Subject: Re: Summary of resource management discussion Posted by Herbert Poetzl on Fri, 16 Mar 2007 14:19:16 GMT View Forum Message <> Reply to Message

On Thu, Mar 15, 2007 at 10:34:35PM +0530, Srivatsa Vaddagiri wrote: > On Thu, Mar 15, 2007 at 04:24:37AM -0700, Paul Menage wrote: > If there really was a grouping that was always guaranteed to match the > way you wanted to group tasks for e.g. resource control, then yes, it > would be great to use it. But I don't see an obvious candidate. The > pid namespace is not it, IMO. > In vserver context, what is the "normal" case then? Atleast for Linux > Vserver pid namespace seems to be normal unit of resource control (as per > Herbert).

it is, but mainly because a 'context' in the Linux-VServer case is a struct, which defines all the properties for a 'guest' excluding separate (name)spaces. as the pid space doesn't exist yet, it is part of the task grouping context

> Even if one wanted to manage a arbitrary group of tasks in vserver

> context, IMHO its still possible to construct that arbitrary group using

> the existing pointer, ns[/task]proxy, and not break existing namespace
> semantics/functionality.
>
> So the normal case I see is:

> pid\_ns1 uts\_ns1 cpu\_ctl\_space1 pid\_ns2 uts\_ns2 cpu\_ctl\_space2 > ٨ Λ (50%) Λ ۸ (50%)> ٨ ٨ > > > task\_proxy1 task\_proxy2 > (Vserver2) (Vserver1) > > > > > But, if someone wanted to manage cpu resource differently, and say that > postgres tasks from both vservers should be in same cpu resource class, > the above becomes: > > pid\_ns1 uts\_ns1 cpu\_ctl\_space2 > pid\_ns1 uts\_ns1 cpu\_ctl\_space1 > ۸ ^ (25%) Λ ۸ (50%) Λ Λ > > > task proxy1 > I task\_proxy2 

```
(Vserver1) | (postgres tasks in VServer1) |
   >
                           -----
>
>
>
                                     pid_ns2 uts_ns2 cpu_ctl_space2
    pid_ns2 uts_ns2 cpu_ctl_space3
>
>
     Λ
          ^ (25%) ^ ^
                                     (50%)
                Λ
                                Т
                                     Λ
>
                >
                                >
   task_proxy3|task_proxy4(Vserver2)|(postgres tasks in VServer2 |
>
>
                           _____
>
>
> (the best I could draw using ASCII art!)
>
> The benefit I see of this approach is it will avoid introduction of
> additional pointers in struct task struct and also additional structures
> (struct container etc) in the kernel, but we will still be able to retain
> same user interfaces you had in your patches.
>
> Do you see any drawbacks of doing like this? What will break if we do
```

> this?

looks good to me, except for the potential issue with the double indirection introducing too much overhear (compared to something like this:

```
ipc uts

^ ^ _

+---+--+--+

| nsproxy |

+----+--+ cpu pid

^ ^ ^ ^

| | |

+---+---+-+

| task |

+---+--++
```

don't forget, accounting for cpu is probably very closely tied to tasks, while this doesn't matter much for other resources like number of tasks or file handles ...

> > Resource control (and other kinds of task grouping behaviour) shouldn't

> > require virtualization.

>

> Certainly. AFAICS, nsproxy[.c] is unconditionally available in the

> kernel (even if virtualization support is not enabled). When reused for

> pure resource control purpose, I see that as a special case of virtualization > where only resources are virtualized and namespaces are not. > > I think an interesting question would be : what more task-grouping > behavior do you want to implement using an additional pointer that you > can't reusing ->task\_proxy? > > > >a. Paul Menage's patches: >>> (tsk->containers->container[cpu ctlr.subsys id] - X)->cpu limit >>> > > > > So what's the '-X' that you're referring to > > Oh ...that's to seek pointer to begining of the cpulimit structure (subsys > pointer in 'struct container' points to a structure embedded in a larger > structure. -X gets you to point to the larger structure). > > > >6. As tasks move around namespaces/resource-classes, their >>> tsk->nsproxy/containers object will change. Do we simple create >>> a new nsproxy/containers object or optimize storage by searching >>> for one which matches the task's new requirements? > > > > I think the latter. > > Yes me too. But maybe to keep in simple in initial versions, we should > avoid that optimisation and at the same time get statistics on duplicates?. I agree here, just let us keep some way to actually check how much overhead we add with nsproxy, etc 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