| | | |
|---|---|---|
| author | Thomas Hellström <thomas.hellstrom@linux.intel.com> | 2025-03-26 16:16:34 +0100 |
| committer | Greg Kroah-Hartman <gregkh@linuxfoundation.org> | 2025-04-25 10:48:01 +0200 |
| commit | 28477f701b63922ff88e9fb13f5519c11cd48b86 (patch) | |
| tree | 8d8c136fff73143497d60c6a80eabdcacfee8680 | |
| parent | b2c11fea2680457cf1290a449cd3f83e562d229f (diff) | |
| download | linux-28477f701b63922ff88e9fb13f5519c11cd48b86.tar.gz | |




## drm/xe: Fix an out-of-bounds shift when invalidating TLB

```
commit 7bcfeddb36b77f9fe3b010bb0b282b7618420bba upstream.

When the size of the range invalidated is larger than
rounddown_pow_of_two(ULONG_MAX),
The function macro roundup_pow_of_two(length) will hit an out-of-bounds
shift [1].

Use a full TLB invalidation for such cases.
v2:
- Use a define for the range size limit over which we use a full
  TLB invalidation. (Lucas)
- Use a better calculation of the limit.

[1]:
[   39.202421] ------------[ cut here ]------------
[   39.202657] UBSAN: shift-out-of-bounds in ./include/linux/log2.h:57:13
[   39.202673] shift exponent 64 is too large for 64-bit type 'long unsigned int'
[   39.202688] CPU: 8 UID: 0 PID: 3129 Comm: xe_exec_system_ Tainted: G     U             6.14.0+ #10
[   39.202690] Tainted: [U]=USER
[   39.202690] Hardware name: ASUS System Product Name/PRIME B560M-A AC, BIOS 2001 02/01/2023
[   39.202691] Call Trace:
[   39.202692]  <TASK>
[   39.202695]  dump_stack_lvl+0x6e/0xa0
[   39.202699]  ubsan_epilogue+0x5/0x30
[   39.202701]  __ubsan_handle_shift_out_of_bounds.cold+0x61/0xe6
[   39.202705]  xe_gt_tlb_invalidation_range.cold+0x1d/0x3a [xe]
[   39.202800]  ? find_held_lock+0x2b/0x80
[   39.202803]  ? mark_held_locks+0x40/0x70
[   39.202806]  xe_svm_invalidate+0x459/0x700 [xe]
[   39.202897]  drm_gpusvm_notifier_invalidate+0x4d/0x70 [drm_gpusvm]
[   39.202900]  __mmu_notifier_release+0x1f5/0x270
[   39.202905]  exit_mmap+0x40e/0x450
[   39.202912]  __mmput+0x45/0x110
[   39.202914]  exit_mm+0xc5/0x130
[   39.202916]  do_exit+0x21c/0x500
[   39.202918]  ? lockdep_hardirqs_on_prepare+0xdb/0x190
[   39.202920]  do_group_exit+0x36/0xa0
[   39.202922]  get_signal+0x8f8/0x900
[   39.202926]  arch_do_signal_or_restart+0x35/0x100
[   39.202930]  syscall_exit_to_user_mode+0x1fc/0x290
[   39.202932]  do_syscall_64+0xa1/0x180
[   39.202934]  ? do_user_addr_fault+0x59f/0x8a0
[   39.202937]  ? lock_release+0xd2/0x2a0
[   39.202939]  ? do_user_addr_fault+0x5a9/0x8a0
[   39.202942]  ? trace_hardirqs_off+0x4b/0xc0
```

```
[   39.202944]  ? clear_bhb_loop+0x25/0x80
[   39.202946]  ? clear_bhb_loop+0x25/0x80
[   39.202947]  ? clear_bhb_loop+0x25/0x80
[   39.202950]  entry_SYSCALL_64_after_hwframe+0x76/0x7e
[   39.202952] RIP: 0033:0x7fa945e543e1
[   39.202961] Code: Unable to access opcode bytes at 0x7fa945e543b7.
[   39.202962] RSP: 002b:00007ffca8fb4170 EFLAGS: 00000293
[   39.202963] RAX: 000000000000003d RBX: 0000000000000000 RCX: 00007fa945e543e3
[   39.202964] RDX: 0000000000000000 RSI: 00007ffca8fb41ac RDI: 00000000ffffffff
[   39.202964] RBP: 00007ffca8fb4190 R08: 0000000000000000 R09: 00007fa945f600a0
[   39.202965] R10: 0000000000000000 R11: 0000000000000293 R12: 0000000000000000
[   39.202966] R13: 00007fa9460dd310 R14: 00007ffca8fb41ac R15: 0000000000000000
[   39.202970]  </TASK>
[   39.202970] ---[ end trace ]---
```

```
Fixes: 332dd0116c82 ("drm/xe: Add range based TLB invalidations")
Cc: Matthew Brost <matthew.brost@intel.com>
Cc: Rodrigo Vivi <rodrigo.vivi@intel.com>
Cc: <stable@vger.kernel.org> # v6.8+
Signed-off-by: Thomas Hellström <thomas.hellstrom@linux.intel.com>
Reviewed-by: Lucas De Marchi <lucas.demarchi@intel.com> #v1
Link: https://lore.kernel.org/r/20250326151634.36916-1-thomas.hellstrom@linux.intel.com
(cherry picked from commit b88f48f86500bc0b44b4f73ac66d500a40d320ad)
Signed-off-by: Lucas De Marchi <lucas.demarchi@intel.com>
Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
```

