Updates from LPC

For 2026-01-08 guest_memfd bi-weekly upstream call

Contact ackerleytng@google.com if you have questions/suggestions!

Two guest_memfd microconference sessions

- One at MM microconference
- One at CoCo microconference

MM MC: folio restructuring: the ask

- To have folio restructuring generalized/extended to also handle HugeTLB folios
- Why
  - guest_memfd is going to have huge page support, with HugeTLB and buddy-allocated as providers
  - Wanted a common/general way to handle restructuring between the two

MM MC: folio restructuring: takeaways

- Community reiterated that HugeTLB is to be used by guest_memfd only as an allocator
  - While in guest_memfd ownership, nothing about the folio is HugeTLB other than HVO
- Generally OK to common code/generalization
  - Ok to having HVO for guest_memfd THP as well
  - David said there was a proposal before for THP in specific zones to gain HVO
- Willy: splitting from PUD size to PTE size involves a large amount of memory allocated post memdesc
  - Suggestion: logarithmic split
  - Willy aims to have memdesc done in Q1 2026
- David: Core-mm doesn’t really support folio sizes greater than PMDs other than PUD-sized folios
  - David suggested splitting from PUD to PMD and then to PTE-sized folios as an intermediate stage

MM MC: folio restructuring: next steps

- Previously
  - Upstream HugeTLB support in guest_memfd first (no conversion support)
  - Upstream HugeTLB with conversion support
  - Upstream HugeTLB, optimizing memory usage by restructuring to PMD-sized folios
- With LPC feedback
  - The later two stages might need to be combined
  - Should not impact HugeTLB support without conversions
    - Will focus on this after upstreaming conversions?
    - Or would people rather have HugeTLB support before conversions?
      - Will need to weigh this against internal requirements

CoCo MC: IOMMU wrt guest_memfd: the ask

- Let guest_memfd unmap from the IOMMU page tables on shared to private conversions
- Why
  - guest_memfd has the responsibility of making sure that guest private memory is not mapped in the host (or anywhere else that could interfere with guest access, like IOMMU page tables)

CoCo MC: IOMMU wrt guest_memfd: takeaways

- Realized I need to understand different CoCo platforms (SNP/pKVM/TDX/ARM CCA) better
- My understanding of the discussion is in the following slides, hope to clarify this together :)
- People seemed to be questioning the need to unmap from IOMMU page tables. Two aspects to this
  - Is unmapping guest private memory required on all platforms?
  - Is the IOMMU page table actually the same page table as the stage 2 page tables?
    - i.e. Is unmapping from stage 2 page tables == unmapping from IOMMU page tables?

IOMMU wrt guest_memfd: platform requires unmapping?

- TDX requires guest private memory to be unmapped because writes to guest private memory could end up taking down the host
- SNP will prevent writes to guest private memory, accesses to guest private memory will result in errors (not a host uptime issue)?
- pKVM - how does it work there?
- ARM CCA - how does it work there?

IOMMU wrt guest_memfd: IOMMU shares page table?

- TDX with traditional IO: requires unmapping. Not the same page table.
- TDX with Confidential IO?
- SNP with traditional IO?
- SNP with Confidential IO?
- pKVM - how does it work there?
- ARM CCA - how does it work there?