Subject: Re: [PATCH] nfs lockd reclaimer: Convert to kthread API
Posted by Trond Myklebust on Thu, 19 Apr 2007 22:04:44 GMT
View Forum Message <> Reply to Message

On Thu, 2007-04-19 at 14:40 -0700, Andrew Morton wrote:
> Using signals to communicate with kernel threads is fairly unpleasant, IMO.
> We have much simpler, faster and more idiomatic ways of communicating
> between threads in-kernel and there are better ways in which userspace can
> communicate with the kernel - system calls, for example...
>
> So I think generally any move which gets us away from using signals in
> kernel threads is moving in a good direction.

I have yet to see a proposal which did. Eric's patch was eliminating
signals in kernel threads that used them without proposing any
replacement mechanism or showing that he had plans to do so. That is a
good reason for a veto.

> > > With pid namespaces all kernel threads will disappear so how do
> > > we cope with the problem when the sysadmin can not see the kernel
> > > threads?
> >
> > Then you have a usability problem. How does the sysadmin reboot the
> > system if there is no way to shut down the processes that are hanging on
> > an unresponsive filesystem?
>
> Where's the hang? A user process is stuck on h_rwsem?
>
> If so, would it be appropriate to convert the user process to use
> down_foo_interruptible(), so that the operator can just kill the user
> process as expected, rather than having to futz around killing kernel
> threads?

If an NFS server reboots, then the locks held by user processes on the
client need to be re-established by when it comes up again. Otherwise,
the processes that thought they were holding locks will suddenly fail.
This recovery job is currently the done by a kernel thread.

The question is then what to do if the server crashes again while the
kernel thread is re-establishing the locks. Particularly if it never
comes back again.
Currently, the administrator can intervene by killing anything that has
open files on that volume and kill the recovery kernel thread.
You'll also note that lockd_down(), nfsd_down() etc all use signals to
inform lockd(), nfsd() etc that they should be shutting down. Since the
reclaimer thread is started by the lockd() thread using CLONE_SIGHAND,
this means that we also automatically kill any lingering recovery
threads whenever we shutdown lockd().

These mechanisms need to be replaced _before_ we start shooting down
sigallow() etc in the kernel.

  Trond
