author	Lorenzo Stoakes <lorenzo.stoakes@oracle.com>	2025-03-21 10:09:37 +0000
committer	Andrew Morton <akpm@linux-foundation.org>	2025-04-11 17:32:37 -0700
commit	41e6ddcaa0f18dda4c3fadf22533775a30d6f72f (patch)	
tree	8946336159763b12035136a6b896489a4a43444b	
parent	9c02223e2d9df5cb37c51aedb78f3960294e09b5 (diff)	
download	linux-41e6ddcaa0f18dda4c3fadf22533775a30d6f72f.tar.gz	

mm/vma: add give_up_on_oom option on modify/merge, use in uffd release

Currently, if a VMA merge fails due to an OOM condition arising on commit merge or a failure to duplicate anon_vma's, we report this so the caller can handle it.

However there are cases where the caller is only ostensibly trying a merge, and doesn't mind if it fails due to this condition.

Since we do not want to introduce an implicit assumption that we only actually modify VMAs after OOM conditions might arise, add a 'give up on oom' option and make an explicit contract that, should this flag be set, we absolutely will not modify any VMAs should OOM arise and just bail out.

Since it'd be very unusual for a user to try to vma_modify() with this flag set but be specifying a range within a VMA which ends up being split (which can fail due to rlimit issues, not only OOM), we add a debug warning for this condition.

The motivating reason for this is uffd release - syzkaller (and Pedro Falcato's VERY astute analysis) found a way in which an injected fault on allocation, triggering an OOM condition on commit merge, would result in uffd code becoming confused and treating an error value as if it were a VMA pointer.

To avoid this, we make use of this new VMG flag to ensure that this never occurs, utilising the fact that, should we be clearing entire VMAs, we do not wish an OOM event to be reported to us.

Many thanks to Pedro Falcato for his excellent analysis and Jann Horn for his insightful and intelligent analysis of the situation, both of whom were instrumental in this fix.

Link: https://lkml.kernel.org/r/20250321100937.46634-1-lorenzo.stoakes@oracle.com
Reported-by: syzbot+20ed41006cf9d842c2b5@syzkaller.appspotmail.com
Closes: https://lore.kernel.org/all/67dc67f0.050a0220.25ae54.001e.GAE@google.com/
Fixes: 47b16d0462a4 ("mm: abort vma_modify() on merge out of memory failure")
Signed-off-by: Lorenzo Stoakes <lorenzo.stoakes@oracle.com>
Suggested-by: Pedro Falcato <pfalcato@suse.de>
Suggested-by: Jann Horn <jannh@google.com>
Cc: <stable@vger.kernel.org>
Signed-off-by: Andrew Morton <akpm@linux-foundation.org>

