Subject: Re: [PATCH 3/7] Containers (V8): Add generic multi-subsystem API to containers
Posted by Srivatsa Vaddagiri on Wed, 11 Apr 2007 16:35:35 GMT
View Forum Message <> Reply to Message

Coming from a "simplify things" pov:

On Fri, Apr 06, 2007 at 04:32:24PM -0700, menage@google.com wrote:
> struct container {
> unsigned long flags; /* "unsigned long" so bitops work */
>
> /*
> - * Count is atomic so can incr (fork) or decr (exit) without a lock.
> - */
> - atomic_t count; /* count tasks using this container */
> -
> - /*
> * We link our 'sibling' struct into our parent's 'children'.
> * Our children link their 'sibling' into our 'children'.
> */
> @@ -43,11 +106,13 @@ struct container {
> struct list_head children; /* my children */
>
> struct container *parent; /* my parent */
> - struct dentry *dentry; /* container fs entry */
> + struct dentry *dentry; /* container fs entry */
>
> -#ifdef CONFIG_CPUSETS
> - struct cpuset *cpuset;
> -#endif
> + /* Private pointers for each registered subsystem */
> + struct container_subsys_state *subsys[CONTAINER_SUBSYS_COUNT];
> +
> + struct containerfs_root *root;

Could this root pointer derived from dentry pointer
(cont->dentry->d_sb->s_fs_info)?

> + struct container *top_container;

and this as well?

cont->dentry->d_sb->s_fs_info->top_container

> };

So we have the foward subsys pointer array being stored in both
'struct container' and 'struct container_group' and reverse container pointer

stored in struct container_subsys_state. Can we reduce this pointer maze by:

```
struct container {
 /* All shared stuff like flags, parent/child pointers etc */
 ..

 struct container_group *my_group;

}
```

The forward mapping from 'struct container' to subsys objects is made via 'my_group'. This also lets 'struct container' be a placeholder strictly for shared state.

On further thoughts, perhaps even my_group can be avoided by having:

```
dentry->d_fsdata point to my_group
```

and cont->dentry->d_fsdata will point to my_group which we wanted to store above.

I don't see distinct adv of doing this, but I suspect it will simplify the structure relationship (and code) a bit.

--
Regards,
vatsa