Sharing IOMMU page tables with TDP in KVM

Lu Baolu baolu.lu@intel.com
Zhao Yan yan.y.zhao@intel.com
Tian Kevin kevin.tian@intel.com

Sep. 2021
Disclaimers

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. No product or component can be absolutely secure.

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
Agenda

- Goal
- Sharing Advantages
- Sharing Prerequisites
- Sharing Interfaces
- Page & Page table Pinning
- Shared Page Table Root Update
- Bootup Performance
- TODOs
Goal
Sharing Advantages

- Reduced memory footprint
- Unified page table management
  - Dirty page tracking, page fault handling, etc.
- Probably higher performance by reducing unnecessary EPT/NPT zap
Sharing Prerequisites

- The same address space
- Compatible page table format
- Non-conflicting page table content
The Same Address Space

• Address space is GPA (L1) \( \rightarrow \) HPA

• Qemu
  – KVM side
    • check TDP is enabled
    • vCPU model does not include EPT/NPT feature
  – IOMMU side
    • no vIOMMU
    • vIOMMU is not in shadow mode. (nested mode on GPA is ok)
The Same Address Space (Cont.)

- Nested VM
  - TBD currently

- SMM in x86
  - A different address space. Cannot be shared to IOMMU.
  - Non-SMM mode EPT must be kept for sharing when vCPU is in SMM mode.
Compatible Page Table Formats

- Unified compatible page table format definition across KVM and IOMMU
- Compatible page table formats
  - FORMAT_EPT_LEVEL_4
  - FORMAT_EPT_LEVEL_5
  - FORMAT_NPT_LEVEL_4
  - FORMAT_NPT_LEVEL_5
  - ...
Sharing Handshake Sequence

1. get current sharable page table format
   - EPT_LEVEL_4

2. Check sharable format (EPT_LEVEL_4)
   - SUCCESS

3. Allocate IOASID with format EPT_LEVEL_4

4. Attach Device to IOASID

Request sharing

Note:
1. Device pass-through is based on the /dev/iommu proposal, which is IOASID oriented.
2. KVM shares TDP used by vCPU 0
Non-conflicting Page Table Content

- Presence of page table entry
  - For KVM user memslots
    - must be present and pinned (staying present) for DMA pages when IO page fault is not supported.
    - Can be present or zapped for non-DMA pages or when IO page fault supported
  - For KVM private memslots
    - Not present in IOPT before sharing
    - Safe to be present in IOPT after sharing

- Local APIC
  - DMA write to 0xfeexxxxx doesn’t go through DMA remapping.

- TSS and IDENTITY_PAGETABLE
  - for lenable_unrestricted_guest, E820 Reserved
Non-conflicting Page Table Content (Cont.)

- Read/Write/Execute bit
  - RO for RO memslots
  - RW for other memslots
  - Execute bit
    - currently ignored in IOMMU and no device uses it.
  - Write protection for live migration
    - Allowed when IO page fault is supported
    - Must be disabled otherwise
      - All pinned ranges are dirty or
      - traversal for Dirty bit
Sharing Interfaces

- Request/stop sharing
- Page/page table pinning for DMAs without IO page fault
- Page fault for IO page fault support
- Notification
  - Page table content update notification
  - Page table root update notification
Page & Page Table Pinning

For sharing without IO page fault,

- Pinning of VM pages
  - `pin_user_pages_*` (FOLL_LONGTERM)

- Pinning of TDP entries
  - Pre-population of pinned ranges
  - No zap/pfn update
  - No reclaiming of mmu pages with parent linked
  - Atomic update of TDP entries when permission or page size change
Atomic update is required for TDP entries for pinned ranges, when
- Splitting huge pages
- Updating of PTE permission

TDP entry being atomically updated from non-zero value to another non-zero value.
Page & Page Table Pinning Interfaces

- For sharing without IO page fault,
  - Pinning of all ranges in user memslots: memslot add
  - Pinning a specific range: extra interface

**Pin/Unpin from Qemu**
Pros:
QEMU doesn’t need to
MAP/UNMAP IOASID
which is using 3rd party TDP

**Pin/Unpin from IOMMU**
Pros:
Straightforward to hold more
IOMMU side info, e.g. snoop bit
**Shared Page Table Root Update**

1. Share TDP root of vcpu 0
2. IOMMU
3. root_count--
4. mmu root page (old role)
5. mmu root page (new role)
6. root_count++
7. root_count--
8. Atomic update done
9. Pre-population of TDP for pinned ranges
10. Root update notification
11. IOMMU
12. Root pointer
13. vcpu 0
14. vmcs
15. eptp
16. vcpu n-1
17. vmcs
18. eptp
19. vcpu n-2
20. vmcs
21. eptp
22. vcpu 0
23. Root pointer
24. Role
25. root_count=n
26. Share TDP root of vcpu 0
27. role
28. role
29. role
30. role
31. role
32. root_count++
33. If !role.smm, root_count++
Bootup Performance

Rough performance data without any optimization yet.

<table>
<thead>
<tr>
<th>8G memory</th>
<th>Bootup Time</th>
<th>Pre-population count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base (no sharing)</td>
<td>29s</td>
<td>0</td>
</tr>
<tr>
<td>Sharing (huge page enabled)</td>
<td>32s</td>
<td>132</td>
</tr>
<tr>
<td>Sharing (huge page disabled)</td>
<td>63s</td>
<td>132</td>
</tr>
</tbody>
</table>

- All VM pages were pinned/unpinned on user memslots creation/deletion.
- TDP was pre-populated on page table changes (when switching to new root, memslot add, and huge page splitting)
- IOTLB was flushed on page table root/content update notification (~1s)

- Quite a lot of time spend on TDP pre-population
  - ~2s with huge page
  - ~32s when huge page is disabled
- In concept can reach equal boot time performance as before sharing by reducing TDP root update count.
TODOs

- Snoop bit handling
- Unified dirty page tracking
- Nested VM (vIOMMU, virtual EPT/NPT)
- Performance optimization
  - Page table root update reduction,
  - Huge page support for P2P, etc.
Why it is KVM manages the shared table

- CPU side has more restrictions in page size
  - Check guest MTRR
  - NX huge page workaround

- CPU side has extra GFN ranges to access
  - Private memslots in kernel space

- IOMMU page tables are not always present.
Overall Design

1. Allocate IOASID
   - KVM_EPT_LEVEL_4

2. Attach/Detach IOASID

3. Notification of root/map update

4. Pin/Unpin
   - ghn range
   - anon_inode
   - KVM-vm

5. IO page fault

Device[n]

vfio
/dev/vfio/devices/dev[n]

IOMMU

IOASID

TDP

Legends:
- uAPI
- existing kernel interface
- KVM sharing on/off interface
- KVM new interface
- KVM notification
Overall Design (alternative)

1. Alloc IOASID
   - KVM_EPT_LEVEL_4

2. Attach/Detach IOASID
   - /dev/vfio/devices/dev[n]

vfio

anon_inode:
- kvm-vm

IOMMU

KVM

Device[n]

IOMMU

MAP_DMA/UNMAP_DMA

4b

Pin/Unpin on memslot create/delete

4a

TDP

KVM

- Pin All
- Pin/Unpin a gfn range
- page fault

- Request/Stop sharing

- notification of root/map update

- Attach/Detach

Page dimensions: 792.0x612.0