
Subject: Re: [BUG]: Crash with CONFIG_FAIR_CGROUP_SCHED=y
Posted by [Srivatsa Vaddagiri](#) on Fri, 09 Nov 2007 06:52:40 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Thu, Nov 08, 2007 at 03:48:05PM -0800, sukadev@us.ibm.com wrote:
> With CONFIG_FAIR_CGROUP_SCHED=y, following commands on 2.6.24-rc1 crash
> the system.

Thanks for reporting the problem. It was caused because of the fact that current task isn't kept in its runqueue in case of sched_fair class tasks.

With the patch below, I could run ns_exec w/o any crash. Can you pls verify it works for you as well?

Ingo,
Once Suka verifies that the patch fixes his crash, I would request you to include the same in your tree and route it to Linus.

--

current task is not present in its runqueue in case of sched_fair class tasks. Take care of this fact in `rt_mutex_setprio()`, `sched_setscheduler()` and `sched_move_task()` routines.

Signed-off-by : [Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com>](mailto:Vatsa@linux.vnet.ibm.com)

```
kernel/sched.c | 45 ++++++-----  
1 files changed, 25 insertions(+), 20 deletions(-)
```

Index: current/kernel/sched.c

```
=====
--- current.orig/kernel/sched.c
+++ current/kernel/sched.c
@@ -3986,11 +3986,13 @@ void rt_mutex_setprio(struct task_struct
    oldprio = p->prio;
    on_rq = p->se.on_rq;
    running = task_running(rq, p);
- if (on_rq) {
+ if (on_rq)
    dequeue_task(rq, p, 0);
- if (running)
- p->sched_class->put_prev_task(rq, p);
- }
+ /* current task is not kept in its runqueue in case of sched_fair class.
+ * Hence we need the 'on_rq?' and 'running?' tests to be separate.
```

```

+ */
+ if (running)
+ p->sched_class->put_prev_task(rq, p);

    if (rt_prio(prio))
        p->sched_class = &rt_sched_class;
@@ -3999,9 +4001,9 @@ void rt_mutex_setprio(struct task_struct

    p->prio = prio;

+ if (running)
+ p->sched_class->set_curr_task(rq);
    if (on_rq) {
- if (running)
- p->sched_class->set_curr_task(rq);
        enqueue_task(rq, p, 0);
        inc_load(rq, p);
    /*
@@ -4298,18 +4300,20 @@ recheck:
        update_rq_clock(rq);
        on_rq = p->se.on_rq;
        running = task_running(rq, p);
- if (on_rq) {
+ if (on_rq)
        deactivate_task(rq, p, 0);
- if (running)
- p->sched_class->put_prev_task(rq, p);
- }
+ /* current task is not kept in its runqueue in case of sched_fair class.
+ * Hence we need the 'on_rq?' and 'running?' tests to be separate.
+ */
+ if (running)
+ p->sched_class->put_prev_task(rq, p);

    oldprio = p->prio;
    __setscheduler(rq, p, policy, param->sched_priority);

+ if (running)
+ p->sched_class->set_curr_task(rq);
    if (on_rq) {
- if (running)
- p->sched_class->set_curr_task(rq);
        activate_task(rq, p, 0);
    /*
        * Reschedule if we are currently running on this runqueue and
@@ -7036,19 +7040,20 @@ void sched_move_task(struct task_struct
        running = task_running(rq, tsk);
        on_rq = tsk->se.on_rq;

```

```
- if (on_rq) {
+ if (on_rq)
    dequeue_task(rq, tsk, 0);
- if (unlikely(running))
- tsk->sched_class->put_prev_task(rq, tsk);
- }
+ /* current task is not kept in its runqueue in case of sched_fair class.
+ * Hence we need the 'on_rq?' and 'running?' tests to be separate.
+ */
+ if (unlikely(running))
+ tsk->sched_class->put_prev_task(rq, tsk);

    set_task_cfs_rq(tsk);

- if (on_rq) {
- if (unlikely(running))
- tsk->sched_class->set_curr_task(rq);
+ if (unlikely(running))
+ tsk->sched_class->set_curr_task(rq);
+ if (on_rq)
    enqueue_task(rq, tsk, 0);
- }

done:
    task_rq_unlock(rq, &flags);
```

--
Regards,
vatsa

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
