Contribution ID: **226** Type: **not specified**

Shared Virtual Addressing (SVA) for in-kernel users

Thursday 23 September 2021 08:30 (30 minutes)

Sharing virtual addresses between DMA and the user process is undoubtedly beneficial. It improves security by limiting DMA to the process virtual address space; The programming model is simplified by eliminating the need for explicit map/unmap operations with behind the scene IO page fault handling. Potential performance gains come after that.

However, applying the same logic to kernel-SVA is not without controversy. The DMA API is the de facto way of doing kernel DMA. It already provides portability and security by means of IOVA. DMA API is IOMMU agnostic and does not support IOMMU specific key concept of Process Address Space ID (PASID) which SVA relies on. IOVA is supported by IOMMU with separate page tables than the CPU counterpart.

In order to support SVA, IOMMU has to walk CPU page tables which undermines security if we allow sharing the entire kernel virtual address (KVA) space. IOTLB flush is also a gap since mmu_notifier is not available for kernel memory.

This proposed session explores the multiple candidates that can make DMA API compatible, KVA usage safe, and address the gap of IOTLB synchronization.

I agree to abide by the anti-harassment policy

I agree

Primary author: PAN, Jacob

Presenter: PAN, Jacob

Session Classification: VFIO/IOMMU/PCI MC

Track Classification: VFIO/IOMMU/PCI MC