Subject: Re: [PATCH 0/13] Pid namespaces (OpenVZ view)
Posted by serue on Fri, 25 May 2007 14:25:50 GMT
View Forum Message <> Reply to Message

Quoting Pavel Emelianov (xemul@openvz.org):
> Serge E. Hallyn wrote:
> > Quoting Pavel Emelianov (xemul@openvz.org):
> >> Serge E. Hallyn wrote:
> >>> Quoting Eric W. Biederman (ebiederm@xmission.com):
> >>>> "Serge E. Hallyn" <serue@us.ibm.com> writes:
> >>>>
> >>>>> 3. Cleaner logic for namespace migration: with this approach
> >>>>>   one need to save the virtual pid and let global one change;
> >>>>>   with Suka's logic this is not clear how to migrate the level
> >>>>>   2 namespace (concerning init to be level 0).
> >>>> This is a very good point.
> >>>>
> >>>> How *would* we migrate the pids at the second level?
> >>> As long as you don't try and restore pids into the initial pid namespace
> >>> it isn't a problem. You just record the pid hierarchy and the pid
> >>> for a task in that hierarchy. There really is nothing special going on
> >>> that should make migration hard.
> >>>
> >>> Or did I miss something?
> >> Hmm, no, i guess you are right. I was thinking that getting the pid for
> >> a process woudl be done purely from userspace, but I guess along with a
> >> kernel helper to *set* pids, we could also have a kernel helper to get
> >> all pids for all pid namespaces "above" that of the process doing the
> >> checkpoint.
> > So do you agree that if we migrate a VS we need to migrate the whole VS?
> >
> > I started to respond, then realized you were probably asking something
> > different than I thought. My original response is below, but here is I
> > think the answer to your question, which is important because I think
> > your question might highlight a misunderstanding about the design of
> > Suka's code.
> >
> > Let's say a vserver is started, and in there a pidns is started for a
> > checkpoint/restart job. So let's say we have PID 13 in the root
> > namespace starting PID 14 in a new namespace. So using (pid, pid_ns) as
> > the terminology, we havd (13,1) as the parent process, and (14,1)=(1,2)
> > as the init of the vserver. Let's ignore other tasks inthe vserver, and
> > just talk about (1402,2) as the init of the checkpoint restart job, so
> > it is (1402,2)=(1,3). And oh, yeah, (1402,2)=(1,3)=(2309,1).
>
> Oh, this is heavy... Lets draw some diagrams.
>
> You have a vserver with a namespace in it with a cpt job in it,

> just like this:
>
> [node. pids look like (N)]
> `- [vserver. pids look like (N,V)]
> `- [cpt job. pids look like (N,V,P)]
>
> Is that OK?

It's different from the notation I was using.

Let's stick to calling every process by a full "upid", i.e.
(pid, pid namespace #) because while it's longer it gives more
information.

> We have task in "node" with pid (13) which spawns the task with
> pid (14,1) into the "vserver", like this:
>
> (13)
> `- (14,1)
>
> If so, then what the notion (14,1)=(1,2) mean?

It means that (pid 14, pid_ns 1) = (pid 1, pid_ns 2). It describes one
task, which in pid namespace 1 is known by pid 14, and in pid namespace
2 is known by pid 1.

(I see the repetative low numbers were confusing...)

> As far as the "cpt job" is concerned we have smth like this:
>
> (13)
> `- (14,1)
> `- (1402,2,1)
>
> where (1402,2,1) is the root of the "cpt job", right?

Sure, and in my notation this would be

```
[(13,1)]
  `- [(14,1)(1,2)]
    `- [(2309,1)(1402,2)(1,3)]
```

Again each level is just one task, but known by several pids.

So coming back to the idea of checkpoint all of pid_ns=2, we would be
checkpointing both task [(14,1)(1,2)] and task [(2309,1)(1402,2)(1,3)].
And my question had been how would we access and store the fact that the
third task has pid (1,3), which we MUST store and reset, because that is

that task's active pid namespace, meaning it only knows itself as (1,3).

The task in pid namespace 2 which is doing the checkpointing generally
only knows the third task as (1402,2), so we need to provide a mechanism
for it to dump all pids in "higher" pid namespaces.

Note that, of course, pids in "lower" pid namespaces can be randomly
set. If we are restarting pid namespace 2 on a new system, it's
perfeclty ok for the pids to look like:

```
  [(467,1)]
    `- [(5597,1)(1,2)]
      `- [(5598,1)(1402,2)(1,3)]
```

Heh, or even

```
  [(14,1)(467,2)]
    `- [(444,1)(5597,2)(1,3)]
      `- [(445,1)(5598,2)(1402,3)(1,4)]
```

thanks,
-serge

> > Now when we want to migrate the vserver, a task in pid_ns 2 will look
> > for all tasks with pids in pidns 2. That will automatically include all
> > tasks in pid_ns 3. I think you thought I was asking how we would
> > include pid_ns 3, and are asking whether it woudl be ok to not migrate
> > pid_ns 3? (answer: it's irrelevant, all tasks in pid_ns 3 are also in
> > pid_ns 2 - and in pid_ns 1).
> >
> > What I was actually asking was, in the same situation, how would the
> > task in pid_ns 2 doing the checkpoint get the pids in pid_ns 3. So it
> > sees the task as (1402,2), but needs to also store (1,3) and, on
> > restart, recreate a task with both those pids.
> >
> > But I guess it will be pretty simple, and fall into place once we get
> > c/r semantics started.
> >
> > thanks,
> > -serge
> >
> > [original response]
> >
> > I think that's the reasonable thing for people to do, but I don't think
> > we should force them to. I.e. there is no reason you shouldn't be able
> > to take one or two tasks out of a pidns and checkpoint them, and restart
> > them elsewhere. If it turns out they were talking to a third process
> > which wasn't checkpointed, well, too bad.

> >
> > What you are more likely to need is a new clean set of namespaces to
> > restart in, but again I don't think we should enforce that. So whatever
> > mechanism we end up doing to implementing "clone_with_pid()", we should
> > handle -EBUSY correctly.
> >
> > Anyway, why do you ask? (How does it follow from the conversation?)
> >
> > I wasn't suggesting that it would be ok to only dump part of the pid
> > information, rather I was asking how we would do it correctly :)
> >