Subject: Re: [RFC][PATCH 5/7] VPIDs: vpid/pid conversion in VPID enabled case
Posted by Alexey Kuznetsov on Mon, 06 Feb 2006 09:48:43 GMT
View Forum Message <> Reply to Message

Hello!

> > 1. Replacing all the references to pid with pair (container, pid) is quite
> > expensive. F.e. it is possible that a task has a pid from one container,
> > but it is in process group and/or session of another container,
> > and its controlling terminal owner by another container. Grr..
>
> If that happens, it also means your container is not fully isolated which
> is also a challenge for the vpid approach when you try to migrate. nop ?

Yes, in this case container is not isolated and its migration
is prohibited.

In openvz containers may be not isolated. openvz does not impose
non-overridable access restrictions: f.e. if administrator wants,
he can give access to some raw networking device or to mount some raw
partition inside container fs namespace. Also "vzctl exec" can pass to child
an open file from host (actually, it always passes open pipes to communicate
to the child). Obviously, in all those cases the container cannot migrate.

> If i take your example with the external process group, what would happen
> if the process group leader dies and then you try to migrate that container
> ? How would you restore the processes in your container that are refering a
> dead external process group leader ?

Container can be migrated only when all the references resolve inside
this container. External references are not a normal situation, but
they are not prohibited.

Typical example is:

vzctl exec 202 "sleep 3600 < /dev/null >& /dev/null &"

sleep will be normal process inside container, but its process group
and session are inherited from host system.

```
root@mops:~ # ps axj | fgrep 3600
 4890  6880  6879  6879 ?           -1 S        0   0:00 sleep 3600
root@mops:~ # vzctl exec 202 "ps axj" | fgrep 3600
    1  7904  6879  6879 ?          -1 S        0   0:00 sleep 3600
```

See? pid and ppid look different (virtual ones inside container), but
pgid and sid are not. Apparently, such container cannot migrate

until the sleep exits.

We could force each process visiting container to daemonize and to setsid().
But do not forget that pid space is just a little part of the whole engine,
to force full isolation we have to close all the files opened
in root container, to get rid of its memory space before entering container
etc. But it makes not so much of sense, because in any case we have to keep
at least some way to communicate to host. F.e. even when we allow to pass
only an open pipe, we immediately encounter the situation when a file
owned by one container could refer to processes of another container.

So that, the only way to enforce full isolation is to prohibit
"vzctl exec/enter" as whole.

> We've been living with the vpid approach also for years and we found issues
> that we haven't solve at restart. So we think we might do a better job with
> another. But, this still needs to be confirmed :)

What are the issues?

The only inconvenience which I encountered until now
is a little problem with stray pids. F.e. this happens with flock().
Corresponding kernel structure contains some useless (actually, illegal
and contradicting to the nature of flock()) reference to pid.
If the process took the lock and exited, stray pid remains forever and points
to nowhere. In this case it is silly to prohibit checkpointing,
but we have to restore the flock to a lock with pointing to the same point
in the sky, i.e. to nowhere. With (container, pid) approach we would
restore it pointing to exactly the same empty place in the sky, with
vpids we have to choose a new place. Ugly, but not a real issue.

> i don't see much changes, when you query a task by pid, you only look in
> your *current* container pidspace.

You can query not only task, you can query process group, session,
which is openvz can be not inside the container. You can lookup pid of
file owner, tty session, lock owner etc. Essentially, each place in kernel,
where a pid is stored has to refer to container as well.
Unless, of course, you do not prohibit containers to share
anything and allow them to communicate only via TCP.

> > 2. It is very inconvenient not to see processes inside VPS from host system.
> > To do ps, strace, gdb etc. we have to move inside VPS. With VPID approach I can
> > gdb even "init" process of VPS in a way invisible to VPS, see?
>

> that's another container model issue again. your init process of a VPS
> could be the real init. why do you need a fake one ? just trying to
> understand all the issues you had to solve and I'm sure they are valid.

It is not a fake init, it is a real init. Main goal of openvz is to allow
to start even f.e. the whole instance of stock redhat inside container
including standard init. It is not a strict architectural requirement,
but this option must be present.

BTW the question sounds strange. I would think that in (container, pid)
approach among another limitations you get requirement that you _must_
have a child reaper inside container. With VPIDs this does not matter,
wait() made by parent inside host always returns global pid of child.
But with (container, pid) children can be reaped only inside container, right?

Alexey

---