# Following up on discussion last time regarding `kvm_mem_is_private()` usage/handling in KVM vs guest_memfd's new features

- Go over how we introduce userspace-visible changes to ensure that use cases will be taken care of
- 4 use cases I can think of, any others?
    a. Backward compatibility for Coco VMs that will use guest_memfd only for private memory and other backing memory for shared memory
    b. Non-coco VMs that will use guest_memfd, will require mmap to userspace
        - guest_memfd should optionally allow removal of memory from kernel direct map
    c. pKVM
    d. Coco VMs that will use guest_memfd for both shared and private memory

## New features stage 1: guest_memfd gains mmap() support

- V8 https://lore.kernel.org/all/20250430165655.605595-1-tabba@google.com/ was posted yesterday. The following points describe the desired state, not v8.
- New guest_memfd flag: `GUESTMEM_FLAG_SUPPORT_SHARED`
    - With this flag, guest_memfd can now be mmap-ed to userspace
    - On binding of guest_memfd to a memslot, validate that the VM type is not some Coco VM if `GUEST_MEMFD_FLAG_SUPPORT_SHARED` is set.
    - For a memslot configured with a guest_memfd that has `GUEST_MEMFD_FLAG_SUPPORT_SHARED` set, KVM will use `kvm_gmem_get_pfn()` for both shared and private guest faults
        - For accesses like instruction emulation, KVM will still use `userspace_addr` to read guest memory.
        - On binding, also validate that `userspace_addr` refers to the same folio as fd+offset
            - Validate only if `userspace_addr` is not `NULL`, so that userspace has the option to disallow any use of `userspace_addr` for accessing memory (e.g. to disallow instruction emulation)
        - guest_memfd only allows MAP_SHARED, so no need to validate mapping type on binding
- New KVM cap to indicate that guest_memfd can now support mmap: `CONFIG_KVM_GMEM_SHARED_MEM`
- Usage
    - Coco VMs can continue to use guest_memfd only for private memory. These VMs will not set `GUEST_MEMFD_FLAG_SUPPORT_SHARED`, so mmap will not be enabled for these VMs
    - Non-coco VMs can now use guest_memfd by setting `GUEST_MEMFD_FLAG_SUPPORT_SHARED` - mmap will be enabled
    - pKVM: cannot use this stage yet? Since there's still nothing to prevent `kvm_gmem_fault()` from returning a guest-owned page

## New features stage 2: guest_memfd gains conversion support

- New KVM CAP to indicate conversion support
- If `GUEST_MEMFD_FLAG_SUPPORT_SHARED` is set,
    - All of the new features from stage 1, except that now Coco VMs can bind even if `GUEST_MEMFD_FLAG_SUPPORT_SHARED` is set
    - guest_memfd will initialize the shareability xarray and will default to initialization as SHARED.
    - guest_memfd will perform conversions (updating of the shareability xarray, merge/split of page when 1G support is added)
    - kvm_gmem_fault() will return a page `if shareability == ALL`, `SIGBUS` otherwise.
    - `kvm_mem_is_private()` will query guest_memfd for private/shared status (aka shareability)
        - Any mismatch between `fault->is_private` and shareability will result in `KVM_EXIT_MEMORY_FAULT`
    - `kvm_gmem_get_pfn()` will not validate shareability state against `fault->is_private`
        - By the time `kvm_gmem_get_pfn()` is called, the fault type is already determined to match with guest_memfd's shareability status.
        - Page preparation will be handled based on shareability status

    - When truncating, guest_memfd will request invalidation of both private and shared mappings, since guest_memfd can no longer rely on unmapping from userspace to trigger invalidation of shared mappings via mmu notifiers
        - Truncation will call unmap, which will trigger invalidation of shared mappings again, but that should be a small performance penalty since the second unmap will not cause a TLB flush for guests.
        - David Hildenbrand suggested to remove mmu notifiers for guest_memfd folios - that can be an optimization?
        - Sean Christopherson said that leaving mmu notifiers could be a feature instead of a bug - there may be different mappings for the same folios, or unmapping may not go through guest_memfd
        - Sean Christopherson also suggested that perhaps a guest_memfd specific, second HVA field could be added, which would take the place of `userspace_addr` for guest_memfd.
- guest_memfd gains new ioctls, `CONVERT_SHARED` and `CONVERT_PRIVATE`, enabled if `GUEST_MEMFD_FLAG_SUPPORT_SHARED` is set
- New guest_memfd flag, `GUEST_MEMFD_FLAG_INIT_PRIVATE`
    - Shareability defaults to ALL (shared with host), which aligns with the default for `kvm->mem_attr_array`
    - Set this flag to initialize guest_memfd with shareability set to `GUEST`
- Usage
    - Coco VMs can continue to use guest_memfd only for private memory, these VMs will not set `GUEST_MEMFD_FLAG_SUPPORT_SHARED`, so mmap will not be enabled for these VMs
        - Coco VMs wanting to use guest_memfd only for private memory can also set both `GUEST_MEMFD_FLAG_SUPPORT_SHARED` and `GUEST_MEMFD_FLAG_INIT_PRIVATE`.
            - In this case, any calls to conversion ioctl will still be handled. It is the fault of the userspace VMM.
            - It does not reopen the hole guest_memfd was meant to patch since host faults are guarded by a shareability check.
    - Non-coco VMs can use guest_memfd by setting `GUEST_MEMFD_FLAG_SUPPORT_SHARED`. Shareability is initialized by default to shared, so no change required here.
        - Any calls to the conversion ioctls will be handled and is the fault of userspace. No additional checks here.
    - pKVM will specify `GUEST_MEMFD_FLAG_SUPPORT_SHARED` and call guest_memfd conversion functions without exiting to userspace.
    - Coco VMs that use guest_memfd for both shared and private memory will specify `GUEST_MEMFD_FLAG_SUPPORT_SHARED` and optionally `GUEST_MEMFD_FLAG_INIT_PRIVATE`, and will use the conversion ioctls to convert memory.

Discussion slides are at: https://lpc.events/event/18/contributions/1764/attachments/1409/3708/2025-05-01-kvm-memory-attributes-vs-guest_memfd-shareability.pdf