**Diffstat**

-rw-r--r-- drivers/gpu/drm/xe/xe_gt_tlb_invalidation.c 12

1 files changed, 10 insertions, 2 deletions

```
diff --git a/drivers/gpu/drm/xe/xe_gt_tlb_invalidation.c b/drivers/gpu/drm/xe/xe_gt_tlb_invalidation.c
index ace1fe831a7b72..98a450271f5cee 100644
--- a/drivers/gpu/drm/xe/xe_gt_tlb_invalidation.c
+++ b/drivers/gpu/drm/xe/xe_gt_tlb_invalidation.c
@@ -310,6 +310,13 @@ int xe_gt_tlb_invalidation_ggtt(struct xe_gt *gt)
        return 0;
 }

+/*
+ * Ensure that roundup_pow_of_two(length) doesn't overflow.
+ * Note that roundup_pow_of_two() operates on unsigned long,
+ * not on u64.
+ */
+#define MAX_RANGE_TLB_INVALIDATION_LENGTH (rounddown_pow_of_two(ULONG_MAX))
+
 /**
  * xe_gt_tlb_invalidation_range - Issue a TLB invalidation on this GT for an
  * address range
@@ -334,6 +341,7 @@ int xe_gt_tlb_invalidation_range(struct xe_gt *gt,
        struct xe_device *xe = gt_to_xe(gt);
 #define MAX_TLB_INVALIDATION_LEN       7
        u32 action[MAX_TLB_INVALIDATION_LEN];
+       u64 length = end - start;
        int len = 0;

        xe_gt_assert(gt, fence);
@@ -346,11 +354,11 @@ int xe_gt_tlb_invalidation_range(struct xe_gt *gt,

        action[len++] = XE_GUC_ACTION_TLB_INVALIDATION;
        action[len++] = 0; /* seqno, replaced in send_tlb_invalidation */
-       if (!xe->info.has_range_tlb_invalidation) {
+       if (!xe->info.has_range_tlb_invalidation ||
+           length > MAX_RANGE_TLB_INVALIDATION_LENGTH) {
                action[len++] = MAKE_INVAL_OP(XE_GUC_TLB_INVAL_FULL);
```

```
        } else {
                u64 orig_start = start;
-               u64 length = end - start;
                u64 align;

                if (length < SZ_4K)
```