author	Omar Sandoval <osandov@fb.com>	2025-04-25 01:51:24 -0700
committer	Greg Kroah-Hartman <gregkh@linuxfoundation.org>	2025-05-02 08:01:44 +0200
commit	50a665496881262519f115f1bfe5822f30580eb0 (patch)	
tree	bfe74104ab3b663f3be4b5a0483b502a807e5154	
parent	ea1fd673d95eca322d2f4fc1de6d41280a13a0ee (diff)	
download	linux-50a665496881262519f115f1bfe5822f30580eb0.tar.gz	

sched/eevdf: Fix se->slice being set to U64_MAX and resulting crash

[Upstream commit bbce3de72be56e4b5f68924b7da9630cc89aa1a8]

There is a code path in dequeue_entities() that can set the slice of a sched_entity to U64_MAX, which sometimes results in a crash.

The offending case is when dequeue_entities() is called to dequeue a delayed group entity, and then the entity's parent's dequeue is delayed. In that case:

1. In the if (entity_is_task(se)) else block at the beginning of dequeue_entities(), slice is set to cfs_rq_min_slice(group_cfs_rq(se)). If the entity was delayed, then it has no queued tasks, so cfs_rq_min_slice() returns U64_MAX.
2. The first for_each_sched_entity() loop dequeues the entity.
3. If the entity was its parent's only child, then the next iteration tries to dequeue the parent.
4. If the parent's dequeue needs to be delayed, then it breaks from the first for_each_sched_entity() loop _without updating slice_.
5. The second for_each_sched_entity() loop sets the parent's ->slice to the saved slice, which is still U64_MAX.

This throws off subsequent calculations with potentially catastrophic results. A manifestation we saw in production was:

6. In update_entity_lag(), se->slice is used to calculate limit, which ends up as a huge negative number.
7. limit is used in se->vlag = clamp(vlag, -limit, limit). Because limit is negative, vlag > limit, so se->vlag is set to the same huge negative number.
8. In place_entity(), se->vlag is scaled, which overflows and results in another huge (positive or negative) number.
9. The adjusted lag is subtracted from se->vruntime, which increases or decreases se->vruntime by a huge number.
10. pick_eevdf() calls entity_eligible()/vruntime_eligible(), which incorrectly returns false because the vruntime is so far from the other vruntimes on the queue, causing the (vruntime - cfs_rq->min_vruntime) * load calulation to overflow.
11. Nothing appears to be eligible, so pick_eevdf() returns NULL.
12. pick_next_entity() tries to dereference the return value of pick_eevdf() and crashes.

Dumping the cfs_rq states from the core dumps with drgn showed tell-tale huge vruntime ranges and bogus vlag values, and I also traced se->slice being set to U64_MAX on live systems (which was usually "benign" since

the rest of the runqueue needed to be in a particular state to crash).

Fix it in dequeue_entities() by always setting slice from the first
non-empty cfs_rq.

Fixes: aef6987d8954 ("sched/eevdf: Propagate min_slice up the cgroup hierarchy")
Signed-off-by: Omar Sandoval <osandov@fb.com>
Signed-off-by: Peter Zijlstra (Intel) <peterz@infradead.org>
Signed-off-by: Ingo Molnar <mingo@kernel.org>
Link: https://lkml.kernel.org/r/f0c2d1072be229e1bdddc73c0703919a8b00c652.1745570998.git.osandov@fb.com
Signed-off-by: Sasha Levin <sashal@kernel.org>

Diffstat

-rw-r--r-- kernel/sched/fair.c 4

1 files changed, 1 insertions, 3 deletions

```
diff --git a/kernel/sched/fair.c b/kernel/sched/fair.c
index 89c7260103e18b..3d9b68a347b764 100644
--- a/kernel/sched/fair.c
+++ b/kernel/sched/fair.c
@@ -7083,9 +7083,6 @@ static int dequeue_entities(struct rq *rq, struct sched_entity *se, int flags)
                h_nr_idle = task_has_idle_policy(p);
                if (task_sleep || task_delayed || !se->sched_delayed)
                        h_nr_runnable = 1;
-       } else {
-               cfs_rq = group_cfs_rq(se);
-               slice = cfs_rq_min_slice(cfs_rq);
        }

        for_each_sched_entity(se) {
@@ -7095,6 +7092,7 @@ static int dequeue_entities(struct rq *rq, struct sched_entity *se, int flags)
                        if (p && &p->se == se)
                                return -1;

+                       slice = cfs_rq_min_slice(cfs_rq);
                        break;
                }
```