we make domains with direct device access migratable?
- How do we support fully virtualized legacy OS's.
- Support for fully virtualized IOMMU aware OSes.