Shared Virtual Memory (SVM) in Xen

Feng Wu
feng.wu@intel.com
Agenda

• Motivation
• Now and Future
• SVM in Hardware
• Why Xen needs SVM
• What needs to do in Xen
• Summary
Motivation

• OpenCL 2.0 supports sharing virtual address between CPU and GPU
• Reduce communication latency between CPU and GPU
  ✓ Data sharing
  ✓ Zero-copy is possible
• Simplified GPU programming
  ✓ Far fewer interfaces to use
  ✓ Simple pointer sharing
What we have now

CPU

MMU

GPU

GTT
(Graphics Translation table)

CPU virtual address space

GPU virtual address space

system memory
allocated for GPU
What we get after SVM

- CPU
- MMU
- GPU
- VT-d (First-level page table)

Unified virtual address space

- System memory
- Allocated for GPU
SVM in Hardware

- Concepts:
  - SVM
    - Shared Virtual Memory
  - PASID
    - Process Address Space ID
    - New extension to PCI-e Spec.
    - Identify the targeted virtual address space
  - DMA Requests without PASID
    - Normal memory requests from endpoint devices
  - DMA Requests with PASID
    - DMA request with application’s virtual address

- Extensions to hardware
  - Extended Root Entry
    - Upper 64-bit is used
    - LCTP: Lower Context Table Pointer
    - UCTP: Upper Context Table Pointer
  - Extended Context Entry
    - Extended to 256 bits
    - PASIDPTR: PASID table pointer
    - SLPTPTR: Second-Level Page Translation Pointer
  - PASID table
    - FLPTPTR: First-Level Page Translation Pointer
First-level Translation in Extended Context

- Used for translating requests with PASID (VA->PA)
Second-level Translation in Extended Context

- Translating requests without PASID (IOVA->PA)
Second-level Translation in Extended Context - Cont’d

- Translation requests with PASID in nested mode (GPA -> HPA)
  - Enabled through setting NESTED bit in extended context.
  - Translation requests with PASID
  - IPA: Intermediate Physical address

```
Ext. Root Entry 0
  ...                          Ext. Context Entry 0
  ...                          Ext. Context Entry 127
  ...                          ...
  Ext. Context Entry 0          Requests with PASID
VA->IPA                        Nested Translation
First Level
  IPA->PA                      Second Level

RTA Register
```
Recoverable Address Translation Faults

Service
Page request faults

CPU

Page Request sent to CPU

Page Request to Remapping Hardware

VT-d Hardware with SVM

Translation Request

Recoverable Fault

Page Request

Request Response to device

Devices with PRS* and ATS* Support

Once success, replay original translation request

Page request buffer

IQ

Time axis

* PRS: PCI-Express Page Request Services (PRS) Capability
* ATS: PCI-Express Address Translation Services (ATS) Capability
Why do we need SVM in Xen

- New iGFX devices can work in SVM mode
- Need to support iGFX pass-through for SVM enabled devices
SVM architecture in Xen

Qemu/hvmloader
- Virtual VT-d w/ SVM Ext.

Guest SVM
- VT-d w/ SVM Ext.
- Ext Context
- Ext Root
- RTA R
- PASID Table
- GVA → GPA
- 1st level translation
  (i.e. ia32-e page tables)

Host SVM
- VT-d/SVM engine

Host shadow Root/Context
- Ext Context
- Ext Root
- RTAR
- VT-d w/ SVM Ext.

Guest address translation fault handling
- GPA → HPA
- 2nd level translation
  (i.e. VT-d page tables)
Virtual VT-d

Key capabilities exposed to guests:

• CM (Cache Mode) = 1
  ✓ Any software updates to any remapping structures requires explicit invalidation of the caches
    ➢ updates to not-present entries
    ➢ Updates to present entries whose programming resulted in translation faults
  ✓ Used for shadow root/context table (described in the next slide)

• PASID
  ✓ With PASID capability, the device can perform DMA on the application’s virtual address.

• PRS (Page Request Service) and QI (Queue Invalidation)
  ✓ Used to handle recoverable address translation faults
VT-d page table virtualization

- Guest QI operations
  - Interception
  - Parsing
- Guest Root/Context -> Host Root/Context
  - Directly copy for most fields
  - Some special fields:
    - SLPTPTR: point to host GPA-to-HPA table
    - NESTE: nested translation enabled
Non-recoverable fault handling

Two solutions:

- No emulation
  ✓ Reset the device in host side
  ✓ Detach device and re-attach device to guest

- Emulation
  ✓ VMM emulates it in the virtual VT-d registers
  ✓ Notify the guests via vMSI
Recoverable fault handling

Guest
- Guest recoverable fault handler

Host
- QI
- vMSI

Guest IQ
- w/ PASID
- Virtual VT-d w/ SVM

Guest PRQ
- w/o PASID

Host IQ
- VT-d w/ SVM

Host PRQ
- w/o PASID

Host recoverable fault handler

Replay
- Request response
- Page request
- Recoverable fault

Gfx PCIe end-point device
Summary

• Share page table between CPU and GPU
• DMA request with/without PASID
• VT-d First-Level page table
• Virtual VT-d
• Shadow extended root/context table
• Non-Recoverable/Recoverable translation fault
Thank you!