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

Posted by Herbert Poetzl on Mon, 05 Mar 2007 18:39:37 GMT

View Forum Message <> Reply to Message

```
On Mon, Mar 05, 2007 at 11:04:01PM +0530, Srivatsa Vaddagiri wrote:
> 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.
which makes it a (name)space, no?
> >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.
as it is currently done for vfs, uts, ipc and soon
pid and network I2/I3, yes?
> Where else do you think the resource control information for a
> container should be stored?
```

indirection, as we had for vfs before) >>> 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. sounds very 'cache intensive' to me ... (especially compared to the one indirection be use atm) > 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), which is very probably not what you want, as it - will definitely hurt interactivity - give strange 'jerky' behaviour - ignore established priorities > then other derefences (->ctlr_data[] and ->limit) should be fast, as > they should be in the cache? please provide real world numbers from testing ... at least for me, that is not really obvious in four way indirection:) TIA. Herbert > Regards, > vatsa Containers mailing list Containers@lists.osdl.org

an alternative for that is to keep the resource stuff as part of a 'context' structure, and keep a reference from the task to that (one less Page 3 of 3 ---- Generated from OpenVZ Forum