Subject: [PATCH 14/28] [PREP 14/14] Hold the struct pid till flushing the proc trees in release_task()
Posted by Pavel Emelianov on Fri, 15 Jun 2007 16:12:24 GMT
View Forum Message <> Reply to Message

When task is released its pids are detached first and then the proc is flushed.
With the namespaces we have to hold the pid till proc flush to have all the
namespaces to flush the dentries from.

This get_pid()/put_pid() pair introduces some performance overhead on nptl perf
test, so if the pid belongs to init namespace only, we can optimize this out.

Signed-off-by: Pavel Emelianov <xemul@openvz.org>

```
 exit.c |   18 +++++++++++++++++-
 1 files changed, 17 insertions(+), 1 deletion(-)

--- ./kernel/exit.c.flushsave 2007-06-15 15:13:07.000000000 +0400
+++ ./kernel/exit.c 2007-06-15 15:13:45.000000000 +0400
@@ -154,6 +154,7 @@ static void delayed_put_task_struct(stru

 void release_task(struct task_struct * p)
 {
+ struct pid *pid;
  struct task_struct *leader;
  int zap_leader;
 repeat:
@@ -161,6 +162,20 @@ repeat:
  write_lock_irq(&tasklist_lock);
  ptrace_unlink(p);
  BUG_ON(!list_empty(&p->ptrace_list) || !list_empty(&p->ptrace_children));
+ /*
+  * we have to keep this pid till proc_flush_task() to make
+  * it possible to flush all dentries holding it. pid will
+  * be put ibidem
+  *
+  * however if the pid belogs to init namespace only, we can
+  * optimize this out
+  */
+ pid = task_pid(p);
+ if (!pid_ns_accessible(&init_pid_ns, pid))
+  get_pid(pid);
+ else
+  pid = NULL;
+
  __exit_signal(p);
```

```
 /*
@@ -185,7 +200,8 @@ repeat:
 }

 write_unlock_irq(&tasklist_lock);
- proc_flush_task(p, NULL);
+ proc_flush_task(p, pid);
+ put_pid(pid);
 release_thread(p);
 call_rcu(&p->rcu, delayed_put_task_struct);
```

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