

---

Subject: Re: [PATCH 0/2] resource control file system - aka containers on top of nsproxy!

Posted by [Herbert Poetzl](#) on Sat, 10 Mar 2007 02:03:48 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On Sat, Mar 10, 2007 at 12:11:05AM +0530, Srivatsa Vaddagiri wrote:

> On Fri, Mar 09, 2007 at 02:16:08AM +0100, Herbert Poetzl wrote:

> > On Thu, Mar 08, 2007 at 05:00:54PM +0530, Srivatsa Vaddagiri wrote:

> > > On Thu, Mar 08, 2007 at 01:50:01PM +1300, Sam Vilain wrote:

> > > > 7. resource namespaces

> > >

> > > It should be. Imagine giving 20% bandwidth to a user X. X wants to

> > > divide this bandwidth further between multi-media (10%), kernel

> > > compilation (5%) and rest (5%). So,

> >

> > sounds quite nice, but ...

> >

> > > > Is the subservient namespace's resource usage counting against

> > > > ours too?

> > >

> > > Yes, the resource usage of children should be accounted when capping

> > > parent resource usage.

> >

> > it will require to do accounting many times

> > (and limit checks of course), which in itself

> > might be a way to DoS the kernel by creating

> > more and more resource groups

>

> I was only pointing out the usefulness of the feature and not

> necessarily saying it -should- be implemented! Ofcourse I understand it

> will make the controller complicated and thats why probably none of the

> recontrollers we are seeing posted on lkml don't support hierarchical

> res mgmt.

>

> > > > Can we dynamically alter the subservient namespace's resource

> > > > allocations?

> > >

> > > Should be possible yes. That lets user X completely manage his

> > > allocation among whatever sub-groups he creates.

> >

> > what happens if the parent changes, how is

> > the resource change (if it was a reduction)

> > propagated to the children?

> >

> > e.g. your guest has 1024 file handles, now

> > you reduce it to 512, but the guest had two

> > children, both with 256 file handles each ...

>

> I believe CKRM handled this quite neatly (by defining child shares to be  
> relative to parent shares).  
>  
> In your example, 256+256 add up to 512 which is within the parent's  
> new limit, so nothing happens :)

yes, but that might as well be fatal, because now the  
children can easily DoS the parent by using up all the  
file handles, where the 'original' setup (2 x 256)  
left 512 file handles 'reserved' ...

of course, you could as well have adjusted that to  
2 x 128 + 256 for the parent, but that is policy and  
IMHO policy does not belong into the kernel, it should  
be handled by userspace (maybe invoked by the kernel  
in some kind of helper functionality or so)

> You also picked an example of exhaustible/non-reclaimable resource,  
> which makes it hard to define what should happen if parent's limit  
> goes below 512.

which was quite intentional, and brings us to another  
issues when adjusting resource limits (not even in  
a hierarchical way)

> Either nothing happens or perhaps a task is killed, don't know.

> In case of memory, I would say that some of child's pages may  
> get kicked out and in case of cpu, child will start getting fewer  
> cycles.

btw, kicking out pages when rss limit is reached might  
be the obvious choice (if we think Virtual Machine here)  
but it might not be the best choice from the overall  
performance PoV, which might be much better off by  
keeping the page in memory (if there is enough memory  
available) but penalizing the guest like the page was  
actually kicked out (and needs to be fetched later on)

note: this is something we should think about when we  
want to address specific limits like RSS, because IMHO  
we should not optimize for the single guest case, but  
for the big picture ...

> > > The patches should give visibility to both nsproxy objects (by  
> > > showing what tasks share the same nsproxy objects and letting  
> > > tasks move across nsproxy objects if allowed) and the resource  
> > > control objects pointed to by nsproxy (struct cpuset, struct

> > > cpu\_limit, struct rss\_limit etc).  
> >  
> > the nsproxy is not really relevant, as it  
> > is some kind of strange indirection, which  
> > does not necessarily depict the real relations,  
> > regardless whether you do the re-sharing of  
> > those namespaces or not ..  
>  
> So what are you recommending we do instead?  
> My thought was whatever is the fundamental unit to which resource  
> management needs to be applied, let's store resource parameters (or  
> pointers to them) there (rather than duplicating the information in  
> each task\_struct).

we do not want to duplicate any information in the task  
struct, but we might want to put some (or maybe all)  
of the spaces back (as pointer reference) to the task  
struct, just to avoid the nsproxy indirection

note that IMHO not all spaces make sense to be separated  
e.g. while it is quite useful to have network and pid  
space separated, others might be joined to form larger  
consistent structures ...

for example, I could as well live with pid and resource  
accounting/limits sharing one common struct/space ...  
(doesn't mean that separate spaces are not nice :)

best,  
Herbert

> --  
> Regards,  
> vatsa  
> \_\_\_\_\_  
> Containers mailing list  
> Containers@lists.osdl.org  
> <https://lists.osdl.org/mailman/listinfo/containers>

Containers mailing list  
Containers@lists.osdl.org  
<https://lists.osdl.org/mailman/listinfo/containers>

---