guest_memfd HugeTLB overview

For 2026-02-23 Hypervisor Live Update call

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

Introduction to guest_memfd

- guest_memfd is a guest-first memory provider that is KVM-specific
- Like tmpfs or HugeTLBfs when used for virtualization
  - Has an fd
  - Can be mmap()-ed
- In addition to those, it has tracking of memory attributes: shared vs private (in the CoCo sense)
  - Private memory cannot be mapped to userspace
    - mmap() is okay but access (faulting) will result in a SIGBUS

Actually providing memory

- guest_memfd wraps existing sources of memory
    - PAGE_SIZE pages from the buddy allocator
    - Huge pages from HugeTLB
- Put those folios in a filemap (like tmpfs or HugeTLBfs)

Memory attribute tracking and “conversions”

- Every page is individually tracked to be either shared or private
- Confidential guests can request a private page to be shared with the host (aka private to shared conversions), or the reverse

Conversion flow: shared to private

1. Guest requests conversions with a hypercall
2. KVM exits to userspace
3. Userspace VMM makes sure that devices stop using the memory requested to be converted
4. Userspace sends the `SET_MEMORY_ATTRIBUTES` ioctl to guest_memfd
5. guest_memfd unmaps requested range from userspace page tables
6. guest_memfd records the page to be private
7. Userspace does a `KVM_RUN`

Conversion flow: shared to private

1. Guest requests conversions with a hypercall
2. KVM exits to userspace
3. Userspace VMM makes sure that devices stop using the memory requested to be converted
4. Userspace sends the `SET_MEMORY_ATTRIBUTES` ioctl to guest_memfd
5. guest_memfd unmaps requested range from userspace page tables
6. guest_memfd records the page to be private
7. Userspace does a `KVM_RUN`

Kernel makes sure that there are no users

- No users is defined as `refcount == guest_memfd’s refcount`
- Refcounts taken on huge pages are tracked for the entire huge page
  - Unable to identify which page within the huge page the refcount was taken for
- Enable per-page refcounting by splitting pages
  - So that a refcount on the last page can be distinguished from a refcount on the first page

Split + merge = folio restructuring

- Split folios have to be merged before returning them to HugeTLB
- Split folios may outlive guest_memfd, may even outlive KVM
  - fd might be closed before the memory is unpinned
- Hence there is more folio metadata tracked outside of KVM, e.g.
  - What was the original size of this folio before splitting?
  - What was this folio’s memcg?
- Memory failure handling also uses this folio metadata to identify guest_memfd HugeTLB folios
  - Traditional memory failure handling may race with restructuring during conversions

Summary of stuff to be preserved during KHO

- For guest_memfd (PAGE_SIZE folios)
  - Filemap and associated folios
  - Mempolicy (stored on inode)
  - Association with the related KVM, memslot bindings
  - Memory attributes (maple tree)
- For guest_memfd HugeTLB
  - Restructuring metadata
  - (What happens if some memory failure happens during KHO?)

Timelines/estimates

- Conversion support (also introduces private/shared tracking)
  - 1 outstanding uAPI issue (memory content preservation during conversions)
  - Hope to merge ~March 2026
- HugeTLB support without restructuring
  - Some remaining implementation details to figure out
  - Hope to merge ~June 2026
- HugeTLB support with restructuring
  - Hopefully September 2026?

- Implement guest_memfd preservation in the same order?