Subject: Re: [RFC][PATCH 0/2] pipe: checkpoint and restart for pipe; brief description and test interface
Posted by Masahiko Takahashi on Fri, 02 Feb 2007 00:32:47 GMT
View Forum Message <> Reply to Message

Hi Serge,

On Wed, 2007-01-31 at 09:11 -0600, Serge E. Hallyn wrote:
> Quoting Eric W. Biederman (ebiederm@xmission.com):
> > Rewind a little bit please. I can clearly see where checkpoint/restart
> > fit into the containers conversation.
> >
> > However I don't see where this checkpoint/restart patchset fits into
> > the checkpoint restart picture.
>
> I agree without a basic infrastructure to initiate checkpoints it
> *feels* a little out of place, but presumably we can test this with any
> of the existing c/r solutions out there?

That is my intention. My patch wouldn't mention to API or
framework yet. Test/debug interface in [PATCH 0/2] is
only a sample interface to be tested from userlevel.

> > I will say that I am not yet convinced that we need the kernel saving
> > and restoring the checkpoint.
>
> Some things will make sense to just do in userspace, but other things,
> i.e. certainly remapped memory, and, perhaps pipes, should have help from
> the kernel.
>
> Or can the program driving the checkpoint just cat the pipe fd to get
> the unread contents out to save in the checkpoint? If not, an
> alternative might be to just add an f_op 'checkpoint', which for pipes
> spits out unread contents, and for ordinary files just spits out pos
> and flags... Though one problem with that is how to tell which two
> processes were sharing the pipe? pipefile_fops->checkpoint() spits out
> the pipefs inode number?

Thank you, Serge. I agree with your opinion. Some c/r things
can be done only in userlevel but others require kernel
support. After I read comments, I guess pipe c/r is not a
good example to prove that some c/r functionality really
needs help from kernel. We can get some information from
/proc/PID/fd but this is not enough. As Serge and Cedric
mentioned, sharing file descriptor is one of difficult
problems to solve if checkpointing is implemented only in
userlevel. For example, consider two file descriptors
opening a same ordinary file. From /proc's view, we can

get only its filename. Can we distinguish between shared
file descriptors and two descriptors just opening the same
file independently ?

I believe some c/r components (maybe not pipe) implemented
in kernel level could be so simple and sophisticated.

> > Regardless of the approach when we start seriously discussing checkpoint/restart
> > one of the big issues is how do we maintain an ABI to user space for all of
> > the data so we can migrate applications across kernel version.
>
> I had planned on waiting until pidspace was more complete to start that
> discussion, but by all means lets start talking about a preferred
> mechanism and api now.

First of all, let me know an assumption. Is there some
super process (or daemon) to control all checkpoint and
restart in that environment ? What functionality does it
have ?

Thanks,

Masahiko.