# KVM memory attributes vs guest_memfd shareability

## updates and questions

ackerleytng, 2025-05-01
guest_memfd upstream call

# Background

- Fuad has been working on supporting conversions to share the same physical memory [1] [2]
- This will solve the double allocation problem

[1] https://lore.kernel.org/all/20250318161823.4005529-1-tabba@google.com/T/
[2] https://lore.kernel.org/all/20250328153133.3504118-1-tabba@google.com/T/

# Current state

- Track `shareability` in guest_memfd inode, determines whether a page
  - Can be faulted to a userspace page table or
  - If the page belongs to the guest
- `kvm->mem_attr_array` also tracks whether the page is private or shared
- pKVM will not be using `kvm->mem_attr_array`
- Confidential VMs convey the private/shared access type during fault handling
  - Don’t need to store private/shared status in two places

# guest_memfd conversion ioctl?

- Previous RFC [1]: userspace informs KVM of conversion using `KVM_SET_MEMORY_ATTRIBUTES`
  - Iterate memslots in range, convert range for each inode in memslot
- New proposal: a guest_memfd (not KVM but directly to guest_memfd) ioctl for conversion, which takes params: `offset`, `size`

[1] https://lore.kernel.org/all/cover.1726009989.git.ackerleytng@google.com/

# Advantages

over using KVM_SET_MEMORY_ATTRIBUTES ioctl

# Advantage 1: Aligned with other memory providers

- Other memory providers use mmu_notifiers when unmapping happens and KVM gets informed
  - Action originates from memory and KVM is notified
- With guest_memfd conversion ioctl, guest_memfd will notify KVM from conversion ioctl

# Advantage 2: Can convert independently of memslots

- Converting via the KVM_SET_MEMORY_ATTRIBUTES ioctl assumes guest_memfd is bound using memslots
- At VM reboot, guest_memfd is disassociated from memslots, and memory needs to be restored to allow private faults
- Having a direct guest_memfd ioctl avoids having to first bind memslots
- Sean's comment from guest_memfd call 2025-05-01:
  - VM reboot is not a strong advantage. Rather, conversion is about setting memory attributes, setting the attributes directly with an ioctl makes sense, instead of setting memory attributes via KVM.

# Advantage 3: Remove duplicate state tracking

- Can avoid duplicate shared/private state tracking in `kvm->mem_attr_array` and guest_memfd's shareability
- When guest_memfd is used for both shared and private memory and guest_memfd tracks shareability, then there won't be a need to have the same information tracked again in `kvm->mem_attr_array`.
- Discussion from guest_memfd call 2025-05-01:
  - Sean: If the memslot is destroyed, conversion happens, then re-attached to VM, then VM didn't know the transition happened?
    - Destroying memslot invalidates memory, fault goes via guest_memfd anyway so VM would know later
  - Sean: VM's memslot's lpage_info needs to be updated with conversion?
    - lpage_info helps determine mapping level
      - max_order is returned from guest_memfd, which contributes to determining mapping level
  - Michael Roth: For SNP, if we have a hugepage, not split, RMP table is still split
    - guest_memfd returns a smaller order, RMP table can get updated
    - guest_memfd owns the RMP table, need arch-specific hooks to set up
  - James Houghton: two VMs use the same inode, same guest_memfd
    - Invalidation iterates and invalidates in all the VMs
    - VMs query the same (single) guest_memfd state

# Advantage 4: Avoid complex error handling in kernel

- In case of conversion failures (e.g. failing to split/merge a page due to memory pressure) we want to restore pre-conversion shareability state
- Using KVM_SET_MEMORY_ATTRIBUTES requires iterating memslots and applying conversions per-inode
  - A failure after the first memslot would require restoring state
  - This complex error handling can be handled in userspace with a direct guest_memfd conversion ioctl

# Addressing gaps

After ignoring mem_attr_array

# No deprecation of shared/private in mem_attr_array

- Tracking shared/private page state in `kvm->mem_attr_array` must be retained for VMs that use guest_memfd only for private memory

# Other uses of `kvm->mem_attr_array`

- Letting userspace handle implicit conversions
  - When there is a mismatch between memory attribute vs fault type (shared/private), exit to userspace
    - Instead of checking memory attributes, for a guest_memfd that supports conversions, query guest_memfd for memory status
- Determining mapping level for a page
  - Query guest_memfd instead
- Determining fault path (`slot->userspace_addr` vs `kvm_gmem_get_pfn()`) for `KVM_X86_SW_PROTECTED_VM`
  - No change, continue to use `KVM_SET_MEMORY_ATTRIBUTES` ioctl