Subject: Re:  Re: [RFC] [PATCH 0/7] Some basic vserver infrastructure
Posted by dev on Fri, 24 Mar 2006 15:36:52 GMT
View Forum Message <> Reply to Message

>> 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.
ok. I send to all on CC our IPC and utsname namespace patch.
We implemented them as separate namespaces (to please Eric), compile
time configurable (to please embedded people) and using "current"
namespace context (as we do in OpenVZ).
I suppose all these points can be disucssed. In many respects the
patches look like a bit changed Eric/DHansen/OpenVZ.
I suppose this should be discussed/worked out and commited to
Linus/Andrew as there are any serious issues here.

>>> 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.
Eric, the namespaces itself can be without any IDs, really.
But on top of that projects like OpenVZ/vserver can implement VPSs,
umbreallas etc. and will introduce IDs and interfaces they are
accustomed of.

> 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.
I think it is not that easy as you wish to think of it.
Consider 2 processes: A and B.
each of them has its own fs namespace, but common network namespace.
process A passes file descriptor fd to process B via unix socket.
So the same file belongs to 2 different fs namespaces.
Will you simply forbid fd passing to another fd namespace?

I see 2 possibilities here.

1. You either introduce fully isolated resource namespaces, than it is flat, not nested.
2. Or you need to state correctly what "nesting" means from your point of view and which operations parent namespace can do with its children.
Last time I had a talk with Herbert we ended up that such a nesting is nothing more than a privileges delegation, e.g. allows you to manage your children.
Eric, can your describe what you want from nesting on some particular namespace, for example fs, ipc, networking, ...? Not by word "nesting", but by concrete operations which parent can do and why a child looks "nested".

Just to make it more clear: my understanding of word "nested" means that if you have, for example, a nested IPC namespace, than parent can see all the resources (sems, shms, ...) of it's children and have some private, while children see only its own set of private resources. But it doesn't look like you are going to implement anything like this.
So what is nesting then? Ability to create namespace? To delegate it some part of own resource limits?

>>>> 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 propose to use "namespace" naming.
1. This is already used in fs.
2. This is what IMHO suites at least OpenVZ/Eric
3. it has good acronym "ns".

>> 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.
agreed.

Thanks,
Kirill