Subject: Re: [ckrm-tech] [PATCH 7/7] containers (V7): Container interface to nsproxy subsystem
Posted by Srivatsa Vaddagiri on Thu, 05 Apr 2007 08:49:20 GMT
View Forum Message <> Reply to Message

On Wed, Apr 04, 2007 at 11:48:57PM -0700, Paul Menage wrote:
> >rcfs_task_count will essentially return number of tasks pointing to A1
> >thr' their nsproxy->ctlr_data[BC_ID].
>
> One small issue with the (last posted) version of your patch is that
> it doesn't take into account the refcounts from the directories
> themselves

You mean dentry->d_fsdata pointing to nsproxy should take a ref count on nsproxy? afaics it is not needed as long as you first drop the dentry before freeing associated nsproxy.

> - I think you probably need to subtract one for each active
> subsystem.

I don't understand this.

> I don't think that's a reasonable assumption. A task can have
> thousands of file handles open - having to scan and move every file
> that the task has open would make a move operation incredibly
> expensive.
>
> Additionally, tasks can share many of those file handles
> with other tasks. So what happens if one task that has a file open
> moves out of the container, but another stays behind? It's cleaner and
> more efficient, and conceptually desirable, IMO, just to keep the file
> associated with the container.

I don't have a authoritative view here on whether open file count should be migrated or not, but from a layman perspective consider this:

- Task T1 is in Container C1, whose max open files can be 100
- T1 opens all of those 100 files
- T1 migrates to Container C2, but its open file count is not migrated
- T2 is migrated to container C1 and tries opening a file but is denied. T2 looks for "who is in my container who has opened all files" and doesn't find anyone.

Isn't that a bit abnormal from an end-user pov?

> >Why refcount 3? I can only be 1 (from T) ..
>

> Plus the refcounts from the two filesystem roots.

Filesystem root dentry's are special case. They will point to
init_nsproxy which is never deleted and hence they need not add
additional ref counts.

For other directories created, say H1/foo, foo's dentry will point to N1
but need not take additional refcount. N1 won't be deleted w/o dropping
foo's dentry first. I think this is very similar to cpuset case, where
dentry->d_fsdata = cs doesnt take additional ref counts on cpuset.

> >The object was created by the task, so I would expect it should get
> >migrated too to the new task's context (which should be true in case of
> >f_bc atleast?). Can you give a practical example where you want to
> >migrate the task and not the object it created?
>
> I gave one above, for files; others could include pages (do you want
> to have to migrate every page when a task switches container? what
> about shared pages?)
>
> Obviously this fundamental difference of opinion means that we're
> going to end up disagreeing on whether the scenario I presented is a
> problem or not ...

Again I am not a VM expert to say whether pages should get migrated or
not. But coming to the impact of this discussion on xxx_rmdir() ..

> The problem with that is that (given the assumption that some
> subsystems might not want to migrate objects) you can then end up with
> a subsystem state object that has refcounts on it from active objects
> like files, but which is unreachable via any container filesystem
> mechanism. Better IMO to be able to fail the rmdir() in that situation
> so that the subsystem object remains accessible (so you can see where
> the resources are being used up).

I agree we shouldn't delete a dir going by just the task count. How abt
a (optional) ->can_destroy callback which will return -EBUSY if additional
non-task objects are pointing to a subsyste's resource object?

--
Regards,
vatsa