Subject: Re: [PATCH 0/7] containers (V7): Generic Process Containers
Posted by Paul Menage on Tue, 20 Feb 2007 22:47:36 GMT
View Forum Message <> Reply to Message

On 2/20/07, Eric W. Biederman <ebiederm@xmission.com> wrote:
> > Sam said "the NSProxy *is* the container". You appear to be planning
> > to have some namespaces, possibly not aggregated within the nsproxy
> > (pid namespace?) but are you planning to have some higher-level
> > "container" object that aggregates the nsproxy and the other
> > namespaces?
>
> No. A reverse mapping is not needed and is not interesting.

... to you.

> As long as I can walk all processes and ask what namespace are
> you in I don't care.

How do you currently do that?

>
> To me at least the interesting part of your work is not the aggregation
> portion. But the infrastructure for building the different process
> groups.

In that case you can easily use it to just assign one namespace type
to each tree of process groups. The aggregation is something that
other groups find useful, but isn't required for the user to actually
make use of.

>
> You seem to be calling your groups of processes lumped together for
> one purpose or another a container.

Correct.

>
> We need a term for the non-aggregated case for the individual process
> groups we build this out of in your infrastructure. Because you
> clearly have more than one kind of process group a set of processes
> can belong to.

Yes. That's why my system supports multiple unrelated hierarchies of groups.

> > I agree that namespaces fit slightly less well into this model than
> > some other subsystems like resource management. But by integrating
> > with it you'd get automatic access to all the various different
> > resource controller work that's being done.

>
> I don't need to integrate with it to get access to the resource
> controller work.

Right, you could certainly do the extra work of tying your virtual
servers together with resource controllers in userspace. But you'd
still need an API to allow those resource controllers to be associated
with groups of processes and configured with limits/guarantees, which
is one of the main aims of my containers patch.

> Now I have some half backed ideas that might be useful for providing
> a better abstraction. But it requires setting down and looking
> at the problem in detail, and understanding what people are trying
> to accomplish with these things they are building. What subset of
> process groups do you find interesting.

I'm primarily interested in getting something in the kernel that can
be used as a base for interesting subsystems that apply behavioural or
QoS changes to defined groups of processes. Resource controllers and
namespaces seem to be good examples of this, but I can think of useful
subsystems for monitoring, permission control, etc.

>
> All that is necessary to have a group of processes do something
> in an unnamed fashion is to hang a pointer off of the task_struct.
> That's easy.

Right, adding a pointer to task_struct is easy. Configuring how/when
to not directly inherit it from the parent, or to change it for a
running task, or configuring state associated with the thing that the
pointer is pointing to, naming that group, and determining which group
a given process is assocaited with, is something that's effectively
repeated boiler plate for each different subsystem, and which can be
accomplished more generically via an abstraction like my containers
patch.

> You are adding a lot more to that, and there is

The main thing that I'm adding on top of the operations mentioned in
the previous paragraph, which pretty much essentially have to be in
the kernel, is the ability to group multiple different subsystems
together so that they share the same process->container mappings. Yes,
that is something that could potentially be done from userspace
instead, and just provide an independent tree for each subsystem or
namespace. But there seems to be interest from other parties
(including me) in having kernel support for it.

Paul