Subject: Re: PDS/HPDA research projects
From: Julia Lawall <julia.lawall@inria.fr>
Date: 19/09/2023 12:09
To: Gaël Thomas <gael.thomas@inria.fr>
CC: François Trahay <francois.trahay@telecom-sudparis.eu>

OFence: Pairing Barriers to Find Concurrency Bugs in the Linux Kernel

```
static void bfq_group_set_weight(struct bfq_group *bfqg, u64 weight, u64 dev_weight)
{
        weight = dev_weight ?: weight;

        bfqg->entity.dev_weight = dev_weight;
        /*
         * Setting the prio_changed flag of the entity
         * to 1 with new_weight == weight would re-set
         * the value of the weight to its ioprio mapping.
         * Set the flag only if necessary.
         */
        if ((unsigned short)weight != bfqg->entity.new_weight) {
                bfqg->entity.new_weight = (unsigned short)weight;
                /*
                 * Make sure that the above new value has been
                 * stored in bfqg->entity.new_weight before
                 * setting the prio_changed flag. In fact,
                 * this flag may be read asynchronously (in
                 * critical sections protected by a different
                 * lock than that held here), and finding this
                 * flag set may cause the execution of the code
                 * for updating parameters whose value may
                 * depend also on bfqg->entity.new_weight (in
                 * __bfq_entity_update_weight_prio).
                 * This barrier makes sure that the new value
                 * of bfqg->entity.new_weight is correctly
                 * seen in that code.
                 */
                smp_wmb();
                bfqg->entity.prio_changed = 1;
        }
}
```