Subject: Re: [ckrm-tech] [PATCH 0/2] resource control file system - aka containers on top of nsproxy!
Posted by Srivatsa Vaddagiri on Mon, 05 Mar 2007 17:34:01 GMT
View Forum Message <> Reply to Message

On Sat, Mar 03, 2007 at 06:32:44PM +0100, Herbert Poetzl wrote:
> > Yes, perhaps this overloads nsproxy more than what it was intended for.
> > But, then if we have to to support resource management of each
> > container/vserver (or whatever group is represented by nsproxy),
> > then nsproxy seems the best place to store this resource control
> > information for a container.
>
> well, the thing is, as nsproxy is working now, you
> will get a new one (with a changed subset of entries)
> every time a task does a clone() with one of the
> space flags set, which means, that you will end up
> with quite a lot of them, but resource limits have
> to address a group of them, not a single nsproxy
> (or act in a deeply hierarchical way which is not
> there atm, and probably will never be, as it simply
> adds too much overhead)

Thats why nsproxy has pointers to resource control objects, rather than
embedding resource control information in nsproxy itself.

>From the patches:

struct nsproxy {

+#ifdef CONFIG_RCFS
+ struct list_head list;
+ void *ctlr_data[CONFIG_MAX_RC_SUBSYS];
+#endif

}

This will let different nsproxy structures share the same resource
control objects (ctlr_data) and thus be governed by the same parameters.

Where else do you think the resource control information for a container
should be stored?

> > It should have the same perf overhead as the original
> > container patches (basically a double dereference -
> > task->containers/nsproxy->cpuset - required to get to the
> > cpuset from a task).
>
> on every limit accounting or check? I think that

> is quite a lot of overhead ...

tsk->nsproxy->ctlr_data[cpu_ctlr->id]->limit (4 dereferences) is what we need to get to the cpu b/w limit for a task.

If cpu_ctlr->id is compile time decided, then that would reduce it to 3.

But I think if CPU scheduler schedules tasks from same container one after another (to the extent possible that is), then other derefences (->ctlr_data[] and ->limit) should be fast, as they should be in the cache?

--
Regards,
vatsa

_____

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