Subject: Re: [RFC] libcg: design and plans
Posted by Paul Menage on Wed, 05 Mar 2008 18:55:02 GMT

On Wed, Mar 5, 2008 at 6:24 AM, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
> Paul Menage wrote:
> > On Wed, Mar 5, 2008 at 3:07 AM, Dhaval Giani <dhaval@linux.vnet.ibm.com> wrote:
> >> OK. Hmm, I've not really thought about it. At first thought, it should
> >> not be very difficult. Only thing I am not sure is the arbitrary
> >> grouping of the groups (ok, a bit confusing).
> >
> > I suspect that the main form of composite grouping is going to be
> > between parents and children. E.g. you might want to say things like:
> >
> > create_group(A, memory=1G, cpu=100)
> > create_group(B, parent=A, memory=inherit, cpu=20)
> > create_group(C, parent=A, memory=inherit, cpu=30)
> >
> > i.e. both B and C inherit/share their memory limit from their parent,
> > but have their own CPU groups (child groups of their parent?)
> >
>
> No, we don't plan on doing that. What we plan on doing is
>
> 1. Specify the mount point for each controller
> 2. In the create group API, specify the name of the group and the various
> parameters.
>
> If for example CPU is mounted at /cpu and Memory at /mem
>
> Then a specification for creation of group A would be of the form
>
> create_group(A, cpu=100, memory=100M)
>
> Then,
>
> /cpu/A has shares set to 100 and /mem/A has memory.limit set to 100M
>
> If you want to create subgroups under A, you specify
>
> create_group(A/B, memory=200M, cpu=50)
>
> That would create /cpu/A/B and /mem/A/B
>
> Please note that memory and CPU hierarchy needs work in the kernel. The shares
> and hierarchy support is pending. We need to make the res_counters
> infrastructure aware of hierarchies.
>

I think there are two different kinds of sharing going on here:

- A and B each have individual limits, and you additionally want their total usage to be capped by some parent limit. E.g. A and B each have a 100MB memory limit, and you want their total combined usage to not exceed 150MB. This kind of sharing has to be handled by the resource counter abstraction

- you want A and B to be treated identically for the purposes of some particular resource, e.g. you want a single CFS group to which all threads in A and B are equal members, or a single memory cgroup for all allocations by A or B, or the same device control table for A and B (but you want A and B to be treated separately for some other resource type). This can be handled in userspace in the way I outlined above, and it would be good if libcg could handle the setup required for this. It could also be done in the kernel with something like the parent/child subsystem group inheritance that I also mentioned above, if there was demand. But if so it should be a property of cgroups rather than any individual resource controller, since it's a feature that could be useful for all cgroups.

Paul

_____