Diffstat

-rw-r--r--	mm/userfaultfd.c	13
-rw-r--r--	mm/vma.c	51
-rw-r--r--	mm/vma.h	9

3 files changed, 66 insertions, 7 deletions

```
diff --git a/mm/userfaultfd.c b/mm/userfaultfd.c
index fbf2cf62ab9f53..7d5d709cc838ed 100644
--- a/mm/userfaultfd.c
+++ b/mm/userfaultfd.c
@@ -1902,6 +1902,14 @@ struct vm_area_struct *userfaultfd_clear_vma(struct vma_iterator *vmi,
                                             unsigned long end)
 {
        struct vm_area_struct *ret;
+       bool give_up_on_oom = false;
+
+       /*
+        * If we are modifying only and not splitting, just give up on the merge
+        * if OOM prevents us from merging successfully.
+        */
+       if (start == vma->vm_start && end == vma->vm_end)
+               give_up_on_oom = true;

        /* Reset ptes for the whole vma range if wr-protected */
        if (userfaultfd_wp(vma))
@@ -1909,7 +1917,7 @@ struct vm_area_struct *userfaultfd_clear_vma(struct vma_iterator *vmi,

        ret = vma_modify_flags_uffd(vmi, prev, vma, start, end,
                                    vma->vm_flags & ~__VM_UFFD_FLAGS,
-                                   NULL_VM_UFFD_CTX);
+                                   NULL_VM_UFFD_CTX, give_up_on_oom);

        /*
         * In the vma_merge() successful mprotect-like case 8:
@@ -1960,7 +1968,8 @@ int userfaultfd_register_range(struct userfaultfd_ctx *ctx,
                new_flags = (vma->vm_flags & ~__VM_UFFD_FLAGS) | vm_flags;
                vma = vma_modify_flags_uffd(&vmi, prev, vma, start, vma_end,
                                            new_flags,
-                                           (struct vm_userfaultfd_ctx){ctx});
+                                           (struct vm_userfaultfd_ctx){ctx},
+                                           /* give_up_on_oom = */false);
                if (IS_ERR(vma))
                        return PTR_ERR(vma);


diff --git a/mm/vma.c b/mm/vma.c
index 5cdc5612bfc19e..839d12f02c885d 100644
--- a/mm/vma.c
+++ b/mm/vma.c
@@ -666,6 +666,9 @@ static void vmg_adjust_set_range(struct vma_merge_struct *vmg)
 /*
  * Actually perform the VMA merge operation.
  *
+ * IMPORTANT: We guarantee that, should vmg->give_up_on_oom is set, to not
+ * modify any VMAs or cause inconsistent state should an OOM condition arise.
+ *
  * Returns 0 on success, or an error value on failure.
  */
 static int commit_merge(struct vma_merge_struct *vmg)
@@ -685,6 +688,12 @@ static int commit_merge(struct vma_merge_struct *vmg)
```

```
        init_multi_vma_prep(&vp, vma, vmg);

+       /*
+        * If vmg->give_up_on_oom is set, we're safe, because we don't actually
+        * manipulate any VMAs until we succeed at preallocation.
+        *
+        * Past this point, we will not return an error.
+        */
        if (vma_iter_prealloc(vmg->vmi, vma))
                return -ENOMEM;

@@ -915,7 +924,13 @@ static __must_check struct vm_area_struct *vma_merge_existing_range(
                if (anon_dup)
                        unlink_anon_vmas(anon_dup);

-               vmg->state = VMA_MERGE_ERROR_NOMEM;
+               /*
+                * We've cleaned up any cloned anon_vma's, no VMAs have been
+                * modified, no harm no foul if the user requests that we not
+                * report this and just give up, leaving the VMAs unmerged.
+                */
+               if (!vmg->give_up_on_oom)
+                       vmg->state = VMA_MERGE_ERROR_NOMEM;
                return NULL;
        }

@@ -926,7 +941,15 @@ static __must_check struct vm_area_struct *vma_merge_existing_range(
 abort:
        vma_iter_set(vmg->vmi, start);
        vma_iter_load(vmg->vmi);
-       vmg->state = VMA_MERGE_ERROR_NOMEM;
+
+       /*
+        * This means we have failed to clone anon_vma's correctly, but no
+        * actual changes to VMAs have occurred, so no harm no foul - if the
+        * user doesn't want this reported and instead just wants to give up on
+        * the merge, allow it.
+        */
+       if (!vmg->give_up_on_oom)
+               vmg->state = VMA_MERGE_ERROR_NOMEM;
        return NULL;
 }

@@ -1068,6 +1091,10 @@ int vma_expand(struct vma_merge_struct *vmg)
                /* This should already have been checked by this point. */
                VM_WARN_ON_VMG(!can_merge_remove_vma(next), vmg);
                vma_start_write(next);
+               /*
+                * In this case we don't report OOM, so vmg->give_up_on_mm is
+                * safe.
+                */
                ret = dup_anon_vma(middle, next, &anon_dup);
                if (ret)
                        return ret;
@@ -1090,9 +1117,15 @@ int vma_expand(struct vma_merge_struct *vmg)
        return 0;

 nomem:
-       vmg->state = VMA_MERGE_ERROR_NOMEM;
        if (anon_dup)
                unlink_anon_vmas(anon_dup);
```

```
+       /*
+        * If the user requests that we just give upon OOM, we are safe to do so
+        * here, as commit merge provides this contract to us. Nothing has been
+        * changed - no harm no foul, just don't report it.
+        */
+       if (!vmg->give_up_on_oom)
+               vmg->state = VMA_MERGE_ERROR_NOMEM;
        return -ENOMEM;
 }

@@ -1534,6 +1567,13 @@ static struct vm_area_struct *vma_modify(struct vma_merge_struct *vmg)
        if (vmg_nomem(vmg))
                return ERR_PTR(-ENOMEM);

+       /*
+        * Split can fail for reasons other than OOM, so if the user requests
+        * this it's probably a mistake.
+        */
+       VM_WARN_ON(vmg->give_up_on_oom &&
+                  (vma->vm_start != start || vma->vm_end != end));
+
        /* Split any preceding portion of the VMA. */
        if (vma->vm_start < start) {
                int err = split_vma(vmg->vmi, vma, start, 1);
@@ -1602,12 +1642,15 @@ struct vm_area_struct
                      struct vm_area_struct *vma,
                      unsigned long start, unsigned long end,
                      unsigned long new_flags,
-                     struct vm_userfaultfd_ctx new_ctx)
+                     struct vm_userfaultfd_ctx new_ctx,
+                     bool give_up_on_oom)
 {
        VMG_VMA_STATE(vmg, vmi, prev, vma, start, end);

        vmg.flags = new_flags;
        vmg.uffd_ctx = new_ctx;
+       if (give_up_on_oom)
+               vmg.give_up_on_oom = true;

        return vma_modify(&vmg);
 }
```

diff --git a/mm/vma.h b/mm/vma.h
index 7356ca5a22d332..149926e8a6d1ac 100644
--- a/mm/vma.h
+++ b/mm/vma.h

```
@@ -114,6 +114,12 @@ struct vma_merge_struct {
         */
        bool just_expand :1;

+       /*
+        * If a merge is possible, but an OOM error occurs, give up and don't
+        * execute the merge, returning NULL.
+        */
+       bool give_up_on_oom :1;
+
        /* Internal flags set during merge process: */

        /*
@@ -255,7 +261,8 @@ __must_check struct vm_area_struct
                      struct vm_area_struct *vma,
                      unsigned long start, unsigned long end,
```

```
                      unsigned long new_flags,
-                     struct vm_userfaultfd_ctx new_ctx);
+                     struct vm_userfaultfd_ctx new_ctx,
+                     bool give_up_on_oom);

 __must_check struct vm_area_struct
 *vma_merge_new_range(struct vma_merge_struct *vmg);
```