IOMMU Observability

Pasha Tatashin, Yu Zhao
LPC'23: VFIO/IOMMU/PCI MC

Outline

- Observability & Accounting
  - Growing IOMMU Page Tables
  - Reduction proposal
  - Observing proposal
  - Accounting proposal
- Stability & Debuggability
  - Retrieving the IOMMU Page Tables entries (and EPT)
  - Extend page_table_check to support IOMMU Page Tables (and EPT)

Growing IOMMU Page Tables

- IOMMU Page Tables are not freed on unmaps
- Can grow in size to multiple gigabytes
- No observability in kernel in the size of the IOMMU Page Tables
- Sample code to repro

```
while ()

    dma_map.iova += 2 << 20

    dma_unmap.iova += 2 << 20

    ioctl(container, VFIO_IOMMU_MAP_DMA, &dma_map)

    ioctl(container, VFIO_IOMMU_UNMAP_DMA, &dma_unmap)
```

Growing IOMMU Page Tables: iova_stress

AMD

```
# ./iova_stress -g 82 -s 45

iova space:     0T
free memory:   999G

…

iova space:    44T
free memory:   911G
```

Size: ~88G

Intel

```
# ./iova_stress -g 82 -s 45

iova space:     1T
free memory:   495G

…

iova space:    44T
free memory:   409G
```

Size: ~86G

ARM

```
# ./iova_stress -g 82 -s 45

iova space:     0T
free memory:   177G

…

iova space:    44T
free memory:    89G
```

Size: ~88G

Reduce Proposal

- Use page->_refcount to count number of entries currently mapped at page table level
- Add potential pagetable levels to per domain freelist
- Run a thread that periodically frees pages from the freelist

Question:

- Can we do mostly arch independent solutions to freeing page table levels on unmaps?
- Should memory shrinker, user configurations for maximum freelist size affect the freeing of the page table list?

Observability Proposal

- Add the size of IOMMU page tables in /proc/meminfo
- We have PageTables, SecPageTables, we can add another field:
- **IO_PageTables**, that shows the total size of all IOMMU PageTables.
- The accounting involves moving all page allocation functions such as: alloc_pages_node, __free_pages(), put_pages_list into common IOMMU functions
- Account using mod_node_page_state()
- Modify all IOMMU architectures to use the new common allocation functions
- Per-node information can be observed via: **nr_iommu_page_tables** field in /sys/devices/system/node/node0/vmstat, which automatically shows all node_stat fields.

Accounting Proposal

- Accounting involves adding GFP_KERNEL_ACCOUNT to the page allocation functions.
- Also, in order to be able to enforce cgroup rules, instead of simply counting node statistics, we should modify memcg counters as well, therefore, use mod_lruvec_page_state() with every page allocation.

Stability & Debuggability

- Retrieving IOMMU Page Table mappings
  - A similar interface to /proc/pid/pagemap should be added for IOMMU Page Tables (and Secondary Page Tables), so they can be accessed from the userland.
  - Where should such access reside?
  - Should be fast and lockless
- Dump IOMMU Page Table contents via crash utility

Stability & Debuggability, page_table_check

- Enhance page_table_check to support IOMMU and EPT mappings, what kind of rules could we add?
- Current page_table_check rules:

Current Mapping	New mapping	Permissions	Rule
Anonymous	Anonymous	Read	Allow
Anonymous	Anonymous	Read / Write	Prohibit
Anonymous	Named	Any	Prohibit
Named	Anonymous	Any	Prohibit
Named	Named	Any	Allow

Links

- https://github.com/soleen/iova_stress