Subject: Re: [RFC] [PATCH 0/7] Some basic vserver infrastructure
Posted by ebiederm on Wed, 22 Mar 2006 07:13:49 GMT
View Forum Message <> Reply to Message

Sam Vilain <sam@vilain.net> writes:

> Dave Hansen wrote:
>
>>>This patch is simple, but does not handle SMP scalability very well
>>>(you'll get a lot of cacheline problems when you start actually using
>>>the container structure; the hashing helps a lot there)
>>>
>>>
>>Could you elaborate a bit on this one? What has cacheline problems?
>>
>>
>
> OK, on reflection this probably doesn't belong this early in the
> series. It only helps speed up lookup by IDs which is only an SMP
> speed enhancement if you expect a lot of those (eg, when storing the
> XIDs on another subsystem, such as a filesystem) and unusual things like
> creating/destroying a lot of vservers quickly. I'll pull it out.
>
> I tried to make it as basic as possible, but my basic problem with the
> patch you linked to is that it shouldn't be so small that you don't get
> a decent bunch of the internal API functions listed in description of
> part 1 of the set
> (http://vserver.utsl.gen.nz/patches/utsl/2.6.16-rc4-vsi/01-VS erver-Umbrella.diff)

Right. I think the smallest thing that we can reasonably discuss with
real problems is the sysvipc namespace. This is why I suggested efforts
in that direction.

>>>and does not provide functions such as looking up a container by ID etc.
>>>
>>>
>>
>>We need something so simple that we probably don't even deal with ids.
>>I believe that Eric claims that we don't really need container _ids_.
>>For instance, the filesystem namespaces have no ids, and work just fine.
>>
>>
>
> For actual namespace structures themselves that are virtualising real
> things, I agree entirely.
>
> But I think the whole point of this patchset (which sadly vger ate for
> the wider audience, and I still don't know why) is to group tasks and to

> give userland, and hence the administrator, something tangible with
> which to group processes with and tack namespace structures onto. I can
> split out the ID related stuff into another patch, but it would need to
> go back in before a useful syscall interface can be added. I'll
> consider it anyway, though.

So as best I can determine that is simply an implementation optimization.
(That is with a little refactoring we can add a structure like that later
 if the performance concerns warrant it.)

Skipping that optimization we should be able to concentrate on the
fundamentals. Which should be simpler and allow better forward progress.

> Note that a given vx_info structure still might share namespace
> structures with other vx_info objects - this is what I was alluding to
> with the "we haven't made any restrictions on the nature of
> virtualisation yet" comment. All we've made is the (XID, PID) tuple
> (although the default for XID=0 is that all processes are visible in one
> big PID space). The intention is that flags will control how you want
> your PIDs to work for that vserver.

Do we need flags or can we move this to user space?

> The only thing that they can't do is share processes - it is a one to
> many relationship. However, if there can be a parent/child
> containership relationship on the XIDs themselves, you can still achieve
> the behaviour of these unusual situations. I'm working on the
> assumption that if we ever have to migrate trees of XIDs then we can
> virtualise how they look inside a vserver using flags.
>
> Eric, perhaps you can comment or refer to the earlier post where you
> made this argument. I tried to follow it but perhaps I missed the jist
> of one of the messages or one of the most important messages entirely.

I'm not certain which argument you are referring to but I will state my general
argument against adding additional global ids.

Ultimately you want to nest you vservers and the like inside of each
other, because if you haven't captured everything you can do in user
space there will be some applications that don't work, and ultimately
it is desirable to support all applications. When doing that you want
to use the same mechanism on the inside as you have on the outside.
Additional global identifiers make this very difficult.

You can always talk about resources by specifying the processes that
use them. It is one level of indirection, but it works and is simple.

>>By starting painfully simply, we can build on complexity in bits, and
>>justify it as we go.
>>
> This is my aim too! I'll keep chopping and changing.
>
>>>And also because the acronym "vx" makes the API look nice, at least to
>>>mine and Herbert's eyes, then when you go to the network virtualisation
>>>you get "nx_info", etc. However I'm thinking any of these terms might
>>>also be right:
>>>
>>> - "vserver" spelt in full
>>> - family
>>> - container
>>> - jail
>>> - task_ns (sort for namespace)
>>>
>>>Perhaps we can get a ruling from core team on this one, as it's
>>>aesthetics :-).
>>>
>>>
>>
>>I was in a meeting with a few coworkers, and we were arguing a bit about
>>naming. One person there was a manager-type who didn't have any direct
>>involvement in the project. We asked him which naming was more clear.
>>
>>We need to think a bit like that. What is more clear to somebody who
>>has never read the code? (Hint "vx_" means nothing. :)
>>
>>
>
> OK, so let's look at all of the various names that stem from the use of
> the term "XID" and try to come up with a good naming system for each. I
> invite the OpenVZ team, Eric and anyone else to put forward their names
> as well.
>
> Sorry if this is a bit long. Again I invite anyone at all to come
> forward with a preference or list another set of alternatives. Let's
> nail this one down, it comes up every time any patchset like this is put
> forward.

> For the record, I like the term "process family" most. It implies the
> possibility strict grouping, like last name, as well as allowing
> heirarchies, but of course in our modern, post-nuclear family age, does
> not imply a fixed nature of anything ;-).

Families don't sound bad, but this all feels like putting the cart
before the horse. It is infrastructure solving a problem that I am

not at all certain is interesting.

Eric