Subject: Thoughts on virtualizing task containers
Posted by Paul Menage on Fri, 24 Aug 2007 00:18:15 GMT
View Forum Message <> Reply to Message

I'm interested in people's thoughts about the right user API for virtualizing task containers and resource controllers.

With namespaces, the model is fairly straightforward - you can split a new namespace off from your own, without caring whether you're the top-level namespace or some child namespace. So nested namespaces just work, but at the expense of not being able to see anything that's at a higher level than your current namespace.

For resource controllers and other task container subsystems, that model isn't necessarily desired. For example, tasks running in a cpuset don't see their own cpuset as the root; it's possible (with appropriate permissions) to modify other cpusets outside of your own one.

For doing full virtual server containers, root in the virtual server may well want to make use of task container subsystems, e.g. to control the amount of CPU cycles or memory that groups of processes within the virtual server can use. What kinds of controls do we want to give the host to determine what kind of container operations the guest virtual server can do? What should the guest see in terms of task container support?

Ideally, the guest would have what appears to it to be a full (and isolated) task container implementation to play with, complete with a full complement of subsystems.

But at the same time, some subsystems might not be easily virtualizable, and the host might not want the guest to have access to all subsystems anyway.

One way to handle this might be to have a "virtualize" subsystem that allows you to set virtualization boundaries; by setting a particular container's virtualize control file to "true" you'd enforce the rule that tasks in that container (or child containers) would:

- only be able to mount container hierarchies with task subsystems mounted in that hierarchy

- see that container as their root container

- not be able to increase the resource limits of their own container. (similar to the way that the "mems" and "cpus" files in the root cpuset container directory are read-only, certain control files would

become read-only in the virtualized container directory).

Possibly rather than needing a separate "virtualize" subsystem you could infer virtualization boundaries based on container boundaries in whichever hierarchy had the nsproxy subsystem mounted (if any). If the nsproxy subsystem wasn't mounted anywhere, then no virtualization would occur.

On top of the implementation issues, there are conceptual issues to deal with such as:

- what do you do with subsystems (e.g. freezer, OOM handler, network flow id assigner, etc) that are inherently hard to virtualize in a hierarchical way?

- if the host mounts a container filesystem hierarchy in the guest's filespace, does the guest see the host's view of containers or the guest's view? i.e. is the virtualization associated with the mount point or with the process doing the viewing? (I'm inclined to say the former)

- how much visibility/control does the host have into any child task containers created by the guest?

- can the guest split the subsystems that are visible to it amongst multiple hierarchies? Or do we rule that guests can only have access to a single hierarchy?

Paul

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers