author	Janusz Krzysztofik <janusz.krzysztofik@linux.intel.com>	2023-03-02 13:08:20 +0100
committer	Greg Kroah-Hartman <gregkh@linuxfoundation.org>	2023-03-22 13:30:05 +0100
commit	5e784a7d07af42057c0576fb647b482f4cb0dc2c (patch)	
tree	e10ee661c30a7fdfa4e2dff87153830d300e8e94	
parent	8f27d4321700687096 54da9ce7e9e8f3aa2f6b8d (diff)	
download	linux-5e784a7d07af42057c0576fb647b482f4cb0dc2c.tar.gz	

drm/i915/active: Fix misuse of non-idle barriers as fence trackers

commit e0e6b416b25ee14716f3549e0cbec1011b193809 upstream.

Users reported oopses on list corruptions when using i915 perf with a number of concurrently running graphics applications. Root cause analysis pointed at an issue in barrier processing code -- a race among perf open / close replacing active barriers with perf requests on kernel context and concurrent barrier preallocate / acquire operations performed during user context first pin / last unpin.

When adding a request to a composite tracker, we try to reuse an existing fence tracker, already allocated and registered with that composite. The tracker we obtain may already track another fence, may be an idle barrier, or an active barrier.

If the tracker we get occurs a non-idle barrier then we try to delete that barrier from a list of barrier tasks it belongs to. However, while doing that we don't respect return value from a function that performs the barrier deletion. Should the deletion ever fail, we would end up reusing the tracker still registered as a barrier task. Since the same structure field is reused with both fence callback lists and barrier tasks list, list corruptions would likely occur.

Barriers are now deleted from a barrier tasks list by temporarily removing the list content, traversing that content with skip over the node to be deleted, then populating the list back with the modified content. Should that intentionally racy concurrent deletion attempts be not serialized, one or more of those may fail because of the list being temporary empty.

Related code that ignores the results of barrier deletion was initially introduced in v5.4 by commit d8af05ff38ae ("drm/i915: Allow sharing the idle-barrier from other kernel requests"). However, all users of the barrier deletion routine were apparently serialized at that time, then the issue didn't exhibit itself. Results of git bisect with help of a newly developed igt@gem_barrier_race@remote-request IGT test indicate that list corruptions might start to appear after commit 311770173fac ("drm/i915/gt: Schedule request retirement when timeline idles"), introduced in v5.5.

Respect results of barrier deletion attempts -- mark the barrier as idle only if successfully deleted from the list. Then, before proceeding with setting our fence as the one currently tracked, make sure that the tracker we've got is not a non-idle barrier. If that check fails then don't use that tracker but go back and try to acquire a new, usable one.

v3: use unlikely() to document what outcome we expect (Andi),
  - fix bad grammar in commit description.
v2: no code changes,
  - blame commit 311770173fac ("drm/i915/gt: Schedule request retirement when timeline idles"), v5.5, not commit d8af05ff38ae ("drm/i915: Allow sharing the idle-barrier from other kernel requests"), v5.4,

```
  - reword commit description.

Closes: https://gitlab.freedesktop.org/drm/intel/-/issues/6333
Fixes: 311770173fac ("drm/i915/gt: Schedule request retirement when timeline idles")
Cc: Chris Wilson <chris@chris-wilson.co.uk>
Cc: stable@vger.kernel.org # v5.5
Cc: Andi Shyti <andi.shyti@linux.intel.com>
Signed-off-by: Janusz Krzysztofik <janusz.krzysztofik@linux.intel.com>
Reviewed-by: Andi Shyti <andi.shyti@linux.intel.com>
Signed-off-by: Andi Shyti <andi.shyti@linux.intel.com>
Link: https://patchwork.freedesktop.org/patch/msgid/20230302120820.48740-1-janusz.krzysztofik@linux.intel.com
(cherry picked from commit 506006055769b10d1b2b4e22f636f3b45e0e9fc7)
Signed-off-by: Jani Nikula <jani.nikula@intel.com>
Signed-off-by: Janusz Krzysztofik <janusz.krzysztofik@linux.intel.com>
Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
```

Diffstat

-rw-r--r-- drivers/gpu/drm/i915/i915_active.c 24

1 files changed, 13 insertions, 11 deletions

```
diff --git a/drivers/gpu/drm/i915/i915_active.c b/drivers/gpu/drm/i915/i915_active.c
index c4c2d24dc50948..0532a5069c04b6 100644
--- a/drivers/gpu/drm/i915/i915_active.c
+++ b/drivers/gpu/drm/i915/i915_active.c
@@ -432,8 +432,7 @@ replace_barrier(struct i915_active *ref, struct i915_active_fence *active)
          * we can use it to substitute for the pending idle-barrer
          * request that we want to emit on the kernel_context.
          */
-        __active_del_barrier(ref, node_from_active(active));
-        return true;
+        return __active_del_barrier(ref, node_from_active(active));
 }

 int i915_active_ref(struct i915_active *ref, u64 idx, struct dma_fence *fence)
@@ -446,16 +445,19 @@ int i915_active_ref(struct i915_active *ref, u64 idx, struct dma_fence *fence)
         if (err)
                 return err;

-        active = active_instance(ref, idx);
-        if (!active) {
-                err = -ENOMEM;
-                goto out;
-        }
+        do {
+                active = active_instance(ref, idx);
+                if (!active) {
+                        err = -ENOMEM;
+                        goto out;
+                }
+
+                if (replace_barrier(ref, active)) {
+                        RCU_INIT_POINTER(active->fence, NULL);
+                        atomic_dec(&ref->count);
+                }
+        } while (unlikely(is_barrier(active)));

-        if (replace_barrier(ref, active)) {
-                RCU_INIT_POINTER(active->fence, NULL);
-                atomic_dec(&ref->count);
-        }
         if (!__i915_active_fence_set(active, fence))
                 __i915_active_acquire(ref);
```