author	Jinjiang Tu <tujinjiang@huawei.com>	2025-03-18 16:39:39 +0800
committer	Andrew Morton <akpm@linux-foundation.org>	2025-03-21 22:03:17 -0700
commit	1b0449544c6482179ac84530b61fc192a6527bfd (patch)	
tree	5003bdeb9ce928169b0ace5a002cd9eedddf953b	
parent	5f5ee52d4f58605330b09851273d6e56aaadd29e (diff)	
download	linux-1b0449544c6482179ac84530b61fc192a6527bfd.tar.gz	

mm/vmscan: don't try to reclaim hwpoison folio

```
Syzkaller reports a bug as follows:

Injecting memory failure for pfn 0x18b00e at process virtual address 0x20ffd000
Memory failure: 0x18b00e: dirty swapcache page still referenced by 2 users
Memory failure: 0x18b00e: recovery action for dirty swapcache page: Failed
page: refcount:2 mapcount:0 mapping:0000000000000000 index:0x20ffd pfn:0x18b00e
memcg:ffff0000dd6d9000
anon flags: 0x5ffffe00482011(locked|dirty|arch_1|swapbacked|hwpoison|node=0|zone=2|lastcpupid=0xfffff)
raw: 005ffffe00482011 dead000000000100 dead000000000122 ffff0000e232a7c9
raw: 0000000000020ffd 0000000000000000 00000002ffffffff ffff0000dd6d9000
page dumped because: VM_BUG_ON_FOLIO(!folio_test_uptodate(folio))
------------[ cut here ]------------
kernel BUG at mm/swap_state.c:184!
Internal error: Oops - BUG: 00000000f2000800 [#1] SMP
Modules linked in:
CPU: 0 PID: 60 Comm: kswapd0 Not tainted 6.6.0-gcb097e7de84e #3
Hardware name: linux,dummy-virt (DT)
pstate: 80400005 (Nzcv daif +PAN -UAO -TCO -DIT -SSBS BTYPE=--)
pc : add_to_swap+0xbc/0x158
lr : add_to_swap+0xbc/0x158
sp : ffff800087f37340
x29: ffff800087f37340 x28: fffffc00052c0380 x27: ffff800087f37780
x26: ffff800087f37490 x25: ffff800087f37c78 x24: ffff800087f377a0
x23: ffff800087f37c50 x22: 0000000000000000 x21: fffffc00052c03b4
x20: 0000000000000000 x19: fffffc00052c0380 x18: 0000000000000000
x17: 296f696c6f662865 x16: 7461646f7470755f x15: 747365745f6f696c
x14: 6f6621284f494c4f x13: 0000000000000001 x12: ffff600036d8b97b
x11: 1fffe00036d8b97a x10: ffff600036d8b97a x9 : dfff800000000000
x8 : 00009fffc9274686 x7 : ffff0001b6c5cbd3 x6 : 0000000000000001
x5 : ffff0000c25896c0 x4 : 0000000000000000 x3 : 0000000000000000
x2 : 0000000000000000 x1 : ffff0000c25896c0 x0 : 0000000000000000
Call trace:
 add_to_swap+0xbc/0x158
 shrink_folio_list+0x12ac/0x2648
 shrink_inactive_list+0x318/0x948
 shrink_lruvec+0x450/0x720
 shrink_node_memcgs+0x280/0x4a8
 shrink_node+0x128/0x978
 balance_pgdat+0x4f0/0xb20
 kswapd+0x228/0x438
 kthread+0x214/0x230
 ret_from_fork+0x10/0x20

I can reproduce this issue with the following steps:
```

1) When a dirty swapcache page is isolated by reclaim process and the
   page isn't locked, inject memory failure for the page.
   me_swapcache_dirty() clears uptodate flag and tries to delete from lru,
   but fails. Reclaim process will put the hwpoisoned page back to lru.

2) The process that maps the hwpoisoned page exits, the page is deleted
   the page will never be freed and will be in the lru forever.

3) If we trigger a reclaim again and tries to reclaim the page,
   add_to_swap() will trigger VM_BUG_ON_FOLIO due to the uptodate flag is
   cleared.

To fix it, skip the hwpoisoned page in shrink_folio_list(). Besides, the
hwpoison folio may not be unmapped by hwpoison_user_mappings() yet, unmap
it in shrink_folio_list(), otherwise the folio will fail to be unmaped by
hwpoison_user_mappings() since the folio isn't in lru list.

Link: https://lkml.kernel.org/r/20250318083939.987651-3-tujinjiang@huawei.com
Signed-off-by: Jinjiang Tu <tujinjiang@huawei.com>
Acked-by: Miaohe Lin <linmiaohe@huawei.com>
Cc: David Hildenbrand <david@redhat.com>
Cc: Kefeng Wang <wangkefeng.wang@huawei.com>
Cc: Nanyong Sun <sunnanyong@huawei.com>
Cc: Naoya Horiguchi <nao.horiguchi@gmail.com>
Cc: <stable@vger,kernel.org>
Signed-off-by: Andrew Morton <akpm@linux-foundation.org>

Diffstat

-rw-r--r-- mm/vmscan.c 7

1 files changed, 7 insertions, 0 deletions

```
diff --git a/mm/vmscan.c b/mm/vmscan.c
index 98e6ac82e42861..2b2ab386cab55d 100644
--- a/mm/vmscan.c
+++ b/mm/vmscan.c
@@ -1127,6 +1127,13 @@ retry:
                if (!folio_trylock(folio))
                        goto keep;

+               if (folio_contain_hwpoisoned_page(folio)) {
+                       unmap_poisoned_folio(folio, folio_pfn(folio), false);
+                       folio_unlock(folio);
+                       folio_put(folio);
+                       continue;
+               }
+
                VM_BUG_ON_FOLIO(folio_test_active(folio), folio);

                nr_pages = folio_nr_pages(folio);
```