Subject: Re: [BUG]: Crash with CONFIG_FAIR_CGROUP_SCHED=y
Posted by Srivatsa Vaddagiri on Fri, 09 Nov 2007 10:04:38 GMT
View Forum Message <> Reply to Message

On Fri, Nov 09, 2007 at 09:45:21AM +0100, Dmitry Adamushko wrote:
> Humm... the 'current' is not kept within the tree but
> current->se.on_rq is supposed to be '1' ,
> so the old code looks ok to me (at least for the 'leaf' elements).

You are damned right! Sorry my mistake with the previous analysis and
(as I now find out) testing :(

There are couple of problems discovered by Suka's test:

- The test requires the cgroup filesystem to be mounted with
  atleast the cpu and ns options (i.e both namespace and cpu
  controllers are active in the same hierarchy).

  # mkdir /dev/cpuctl
  # mount -t cgroup -ocpu,ns none cpuctl
  (or simply)
  # mount -t cgroup none cpuctl -> Will activate all controllers
      in same hierarchy.

- The test invokes clone() with CLONE_NEWNS set. This causes a a new child
  to be created, also a new group (do_fork->copy_namespaces->ns_cgroup_clone->
  cgroup_clone) and the child is attached to the new group (cgroup_clone->
  attach_task->sched_move_task). At this point in time, the child's scheduler
  related fields are uninitialized (including its on_rq field, which it has
  inherited from parent). As a result sched_move_task thinks its on
  runqueue, when it isn't.

  As a solution to this problem, I moved sched_fork() call, which
  initializes scheduler related fields on a new task, before
  copy_namespaces(). I am not sure though whether moving up will
  cause other side-effects. Do you see any issue?

- The second problem exposed by this test is that task_new_fair()
  assumes that parent and child will be part of the same group (which
  needn't be as this test shows). As a result, cfs_rq->curr can be NULL
  for the child.

  The solution is to test for curr pointer being NULL in
  task_new_fair().

With the patch below, I could run ns_exec() fine w/o a crash.

Suka, can you verify whether this patch fixes your problem?

--

Fix copy_namespace() <-> sched_fork() dependency in do_fork, by moving
up sched_fork().

Also introduce a NULL pointer check for 'curr' in task_new_fair().

Signed-off-by : Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com>

---
 kernel/fork.c | 6 ++++---
 kernel/sched_fair.c | 2 +-
 2 files changed, 4 insertions(+), 4 deletions(-)

Index: current/kernel/fork.c
===
--- current.orig/kernel/fork.c
+++ current/kernel/fork.c
@@ -1121,6 +1121,9 @@ static struct task_struct *copy_process(
  p->blocked_on = NULL; /* not blocked yet */
 #endif

+ /* Perform scheduler related setup. Assign this task to a CPU. */
+ sched_fork(p, clone_flags);
+
  if ((retval = security_task_alloc(p)))
   goto bad_fork_cleanup_policy;
  if ((retval = audit_alloc(p)))
@@ -1210,9 +1213,6 @@ static struct task_struct *copy_process(
  INIT_LIST_HEAD(&p->ptrace_children);
  INIT_LIST_HEAD(&p->ptrace_list);

- /* Perform scheduler related setup. Assign this task to a CPU. */
- sched_fork(p, clone_flags);
-
  /* Now that the task is set up, run cgroup callbacks if
   * necessary. We need to run them before the task is visible
   * on the tasklist. */
Index: current/kernel/sched_fair.c
===
--- current.orig/kernel/sched_fair.c
+++ current/kernel/sched_fair.c
@@ -1023,7 +1023,7 @@ static void task_new_fair(struct rq *rq,
  place_entity(cfs_rq, se, 1);

```
  if (sysctl_sched_child_runs_first && this_cpu == task_cpu(p) &&
-   curr->vruntime < se->vruntime) {
+   curr && curr->vruntime < se->vruntime) {
    /*
     * Upon rescheduling, sched_class::put_prev_task() will place
     * 'current' within the tree based on its new key value.
```

_____

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