Contribution ID: **287** Type: **not specified**

Post-copy live migration with guest_memfd

Thursday, 19 September 2024 13:00 (30 minutes)

Problem: traditional implementation of post-copy live migration

The key challenge with post-copy live migration is intercepting accesses to particular pages of guest memory. Today, the only way to do this is to use userfaultfd, an mm feature that intercepts page faults (and other events). KVM, after translating a GFN to an HVA, will call GUP to translate the HVA to an HPA, thereby triggering userfaultfd.

When using guest_memfd, KVM does not use GUP to translate from GFN to HPA, nor is there a GFN-to-HVA translation step. Therein lies the problem: userfaultfd cannot intercept these translations, making post-copy live migration impossible.

Solution

Given that guest_memfd is entirely separate from the main mm fault handling logic, either (1) userfaultfd must be extended to handle non-mm faults, or (2) something else needs to be created.

There are at least two options for how userfaultfd could potentially be extended: (1) add KVM-related operations to it, or (2) add file-related operations to it (for guest_memfd). Both are complex.

We can limit the overall added complexity by adding a KVM-based post-copy solution. Let's call it KVM Userfault. At its core, we need:

1. A way to inform KVM that certain pages should generate faults.
2. A mechanism for informing userspace of faults as they happen.
3. A way for userspace to fully handle the faults and resume VM execution.

The most straightforward way to inform KVM of which pages should fault and which should not is to use a new memory attribute. Doing so has several challenges, especially with respect to performance/scalability. Another possibility is to use a separate, potentially bitmap-based UAPI.

With respect to notifying userspace of faults, for vCPU faults, we can use KVM_EXIT_MEMORY_FAULT. For other faults (e.g. when KVM itself is accessing guest memory), we likely need to use a userfaultfd-like asynchronous notification. Although KVM does not itself read guest_memfd memory today, after guest_memfd supports shared memory, this will become a possibility.

The KVM-based solution is crudely implemented in the KVM Userfault RFC, using memory attributes and including asynchronous userfaults.

Main points of discussion

1. Is KVM Userfault an appropriate solution for post-copy with guest_memfd?
2. Should KVM Userfault use memory attributes? Should the API to set/clear userfault-enabled pages be bitmap-based?
3. What should the API look like for asynchronous page faults? How should asynchronous page faults be implemented (e.g. with wait_queue)? Is it possible to avoid the need for asynchronous userfaults?

4. Should “userfault-enabled” for a gfn be a property of the VM or of the memslot the gfn resides in?
5. Should KVM Userfault support traditional memslots?

Primary author: HOUGHTON, James (Google)

Co-author: MATLACK, David (Google)

Presenter: HOUGHTON, James (Google)

Session Classification: KVM Microconference

Track Classification: KVM Microconference