Minimum Support for Memory Failure Handling to land guest_memfd//HugeTLB

For 2025-07-24 guest_memfd bi-weekly upstream call

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

Overview

- Last time: proposed `memory_failure()` handling for guest_memfd
- Motivation: need to address inconsistencies that current `memory_failure()` handling will create once guest_memfd gains HugeTLB support

Goal for today

- Seek community alignment on minimal memory handling for guest_memfd HugeTLB support to land in the kernel
  - Let us use HugeTLB before full memory failure handling support is complete
  - Hope to address the inconsistencies and at least have defined behavior on memory failure

What does minimum support look like?

When memory failure is detected for a given PFN *that belongs to guest_memfd*,

- Track memory failure in global data structure
- Then, if
  - Memory was unconsumed, but failure detected
    - Let all processes continue to run, defer until failed memory is consumed
  - Failed memory was consumed
    - Send SIGBUS to `current`
- When freeing folio on inode release, indicate any memory failures to HugeTLB using folio's HWpoison flag

- Don't unmap failed memory
- Only mark HWpoison on folios when releasing folios from guest_memfd
- On faults, don't check for memory failure
- Don't handle `PR_MCE_KILL_EARLY`

Risk: keeping failed memory mapped

- Give up on a chance to contain memory failures to one VMM/guest
  - Subsequent accesses (in certain cases) may take down host
  - Raised by Dan last week
- Can we accept this denial of service risk since
  - Memory failure is in itself not common, hence exploitation will be hard
    - No #MCE virtualization - reliable exploitation is harder without knowledge of memory failure
- Sean: Unacceptable risk
  - To prevent guest from continuously accessing, there's some unmapping that happens
- Dan: Unmap on error is non-negotiable
- Sean: Kill the guest completely and unmap everything is possible
- James: Make guest_memfd completely unusable is a good idea
  - Unconsumed error: can defer, can handle it too. This is rare.
    - Dan: guest: has to trust. Best effort for unconsumed
  - Dan: first cut, Sean: stick approach
  - HugeTLB: handling consumed memory error is best-effort. If in the middle of migrating a page, then kind of ignore it. But migration is a transient thing for HugeTLB, here we're talking about long term usage.
- David: In QEMU, people are used to complete VM dying

- Unmapping complicates handling
  - guest_memfd does runtime folio restructuring, unmapping at the right granularity requires locking
- Unmapping will break CoCo VMs since we can't pair unmapping with #MCE virtualization

Did not discuss the slides after this one

Risk: folios with failed memory not marked HWpoison

- There may be kernel users relying on HWpoison flag (while folios are allocated and owned by guest_memfd) that may access failed memory and take down host?
- Clarifying this concern
  - Was the concern about folios that guest_memfd already owns and have been given to the VM to use?
  - What are the kernel users that may access a folio already allocated and owned by guest_memfd, that rely on this flag to gate access to failed memory?
- Can we accept this denial of service risk?

- Marking HWpoison complicates folio restructuring
  - guest_memfd does runtime folio restructuring, marking HWpoison requires tracking and taking locks

Limitation: SIGBUS-ing `current` for consumed failure

- `current` may not always be the VM
- Can we accept this limitation?
  - QEMU uses `PR_MCE_KILL_EARLY` only for the main thread to log the error, the vCPU threads use `PR_MCE_KILL_LATE`.
  - `current` is most likely the VMM for consumed memory failures on private memory
  - For consumed memory failures on shared memory, `current` will probably be VMM related and can coordinate with the VMM to handle SIGBUS.

- There is existing support for finding the owner process of a failed PFN, but only for shared folios
  - Shared/private status is dynamic and it's awkward to have partial support
- For private folios, guest_memfd tracks the owning `struct kvm`, but no support for finding the VMM process for a given `struct kvm`

Limitation: No handling for `PR_MCE_KILL_EARLY`

- Even if `PR_MCE_KILL_EARLY` is set, the requesting thread will not be notified
- For now, no way to find owning process of private memory
  - SIGBUS-ing `current` only fulfills the purpose of `PR_MCE_KILL_EARLY` if `current` is the VMM thread that wants to get notified of unconsumed (but detected) memory failures
    - But for unconsumed memory failures, `current` can be any process and is probably not the userspace VMM
- Can support `PR_MCE_KILL_EARLY` together with proper support for SIGBUS-ing owner of private memory

Future work

- Unmap only subfolios that have failed memory
  - Allow opt-in to unmapping from userspace using guest_memfd flag
- Indicate HWpoison on folios: sort out locking to either
  - Support marking HWpoison on guest_memfd folios
  - Or only mark HWpoison for shared folios? When private memory doesn't have folios?
  - Will have to allow faulting in spite of HWpoison flag with some opt-in mechanism
- Build mechanism for sending signal (like SIGBUS) to processes associated with a `struct kvm`
  - Probably have a new kvm request type and use `kvm_make_all_cpus_request()`
  - Can then support `PR_MCE_KILL_EARLY`