
Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by [Dmitry Adamushko](#) on Sun, 16 Dec 2007 13:01:17 GMT
[View Forum Message](#) <> [Reply to Message](#)

Ingo,

what about the following patch instead?

maybe `task_is_current()` would be a better name though.

Steven,

I guess, there is some analogue of `UNLOCKED_CTXSW` on `-rt`
(to reduce contention for `rq->lock`).

So there can be a race `schedule()` vs. `rt_mutex_setprio()` or `sched_setscheduler()`
for some paths that might explain crashes you have been observing?

I haven't analyzed this case for `-rt`, so I'm just throwing in the idea in case it can be useful.

From: Dmitry Adamushko <dmitry.adamushko@gmail.com>

sched: introduce `task_current()`

Some services (e.g. `sched_setscheduler()`, `rt_mutex_setprio()` and `sched_move_task()`)
must handle a given task differently in case it's the '`rq->curr`' task on its run-queue.
The `task_running()` interface is not suitable for determining such tasks
for platforms with one of the following options:

```
#define __ARCH_WANT_UNLOCKED_CTXSW  
#define __ARCH_WANT_INTERRUPTS_ON_CTXSW
```

Due to the fact that it makes use of '`p->oncpu == 1`' as a criterion but
such a task is not necessarily '`rq->curr`'.

The detailed explanation is available here:

<https://lists.linux-foundation.org/pipermail/containers/2007-December/009262.html>

Signed-off-by: Dmitry Adamushko <dmitry.adamushko@gmail.com>

```
---  
diff --git a/kernel/sched.c b/kernel/sched.c  
index dc6fb24..15d088b 100644  
--- a/kernel/sched.c  
+++ b/kernel/sched.c
```

```

@@ -619,10 +619,15 @@ EXPORT_SYMBOL_GPL(cpu_clock);
# define finish_arch_switch(prev) do { } while (0)
#endif

+static inline int task_current(struct rq *rq, struct task_struct *p)
+{
+ return rq->curr == p;
+}
+
#ifdef __ARCH_WANT_UNLOCKED_CTXSW
static inline int task_running(struct rq *rq, struct task_struct *p)
{
- return rq->curr == p;
+ return task_current(rq, p);
}

static inline void prepare_lock_switch(struct rq *rq, struct task_struct *next)
@@ -651,7 +656,7 @@ static inline int task_running(struct rq *rq, struct task_struct *p)
#ifdef CONFIG_SMP
return p->oncpu;
#else
- return rq->curr == p;
+ return task_current(rq, p);
#endif
}

@@ -3340,7 +3345,7 @@ unsigned long long task_sched_runtime(struct task_struct *p)

rq = task_rq_lock(p, &flags);
ns = p->se.sum_exec_runtime;
- if (rq->curr == p) {
+ if (task_current(rq, p)) {
update_rq_clock(rq);
delta_exec = rq->clock - p->se.exec_start;
if ((s64)delta_exec > 0)
@@ -4033,7 +4038,7 @@ void rt_mutex_setprio(struct task_struct *p, int prio)

oldprio = p->prio;
on_rq = p->se.on_rq;
- running = task_running(rq, p);
+ running = task_current(rq, p);
if (on_rq) {
dequeue_task(rq, p, 0);
if (running)
@@ -4334,7 +4339,7 @@ recheck:
}
update_rq_clock(rq);
on_rq = p->se.on_rq;

```

```

- running = task_running(rq, p);
+ running = task_current(rq, p);
  if (on_rq) {
    deactivate_task(rq, p, 0);
    if (running)
@@ -7360,7 +7365,7 @@ void sched_move_task(struct task_struct *tsk)

  update_rq_clock(rq);

- running = task_running(rq, tsk);
+ running = task_current(rq, tsk);
  on_rq = tsk->se.on_rq;

  if (on_rq) {

```

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