Subject: Re: Getting the new RxRPC patches upstream
Posted by David Howells on Tue, 24 Apr 2007 15:51:07 GMT
View Forum Message <> Reply to Message

Oleg Nesterov <oleg@tv-sign.ru> wrote:

> Let's look at (2). cancel_delayed_work() (on top of del_timer()) returns 0,
> and this is correct, we failed to cancel the timer, and we don't know whether
> work->func() finished, or not.

Yes.

> The current code uses del_timer_sync(). It will also return 0. However, it
> will spin waiting for timer->function() to complete. So we are just wasting
> CPU.

That's my objection to using cancel_delayed_work() as it stands, although in most cases it's a relatively minor waste of time. However, if the timer expiry routine gets interrupted then it may not be so minor... So, yes, I'm in full agreement with you there.

> I guess I misunderstood you. Perhaps, you propose a new helper which use
> try_to_del_timer_sync(), yes? Unless I missed something, this doesn't help.
> Because the return value == -1 should be treated as 0. We failed to stop
> the timer, and we can't free dwork.

Consider how I'm using try_to_cancel_delayed_work(): I use this when I want to queue a delayed work item with a particular timeout (usually for immediate processing), but the work item may already be pending.

If try_to_cancel_delayed_work() returns 0 or 1 (not pending or pending but dequeued) then I can go ahead and just schedule the work item (I'll be holding a lock to prevent anyone else from interfering).

However, if try_to_cancel_delayed_work() returns -1 then there's no usually no point attempting to schedule the work item because I know the timer expiry handler is doing that or going to do that.

The code looks like this in pretty much all cases:

```
 if (try_to_cancel_delayed_work(&afs_server_reaper) >= 0)
  schedule_delayed_work(&afs_server_reaper, 0);
```

And so could well be packaged into a convenience routine and placed in workqueue.[ch]. However, this would still concern Dave Miller as my patches would still be altering non-net stuff or depending on non-net patches he doesn't have in his GIT tree.

Using cancel_delayer_work() instead would be acceptable, functionally, as that just waits till the -1 return case no longer holds true, and so always returns 0 or 1.

In RxRPC, this is only used to cancel a pair global delayed work items in the rmmod path, and so the inefficiency of cancel_delayed_work() is something I can live with, though it's something I'd want to reduce in the longer term.

In AFS, this is not only used in object destruction paths, but is also used to cancel the callback timer and initiate synchronisation processing with immediate effect.

David

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