Shared Virtual Addressing in KVM

Yi Liu (yi.l.liu@intel.com)
Jacob Pan (jacob.jun.pan@intel.com)
KVM Forum 2018
No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others

© Intel Corporation.
Shared Virtual Addressing (SVA)

- CPUs
- MMU
- Memory
- Root Complex
- IOMMU
- OS Managed
- CPU page tables
- IOMMU page tables
- Non-SVA capable devices (E.g.: Legacy Devices)
- Discrete Devices (E.g.: PCI-E attached devices)
- Integrated Devices (E.g.: Processor Graphics)
- Host/Physical Memory
- CPU
- VA
- IOVA
- Device
- PA
- PA
Shared Virtual Addressing (SVA)

Previously called "Shared Virtual Memory"
SVA on Intel® VT-d

• Process Address Space ID (PASID)
  ➢ Identify process address space

• First-level/Second-level translation
  ➢ Supports different usages (IOVA/SVA) by different translation types

• Translation Types
  ➢ First-Level translation
  ➢ Second-Level translation
  ➢ Nested translation
  ➢ Pass-Through (address translation bypassed)

• Intel® VT-d 3.0 introduced Scalable Mode
  ➢ SVA and Intel® Scalable I/O Virtualization technology are orthotropic
SVA on Intel® VT-d (Cont.)

• Nested Translation
  - Use both first-level and second-level for address translation
  - Enable SVA in virtualization environment
    - First-level: GVA->GPA
    - Second-level: GPA->HPA

• Most vendor supports nested translation for SVA usage in Virtual Machine
vSVA on Intel® VT-d

- CPUs
  - MMU
  - Root Complex
    - IOMMU
    - Non-SVA capable devices (E.g.: Legacy Devices)
    - Discrete Devices (E.g.: PCI-E attached devices)
    - Integrated Devices (E.g.: Processor Graphics)

  - memory

- Device
  - CPU
  - GVA
  - EPT tables
    - OS Managed
    - VMM Managed
  - VT-d tables
  - GPA
  - Host/Physical Memory
  - HPA

- Host/Physical Memory
  - Managed

- Non-SVA capable devices
  (E.g.: Legacy Devices)
- Discrete Devices
  (E.g.: PCI-E attached devices)
- Integrated Devices
  (E.g.: Processor Graphics)
Enable SVA in VM

• Need a virtual IOMMU with SVA capability
  ➢ Proper emulation according to IOMMU spec (e.g. Intel® VT-d specification)
    • either fully-emulated or virtio-based IOMMU

• Notification for guest translation structure modifications
  ➢ Notification mechanism is vendor specific
  ➢ For Intel® VT-d
    • “caching-mode”: explicit cache invalidation is required for any translation structure change in software

• Enable nested translation on physical IOMMU for given PASID
SVA Architecture in KVM

- **Qemu**
  - vIOMMU emulation is in Qemu

- **VFIO: Virtual Function I/O**
  - Program host IOMMU via VFIO

- **IOMMU driver**
  - New APIs exposed to VFIO for guest SVA
SVA Architecture in KVM (Cont.)

- Bind PASID
  - VT-d: guest CPU page table
• Bind PASID
  ➢ VT-d: guest CPU page table

• Forward guest CPU page table cache invalidation to host
SVA Architecture in KVM (Cont.)

- **Bind PASID**
  - VT-d: guest CPU page table

- **Forward guest CPU page table cache invalidation to host**

- **Page fault reporting and servicing**
SVA Architecture in KVM (Cont.)

- Bind PASID
  - VT-d: guest CPU page table

- Forward guest CPU page table cache invalidation to host

- Page fault reporting and servicing
SVA Architecture in KVM (Cont.)

