Subject: Re: [ckrm-tech] [RFC][PATCH] UBC: user resource beancounters
Posted by Rohit Seth on Thu, 24 Aug 2006 17:27:41 GMT
View Forum Message <> Reply to Message

On Wed, 2006-08-23 at 19:04 -0700, Chandra Seetharaman wrote:
> On Wed, 2006-08-23 at 18:44 -0700, Rohit Seth wrote:

> No, what I mean is that the infrastructure should allow the task moving
> from one group to another, it should also notify the controller about
> that movement and let the controller decide if it wants to take any
> action. (instead of not having the capability stating that it is not
> useful for all type of controllers).
>
> >
Okay.

> > >
> > > >
> > > > > As I mentioned UBC might be perfect for container resource management,
> > > > > but what I am talking for is resource management _without_ a container.
> > > > >
> > > >
> > > > Can you explain that part a bit more?
> > >
> > > Basically I was saying that even though resource management in container
> > > and non-container have mostly same requirements, there are few
> > > requirements that are critical in non-container scenario which are non-
> > > issue in container scenario (for example, moving tasks from one resource
> > > group to another).
> > >
> > > In effect, Design of the infrastructure should not limit non-container
> > > usages.
> > >
> > > IMO, non-container requirements are a superset of container requirements
> > > (resource management purposes only :).
> > >
> >
> > hmm, non-container world (and its resource management part) already
> > exist. And sure those requirements are superset of this discussion.
>
> What do you mean by "resource management part for non-container world
> already exist ?
>
> It does not. CKRM/Resource Groups is trying to do that, but is not in
> Linus's tree.
>

Please, non-container is the environment that exist today in Linux.

Actually cpuset does provide some part of it. But beyond that no.

But then we are all using different terminology like beancounters,
containers, resource groups and now non-containers...

> > And hopefully container support will not break/modify that much.
> >
> > > > >
> > > > > > >
> > > > > > > > - No ability to maintain resource specific data in the controller.
> > > > > > > > it's false. fields can be added to user_beancounter struct easily.
> > > > > > > > and that's what our controllers do.
> > > > > > >
> > > > > > > With the model of static array for resources (struct ubparm ub_parms
> > > > > > > [UB_RESOURCES] in struct user_beancounter), it is not be possible to
> > > > > > > attach _different_ "controller specific" information to each of the
> > > > > > > entries.
> > > > > > >
> > > > > > > I do not think it is good idea to add controller specific information of
> > > > > > > _different_ controllers to the user_beancounter. Think of all the fields
> > > > > > > it will have when all the numproc controller needs is just the basic 3-4
> > > > > > > fields.
> > > > > > >
> > > > > >
> > > > > > IMO it is okay to add the fields whenever necessary as Kirill
> > > > > > suggested. I think once the container subject gets baked a little more,
> > > > > > the controllers will also get tightly coupled.
> > > > >
> > > > > I think my point is not understood. I do not think it is right to add
> > > > > _controller specific_ fields to the generic data structure (struct
> > > > > user_beancounter), as we will end up with a generic data structure which
> > > > > will have so many fields that are not used in so many controllers.
> > > >
> > > A single centralized structure that has fields that are mostly used by
> > > every one should be okay I think.
> >
> > You mean to say definition like
> >
> > struct user_beancounter {
> > fields;/* fields that exists now */
> >
> > int kmemsize_ctlr_info1;
> > char *kmemsize_ctlr_info2;
> >
> > char *oomguar_ctlr_info1;
> > char *oomguar_ctlr_info2;

>
> /* and so on */
> }
>
> is the right thing to do ? even though oomguar controller doesn't care
> about kmemsize_ctlr_info* etc.,
>

No. I think it is appropriate to add all the accounting related fields
and object fields in the core container definition. Controllers only
make decisions based on the information contained in container. And it
should maintain its own data structures if needed(like what Alan said in
one of the later mails).

> >
> > > >
> > > > > >
> > > > > > > - No ability to get the list of tasks belonging to a UBC.
> > > > > > it is not true. it can be read from /proc or system calls interface,
> > > > > > just like the way one finds all tasks belonging to one user :)
> > > > > >
> > > > > > BTW, what is so valueable in this feature?
> > > > > >
> > > > > Again, it may not be useful for container type usages (you can probably
> > > > > get the list from somewhere else, but for resource management it is
> > > > > useful for sysadmins).
> > > > > >
> > > > >
> > > > I'm also debating about whether printing task information is really any
> > > > useful. If a sysadm wants to get information about any particular task
> > > > then that can come from /proc/<pid>/container Though container list
> > > > will be one place where one can easily get the list of all the contained
> > > > tasks (and other resources like files).
> > > >
> > > In non-container environment, there is _no_ /proc/pid/container, as
> > > there is no concept of container :). This will be useful for non-
> > > container scenario.
> > > >
> > >
> > I'm sure when container support gets in then for the above scenario it
> > will read -1 ...
>
> So, how can one get the list of tasks belonging to a resource group in
> that case ?
> >

...and that brings to the starting question...why do you need it?
Commands like ps and top will show appropriate container number for each

task.

-rohit