Subject: Re: [PATCH 14/15] Destroy pid namespace on init's death
Posted by Oleg Nesterov on Sun, 29 Jul 2007 10:39:47 GMT
View Forum Message <> Reply to Message

On 07/26, Pavel Emelyanov wrote:
>
> @@ -895,6 +915,7 @@ fastcall NORET_TYPE void do_exit(long co
> {
> struct task_struct *tsk = current;
> int group_dead;
> + struct pid_namespace *pid_ns = tsk->nsproxy->pid_ns;
>
> profile_task_exit(tsk);
>
> @@ -905,9 +926,10 @@ fastcall NORET_TYPE void do_exit(long co
> if (unlikely(!tsk->pid))
> panic("Attempted to kill the idle task!");
> if (unlikely(tsk == task_child_reaper(tsk))) {
> - if (task_active_pid_ns(tsk) != &init_pid_ns)
> - task_active_pid_ns(tsk)->child_reaper =
> - init_pid_ns.child_reaper;
> + if (pid_ns != &init_pid_ns) {
> + zap_pid_ns_processes(pid_ns);
> + pid_ns->child_reaper = init_pid_ns.child_reaper;
> + }
> else
> panic("Attempted to kill init!");

No, no, this is wrong. Yes, the current code is buggy too, I'll send
the fix.

I think this code should be moved below under the "if (group_dead)",
and we should use tsk->group_leader.

> +void zap_pid_ns_processes(struct pid_namespace *pid_ns)
> +{
> + int i;
> + int nr;
> + int nfree;
> + int options = WNOHANG|WEXITED|__WALL;
> +
> +repeat:
> + /*
> + * We know pid == 1 is terminating. Find remaining pid_ts
> + * in the namespace, signal them and then wait for them
> + * exit.
> + */
> + nr = next_pidmap(pid_ns, 1);

```
> + while (nr > 0) {
> +  kill_proc_info(SIGKILL, SEND_SIG_PRIV, nr);
> +  nr = next_pidmap(pid_ns, nr);
> + }
> +
> + nr = next_pidmap(pid_ns, 1);
> + while (nr > 0) {
> +  do_wait(nr, options, NULL, NULL, NULL);
```

When the first child of init exits, it sends SIGCHLD. After that,
do_wait() will never sleep, so we are doing a busy-wait loop.
Not good, especially when we have a niced child, can livelock.

```
> +  nr = next_pidmap(pid_ns, nr);
> + }
> +
> + nfree = 0;
> + for (i = 0; i < PIDMAP_ENTRIES; i++)
> +  nfree += atomic_read(&pid_ns->pidmap[i].nr_free);
> +
> + /*
> +  * If pidmap has entries for processes other than 0 and 1, retry.
> +  */
> + if (nfree < (BITS_PER_PAGE * PIDMAP_ENTRIES - 2))
> +  goto repeat;
```

This doesn't look right.

Suppose that some "struct pid" was pinned from the parent namespace.
In that case zap_pid_ns_processes() will burn CPU until put_pid(), bad.

I think we can rely on forget_original_child() and do something like
this:

```
 zap_active_ns_processes(void)
 {
  // kill all tasks in our ns and below
  kill(-1, SIGKILL);

  do {
   clear_thread_flag(TIF_SIGPENDING);
  } while (wait(NULL) != -ECHLD);
 }
```

Oleg.