- **Bind PASID**
  - VT-d: guest CPU page table

- **Forward guest CPU page table cache invalidation to host**

- **Page fault reporting and servicing**

Neutral Kernel APIs for both emulated and virtio-based vIOMMUs
Changes to Qemu/VFIO/IOMMU

- **Qemu**
  - Vendor specific vIOMMU emulation
  - Capture guest IOMMU translation modifications and program host IOMMU via VFIO IOCTL

- **VFIO: Virtual Function I/O**
  - New IOCTL will be introduced:
    - `VFIO_IOMMU_BIND_PROCESS`  
      Binding to host CPU page table
    - `VFIO_IOMMU_BIND_GUEST_PGTBL`  
      Binding to guest CPU page table
    - `VFIO_IOMMU_BIND_GUEST_PASID_TBL`  
      Binding to guest PASID Table
    - `VFIO_IOMMU_SVA_INVALIDATE`  
      Invalidate tlb for guest
    - `VFIO_DEVICE_DMA_FAULT_FD_SET`  
      Set fault eventfd for notifying userspace (Qemu)
    - `VFIO_DEVICE_GET_DMA_FAULT_INFO`  
      Get dma fault info to userspace (Qemu)

- **IOMMU driver**
  - Jacob will introduce detail on it
• Qemu vSVA enabling has two parts
  ➢ vIOMMU emulation
    ▪ Earliest RFC patch for vSVA back to 2017-April
  ➢ Notification framework between vIOMMU device-model and VFIO within Qemu
    ▪ Yi Liu: Notifier framework in v3, proposed PCISVAOps for communication between vIOMMU emulator and VFIO
    ▪ Eric Auger: vSMMUv3/pSMMUv3 2 stage VFIO integration v2
    ▪ Shares the notification framework work

• TODO:
  ➢ consolidate the common part between different tracks
  ➢ Hardware IOMMU capability query interface
    ▪ vIOMMU should not report capabilities with no host support if VM has assigned devices
Upstream Status (Kernel)

• IOMMU/VFIO extension for virtual SVA
  - Earliest RFC patch for vSVA support
  - Reuse and extend the above IOMMU API with ARM SMMU support by Eric Auger ([https://lkml.org/lkml/2018/9/18/1087](https://lkml.org/lkml/2018/9/18/1087))

• Native SVA support

• Shared requirements in the two tracks
  - binding PASID, fault reporting

• Dependent changes
  - VT-d v3 support by Lu, Baolu ([https://lkml.org/lkml/2018/10/7/54](https://lkml.org/lkml/2018/10/7/54))
Terminology puzzle

VT-d
PASID
PASID table
1st & 2nd level translation
Device context table
PCI requesterID

SMMU
SubstreamID
Context descriptor table
Stage 1 & 2 translation
Stream table (entry)
PCI requesterID maps to StreamID
# A tale of two SVAs

<table>
<thead>
<tr>
<th>Key differentiation Features</th>
<th>Intel VT-d v3</th>
<th>ARM SMMUv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guest PASID allocation</td>
<td>Allocated by host system wide via virtual command interface</td>
<td>Allocated by guest in its own space</td>
</tr>
<tr>
<td>Device PASID table</td>
<td>Managed by host, shadowed</td>
<td>Managed solely by guest</td>
</tr>
<tr>
<td>GPA-HPA translation</td>
<td>2nd level per PASID</td>
<td>Stage 2, shared by all PASIDs per streamID</td>
</tr>
<tr>
<td>PASID 0</td>
<td>Available for allocation if RID2PASID is not enabled</td>
<td>Reserved for request w/o PASID</td>
</tr>
<tr>
<td>Page request/response</td>
<td>Has private data, needs page response w/o last page in group (LPIG)</td>
<td>Support non-PCI and PCI PRI-like stall model, no dependency on ATS**</td>
</tr>
<tr>
<td>IOMMU domains</td>
<td>Does not support default domain with DMA API*</td>
<td>Supports default DMA domain, can switch in/out default domain</td>
</tr>
</tbody>
</table>

* In progress to align with other IOMMU drivers
** All stall faults need response, faults contain more info such as which stage
IOMMU SVA API development

Common

- PASID
- PASID management
- IOMMU dev Fault

Guest Shared Virtual Addressing
- Set PASID table
- Cache invalidation
- Bind/unbind PASID (guest mm)
- Bind guest MSI

