Subject: Re: [PATCH 1/4] Virtualization/containers: introduction
Posted by serue on Tue, 07 Feb 2006 20:19:08 GMT
View Forum Message <> Reply to Message

Quoting Hubertus Franke (frankeh@watson.ibm.com):
> The vpid approach has the drawbacks of having to identify the conversion
> spots
> of all vpid vs. pid semantics. On the otherhand it does take advantage
> of the fact that no virtualization has to take place until a "container"
> has been migrated, thus rendering most of the vpid<->pid calls to be
> noops.
>
> What I like about the pspace approach is that it explicitely defines in the
> code
> when I am using a different pspace for the lookup. That is kind of hidden in
> the vpid/pid approach.

I agree with this. From a maintenance pov, imagining making a minor change to some pid-related code, if I see something doing effectively "if (pspace1 == pspace2 && pid1==pid2)" that is clear, whereas trying to remember whether I'm supposed to return the pid or vpid can get really confusing. We actually had some errors with that while we were developing the first vpid patchset we posted in december.

I believe that from a vserver point of view, either approach will work. You either create a new pspace and make 'init' pid 1 in that pspace, or, in the openvz approach, you start virtualizing with a hashtable so userspace in the new vserver/container/vz/whateveritscalled sees that init as pid1, while the rest of the system sees it as pid 3270 or something.

Likewise, for checkpoint/restore and migration, either approach works. All we really need is, on restore/migrate, to be able to create processes with their original pids, so we can do that either with real pids in a new container, or virtualized pids faked for process in the same vz.

Are there other uses of pid virtualization which one approach or the other cannot accomodate?

If not, then I for one lean towards the more maintainable code. (which I'm sure we're not agreed on is Eric's, but imvho it is)

-serge