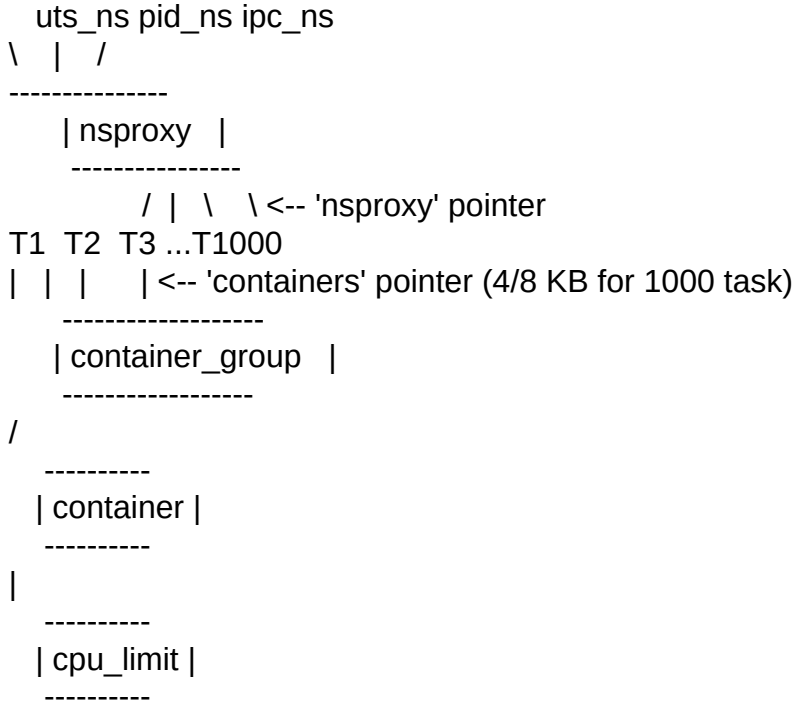
Subject: Re: [PATCH 0/2] resource control file system - aka containers on top of nsproxy!
Posted by Srivatsa Vaddagiri on Sat, 10 Mar 2007 02:02:20 GMT
View Forum Message <> Reply to Message

I think maybe I didnt communicate what I mean by a container here
(although I thought I did). I am referring to a container in a vserver
context (set of tasks which share the same namespace).

On Fri, Mar 09, 2007 at 02:09:35PM -0800, Paul Menage wrote:
> >2. Regarding space savings, if 100 tasks are in a container (I dont know
> > what is a typical number) -and- lets say that all tasks are to share
> > the same resource allocation (which seems to be natural), then having
> > a 'struct container_group *' pointer in each task_struct seems to be not
> > very efficient (simply because we dont need that task-level granularity
> > of
> > managing resource allocation).
>
> I think you should re-read my patches.
>
> Previously, each task had N pointers, one for its container in each
> potential hierarchy. The container_group concept means that each task
> has 1 pointer, to a set of container pointers (one per hierarchy)
> shared by all tasks that have exactly the same set of containers (in
> the various different hierarchies).

Ok, let me see if I can convey what I had in mind better:



(T1, T2, T3 ..T1000) are part of a vserver lets say sharing the same uts/pid/ipc_ns. Now where do we store the resource control information for this unit/set-of-tasks in your patches?

(tsk->containers->container[cpu_ctlr.hierarchy] + X)->cpu_limit

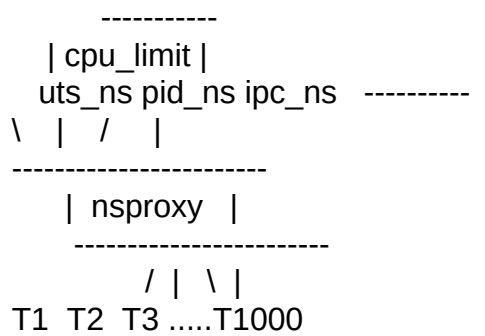
(The X is to account for the fact that cotainer structure points to a 'struct container_subsys_state' embedded in some other structure. Its usually zero if the structure is embedded at the top)

I understand that container_group also points directly to 'struct container_subsys_state', in which case, the above is optimized to:

(tsk->containers->subsys[cpu_ctlr.subsys_id] + X)->cpu_limit

Did I get that correct?

Compare that to:



We save on 4/8 KB (for 1000 tasks) by avoiding the 'containers' pointer in each task_struct (just to get to the resource limit information).

So my observation was (again note primarily from a vserver context): given that (T1, T2, T3 ..T1000) will all need to be managed as a unit (because they are all sharing the same nsproxy pointer), then having the '->containers' pointer in -each- one of them to tell the unit's limit is not optimal. Instead store the limit in the proper unit structure (in this case nsproxy - but whatever else is more suitable vserver datastructure (pid_ns?) which represent the fundamental unit of res mgmt in vservers).

(I will respond to remaining comments later ..too early in the morning now!)

--
Regards,
vatsa

_____
Containers mailing list