Native Shared Virtual Addressing
- Init/shutdown SVA device
- Bind/unbind PASID mm
- IO page fault

Color coding of patchset

- vSVA Intel VT-d 3
- Native SVA ARM sMMU3
- vSMMUv3/VFIO 2 stage integration

VFIO
Kernel driver
## IOMMU API extensions proposed

<table>
<thead>
<tr>
<th>API</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>iommu_set_pasid_table</td>
<td>Guest owns PASID table. PASID managed by guest.</td>
</tr>
<tr>
<td>iommu_bind/unbind_pasid_for_guest*</td>
<td>Bind guest process to host allocated PASID. Host owns system wide PASID table</td>
</tr>
<tr>
<td>iommu_cache Invalidate</td>
<td>Translation cache invalidation passed down from guest</td>
</tr>
<tr>
<td>iommu_report_device_fault</td>
<td>Report IOMMU detected device faults outside IOMMU subsystem, e.g. page request to be handled by guest.</td>
</tr>
<tr>
<td>iommu_sva_suspend/resume_pasid(*)</td>
<td>Device switch context while maintain PASID bond</td>
</tr>
<tr>
<td>iommu_page_response()</td>
<td>Send page response after page request is handled</td>
</tr>
<tr>
<td>iommu_sva_init/shutdown_device()</td>
<td>Prepare device for SVA, e.g. enable PRI, mm_exit notifier</td>
</tr>
<tr>
<td>iommu_sva_bind/unbind_device()</td>
<td>Create bond between mm, PASID, and device</td>
</tr>
<tr>
<td>PASID management APIs</td>
<td>Management and helper function for lookup</td>
</tr>
<tr>
<td>iommu_bind_guest_msi</td>
<td>Reuse gIOVA doorbell in host</td>
</tr>
</tbody>
</table>

* not yet published
Summary

• Shared Virtual Addressing (SVA) enables efficient workload submission by directly programming CPU virtual addresses on the device.

• Intel® VT-d 3.0 specification extends SVA usage together with Intel® Scalable I/O Virtualization.

• Holistic enhancements are introduced cross multiple kernel/user space components, to enable SVA virtualization in KVM.

• New kernel APIs are kept neutral to support all kinds of virtual IOMMUs (either emulated or para-virtualized).
Backup
Enable SVA in VM (Cont.)

Guest VT-d

Translation structure

PASID Table

\[ \text{RTAR} \]

1\text{st} level translation (i.e. CPU page tables)

GVA -> GPA

Host VT-d

Translation structure

PASID Table

\[ \text{RTAR} \]

2\text{nd} level translation (i.e. VT-d tables)

Guest SVA Support on Intel® VT-d
Enable SVA in VM (Cont.)

Guest VT-d

Translation structure

PASID Table

GVA -> GPA

1st level translation (i.e. CPU page tables)

Save guest cpu page table pointer to host

Host VT-d

Translation structure

PASID Table

2nd level translation (i.e. VT-d tables)

Guest SVA Support on Intel® VT-d
Enable SVA in VM (Cont.)

Guest VT-d

Translation structure

PASID Table

GVA -> GPA

1st level translation (i.e. CPU page tables)

Save guest cpu page table pointer to host

Host VT-d

Translation structure

PASID Table

RTAR

2nd level translation (i.e. VT-d tables)

RTAR

Guest SVA Support on Intel® VT-d

In nested translation, hardware treats 1st-level page table pointer as GPA
Enable SVA in VM (Cont.)

Guest VT-d

Translation structure

PASID Table

... RTAR

1st level translation (i.e. CPU page tables)

Save guest cpu page table pointer to host

Host VT-d

Translation structure

PASID Table

... RTAR

2nd level translation (i.e. VT-d tables)

GVA -> GPA

GPA -> HPA

Nested Translation

Guest SVA Support on Intel® VT-d
VT-d Extended Context Mode (Deprecated)
VT-d Scalable Mode (New)

**Key Difference:** PASID is a global ID space shared by all VMs.

ALL page-table pointers moved to PASID Granular table.