Subject: Re: Getting the new RxRPC patches upstream
Posted by Oleg Nesterov on Tue, 24 Apr 2007 19:34:04 GMT
View Forum Message <> Reply to Message

On 04/24, David Howells wrote:
>
> Oleg Nesterov <oleg@tv-sign.ru> wrote:
>
> > Sure, I'll grep for cancel_delayed_work(). But unless I missed something,
> > this change should be completely transparent for all users. Otherwise, it
> > is buggy.
>
> I guess you will have to make sure that cancel_delayed_work() is always
> followed by a flush of the workqueue, otherwise you might get this situation:
>
```
>  CPU 0    CPU 1
>  =============================== ======================
>     <timer expires>
>  cancel_delayed_work(x) == 0 -->delayed_work_timer_fn(x)
>  kfree(x);   -->do_IRQ()
>  y = kmalloc(); // reuses x
>     <--do_IRQ()
>     __queue_work(x)
>  --- OOPS ---
```
>
> That's my main concern. If you are certain that can't happen, then fair
> enough.

Yes sure. Note that this is documented:

```
 /*
  * Kill off a pending schedule_delayed_work().  Note that the work callback
  * function may still be running on return from cancel_delayed_work().  Run
  * flush_workqueue() or cancel_work_sync() to wait on it.
  */
```

This comment is not very precise though. If the work doesn't re-arm itself,
we need cancel_work_sync() only if cancel_delayed_work() returns 0.

So there is no difference with the proposed change. Except, return value == 0
means:

currently (del_timer_sync): callback may still be running or scheduled

with del_timer: may still be running, or scheduled, or will be scheduled
right now.

However, this is the same from the caller POV.

```
> Can you show me a patch illustrating exactly how you want to change
> cancel_delayed_work()?  I can't remember whether you've done so already, but
> if you have, I can't find it.  Is it basically this?:
>
>  static inline int cancel_delayed_work(struct delayed_work *work)
>  {
>   int ret;
>
> - ret = del_timer_sync(&work->timer);
> + ret = del_timer(&work->timer);
>   if (ret)
>    work_release(&work->work);
>   return ret;
>  }
```

Yes, exactly. The patch is trivial, but I need some time to write the
understandable changelog...

```
> I was thinking this situation might be a problem:
>
>  CPU 0    CPU 1
>  ================================ =======================
>     <timer expires>
>  cancel_delayed_work(x) == 0 -->delayed_work_timer_fn(x)
>  schedule_delayed_work(x,0) -->do_IRQ()
>  <keventd scheduled>
>  x->work()
>     <--do_IRQ()
>     __queue_work(x)
>
> But it won't, will it?
```

Yes, I think this should be OK. schedule_delayed_work() will notice
_PENDING and abort, so the last "x->work()" doesn't happen.

What can happen is

```
    <timer expires>
 cancel_delayed_work(x) == 0
    -->delayed_work_timer_fn(x)
    __queue_work(x)
    <keventd scheduled>
    x->work()
 schedule_delayed_work(x,0)
 <the work is scheduled again>
```

, so we can have an "unneeded schedule", but this is very unlikely.

Oleg.