Subject: Re: [RFC] Control Groups Roadmap ideas
Posted by serue on Mon, 14 Apr 2008 14:11:19 GMT
View Forum Message <> Reply to Message

Quoting Paul Menage (menage@google.com):
> On Fri, Apr 11, 2008 at 7:48 AM, Serge E. Hallyn <serue@us.ibm.com> wrote:
> > > 2) More flexible binding/unbinding/rebinding
> > > -----
> > >
> > > Currently you can only add/remove subsystems to a hierarchy when it
> > > has just a single (root) cgroup. This is a bit inflexible, so I'm
> > > planning to support:
> > >
> > > - adding a subsystem to an existing hierarchy by automatically
> > > creating a subsys state object for the new subsystem for each existing
> > > cgroup in the hierarchy and doing the appropriate
> > > can_attach()/attach_tasks() callbacks for all tasks in the system
> > >
> > > - removing a subsystem from an existing hierarchy by moving all tasks
> > > to that subsystem's root cgroup and destroying the child subsystem
> > > state objects
> > >
> > > - merging two existing hierarchies that have identical cgroup trees
> > >
> > > - (maybe) splitting one hierarchy into two separate hierarchies
> > >
> > > Whether all these operations should be forced through the mount()
> > > system call, or whether they should be done via operations on cgroup
> > > control files, is something I've not figured out yet.
> >
> > I'm tempted to ask what the use case is for this (I assume you have one,
> > you don't generally introduce features for no good reason), but it
>
> Back during the early versions of control groups, Paul Jackson
> proposed a bind/unbind API that would let you affect the subsystems on
> an active hierarchy, and it was always a goal of mine to implement
> that - current inflexibility is something that I've never been that
> keen on, but it was OK for the first big release and could be extended
> later.
>
> One of the potential scenarios was that you might want to have a very
> early boot script set up cpusets and node isolation for a set of
> system daemons, and then bind other subsystems on to the same
> hierarchy later in the boot process.
>
> > I'd stick with mount semantics. Just
> >   mount -t cgroup -o remount,devices,cpu none /devwh"
> > should handle all cases, no?

>
> Yes, probably - particularly if we restrict it to adding/removing
> subsystems from an existing tree, rather than splitting and merging
> multiple hierarchies.
>
> >
> > I guess I'm hoping that if libcg goes well then a userspace daemon can
> > do all we need. Of course the use case I envision is having a container
> > which is locked to some amount of ram, wherein the container admin wants
> > to lock some daemon to a subset of that ram. If the host admin lets the
> > container admin edit a config file (or talk to a daemon through some
> > sock designated for the container) that will only create a child of the
> > container's cgroup, that's probably great.
>
> That's a different issue, and one that I left out of the roadmap
> email. We can have a virtualization subsystem that controls what
> subset of a given hierarchy you can see - if the virtualization
> subsystem is bound to a given hierarchy, and a cgroup is marked as
> virtualized, then a mount of that hierarchy by a process in the
> virtualized cgroup will see that cgroup as the root of the hierarchy.
> It would be a bit like doing a bind mount of a subtree of the main
> hierarchy, but automatically enforced by the kernel.

That seems to work. Now we don't necessarily want that for every group
composed with the virtualized subsystem right? I.e. if I do

mount -o cgroup -t ns,cpuset,virt none /containers

then all tasks are mapped under /containers. If login does a
clone(CLONE_NEWNS) for hallyn's login to give him a private /tmp,
then hallyn ends up under /containers/node_xyz, but we don't want him
to be virtualized under there. So I assume we'd want a virt.lock file
or something like that so, that when I create a container, my
start_container script can echo 1 > /containers/node_abc/virt.lock

I assume the container will also have to remount a fresh copy of the
cgroup composition so it can have the dentry for /containers/node_abc
as the root dentry for /containers?

Anyway that sounds like it address the problem very well.

> > > 8) per-mm owner field
> > > ----
> > >
> > > To remove the need for per-subsystem counted references from the mm.
> > > Being developed by Balbir Singh
> >
> > I'm slooowly trying to whip together a swapfile namespace - not a

> > cgroup - which ties a swapfns to a list of swapfiles (where each
> > swapfile belongs to only one swapfns).
>
> This would be to allow virtual servers to mount their own swapfiles?
> Presumably there'd still be a use for a swap cgroup for job systems
> that want to isolate swap usage without virtualization or requiring
> jobs to mount their own swapfiles?

Yes. Main reason for having this would be so that a container which you're going to migrate could have its own swapfile which can move with it (or live on network fs).

-serge

_____

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